

Automatic Projector Tilt Compensation System

Ganesh Ajjanagadde Shantanu Jain James Thomas

October 11, 2015

Abstract

We designed a system that corrects the input to a projector if it is tilted so that its output appears unskewed. We read input from a NTSC (National Television System Committee) video camera and store it in an internal block memory. We then process the frame stored in memory using a perspective transformation to pre-warp the image that is sent to the projector via a VGA (Video Graphics Array) signal. The parameters of the perspective transformation are obtained from an accelerometer, which senses two axes of tilt. This allows automatic keystone correction in the two directions sensed by the accelerometer provided the output screen is vertical. Our method also includes options for manual keystone correction to any degree desired, for any projector and screen orientations. For ease of manual correction, we provide the option of using a test pattern (a checkerboard). We also play some useful audio for the percentage of pixels kept by the transformation.

Acknowledgments

First and foremost, we would like to thank our 6.111 instructor Gim Hom for his tremendous patience, experience, and intuition regarding digital systems. This project would never have been possible without him, and countless number of times he saved our group a huge amount of time by some very crucial observations. Not only that, he also played a very important role in our choice of project topic. Initially, we were planning to create some sort of Bitcoin miner. I think it is safe to say that we are all glad that we chose this topic instead on his suggestion.

We would also like to thank all the teaching assistants, lab assistants, and CI-M writing instructors for their very useful suggestions and feedback. We particularly appreciate the guidance of one of our TA's José E. Cruz Serallés for his wealth of experience with Verilog and Xilinx's tools. His project on recursive augmented reality was also very useful as a reference point for certain aspects, such as generation of clocks of different frequencies and the correct timing of sync and blank signals for a 640×480 @60 Hz VGA display. Furthermore, he was able to identify a subtle bug with our memory address computation.

Last, but not least, we would like to thank all our peers in this class. Their thoughtful questions during our proposal presentation raised issues that we had not anticipated. Furthermore, they often were willing to help with some commonly faced issues in the lab, thereby avoiding duplication of effort. We particularly appreciate the note on Piazza (our online discussion forum) by Andres Erbsen regarding test benches and use of Icarus Verilog.

Contents

1	Introduction	5
2	Previous Work	6
3	Module Architecture	6
3.1	Accelerometer interface	7
3.2	Perspective transformation	8
3.3	I/O (Input/Output) interface	8
3.4	Audio system	8
4	Design Decisions	9
4.1	Use of 640×480 @60 Hz VGA	9
4.2	Use of NTSC Camera as Input	9
4.3	Choice of Memory Architecture	9
4.4	Choice of clocks	10
4.5	User interface considerations	10
5	Module Descriptions	11
5.1	par_to_ser (James)	11
5.2	ser_to_par (James)	11
5.3	moving_avg (James)	11
5.4	acc (James)	11
5.5	accel_lut (Ganesh)	14
5.6	pixels_kept (Ganesh)	14
5.7	bram (Ganesh)	15
5.8	addr_map (Ganesh)	15
5.9	slow_clk (Ganesh, James)	15
5.10	move_cursor (Ganesh)	16
5.11	perspective_params (Ganesh)	16
5.12	pixel_map (Ganesh)	18
5.13	ntsc_to_bram (James)	19
5.14	audioManager (Shantanu)	20
5.15	BCD (Shantanu)	21
6	Final Product and User Interface	22
7	Testing And Debugging	23
7.1	Icarus Verilog test benches/verification	24
7.2	Xilinx ModelSim test benches/verification	24
7.3	Julia implementations/tests of algorithms	25
7.4	Labkit I/O	25
7.5	Staring at code/data-sheets	25
7.6	USB input to flash memory issues	26
7.7	Integration tests	26

8	Future Work	27
9	Conclusion	28
A	Source Code	30
A.1	Staff Modules	30
A.1.1	debounce.v	30
A.1.2	delay.v	31
A.1.3	display_16hex.v	31
A.1.4	vga.v	36
A.1.5	divider.v	38
A.1.6	ycrcb2rgb.v	39
A.1.7	ntsc2zbt.v	41
A.1.8	video_decoder.v	43
A.1.9	flash_int.v	67
A.1.10	flash_manager.v	70
A.1.11	test_fsm.v	74
A.1.12	usb_input.v	85
A.1.13	usb_transfer_script.py	88
A.2	Labkit	88
A.2.1	labkit.v	88
A.2.2	labkit.ucf	104
A.3	Our Modules	121
A.3.1	acc.v	121
A.3.2	accel_lut.v	127
A.3.3	accel_lut.jl	216
A.3.4	accel_lut.txt	220
A.3.5	pixels_kept.v	220
A.3.6	bram.v	222
A.3.7	addr_map.v	224
A.3.8	slow_clk.v	224
A.3.9	move_cursor.v	225
A.3.10	perspective_params.v	229
A.3.11	pixel_map.v	234
A.3.12	audioManager.v	238
A.3.13	binaryToDecimal.py	244
A.3.14	BCD.v	245
A.3.15	ClockDivider.v	248
A.3.16	Square.v	249

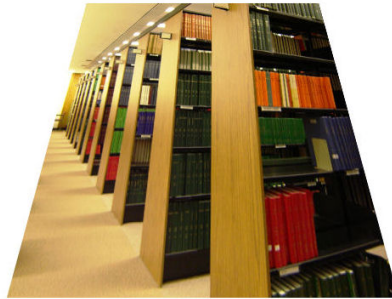
1 Introduction

Due to the advances in semiconductor technology, today’s display projectors can incorporate fairly sophisticated digital processing algorithms for various enhancements to the visual appearance. Moreover, there is an increasing prevalence of portable projectors that benefit from fast, automated setup. One desired functionality is keystone/tilt correction. In other words, even when the projector is tilted, the projector should be able to “pre-warp” the image so that its output appears unskewed. Figure 1 shows what is meant by the keystone effect and what its desired correction should look like ¹.

Figure 1: Keystone Correction In Vertical Direction

With Keystone Correction

Without Keystone Correction



In this project, we project the output of a camera, connect the camera’s output to the FPGA (Field Programmable Gate Array) board, and the FPGA board’s VGA output to the projector. We mount an accelerometer on the projector and measure its signals to determine the projector’s tilt angle on two axes. We then run a perspective transformation algorithm on the FPGA that warps the camera output based on the tilt angles and produces the results at the VGA output for the projector to display. We also provide a manual correction mode so that any desired correction can be achieved. This manual correction mode is exposed to the user via the arrow keys and switches on the FPGA kit. This is of use in mainly two cases:

- The automatic correction obtained using the accelerometer readings is unsatisfactory or inadequate.
- The user desires to correct for projector orientation in the third axis, or in the case when the screen is non-vertical.

¹<https://en.wikipedia.org/wiki/File:Vertical-keystone.jpg>

Finally, we also provide a useful voice output for the percentage of pixels kept after the perspective transformation (this is a lossy transformation in general). The audio is triggered by pressing a button on the FPGA kit.

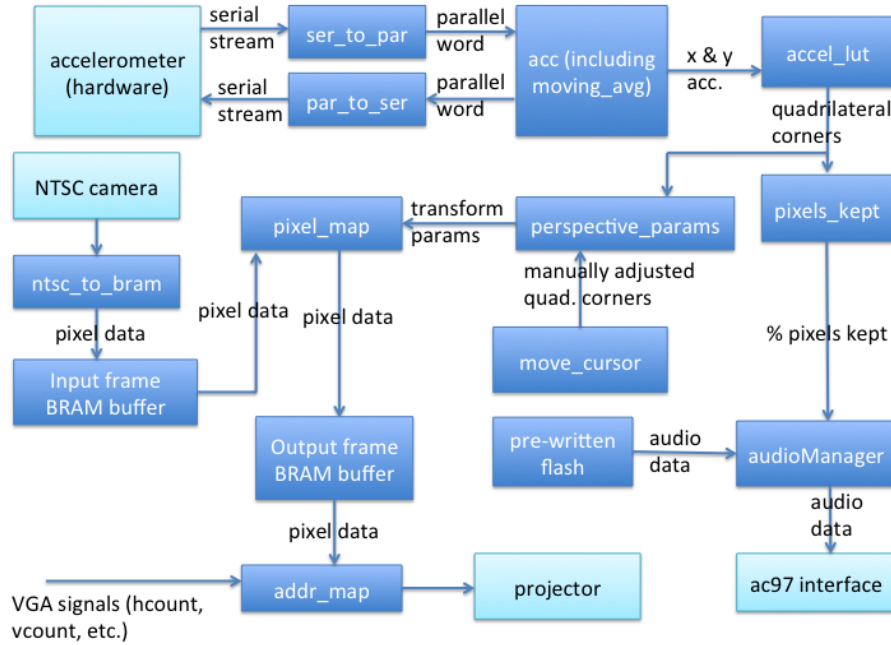
2 Previous Work

Given the practical importance of keystone correction, there has been significant research on doing automatic keystone correction. While Raskar and Beardsley [2001] and Sukthankar et al. [2001] provide a solution to this problem, their methods suffer from one principal weakness, already noted in Li and Sezan [2004], namely that their algorithms are not well suited for implementation in an embedded system. For instance, their methods require solving an 8x8 system of linear equations via Gaussian elimination or similar methods, which is not suitable for embedded platforms, particularly those with slow dividers. Our key technical contribution is the elimination of the Gaussian elimination algorithm, and instead replacing it with closed form solutions for the equations. Moreover, we implement our algorithm on a 6 million gate Xilinx Virtex 2 FPGA, demonstrating that our algorithm is suitable for embedded systems use. The work done in Li and Sezan [2004] subsumes most of this work. However, their system is much more complex, as evidenced by the use of a camera for computing the precise perspective transformation. For the scope of this term final project, we use a simpler accelerometer based approach instead, paying the price of loss of automatic correction along one axis. Note that we still retain fully general manual correction ability.

3 Module Architecture

Figure 2 shows a block diagram including all of the major modules in our system, which are explained further below and in later sections:

Figure 2: Block Diagram



Our keystone correction system can be divided into roughly four functional components:

1. Accelerometer interface
2. Perspective transformation
3. I/O (Input/Output) interface
4. Audio system

Each functional component handles a specific collection of tasks, and is further broken down into a set of one or more modules.

3.1 Accelerometer interface

The accelerometer presents a SPI interface for data transfer. The **par.to.ser** module converts a multi-bit value to a serial stream of bits for transfer to the accelerometer. The **ser.to.par** module converts a serial stream of bits read from the accelerometer to a multi-bit value of desired width. The **moving_avg** module computes the average of 32 accelerometer readings. This is done to counteract some of the noise in the accelerometer readings. The **acc** module

uses the `par_to_ser` and `ser_to_par` modules to initialize the accelerometer and fetch x and y acceleration readings from it in a loop, averaging them using the `moving_avg` module.

3.2 Perspective transformation

The perspective transformation component is where all core computations are performed. The `accel_lut` module accepts the accelerometer readings from the accelerometer as an index into a ROM (read-only memory) containing coordinates of four coordinates of a quadrilateral. The corners of the quadrilateral (after multiplexing with manual override parameters) are fed into the `perspective_params` module, which computes necessary perspective transformation parameters. The perspective parameters are then fed into `pixel_map`, which maps the screen rectangle onto the quadrilateral described above via a perspective transformation. The `pixels_kept` module is a side module that accepts the corners of the quadrilateral, and computes the percentage of pixels kept during the transformation. This is of use in the audio system.

3.3 I/O (Input/Output) interface

The input/output interface to various buttons/switches is contained in the top-level `labkit.v` file. Here, the hardware of the labkit is mapped to the modules, and the functional components are connected together. The relevant buttons/switches are passed to `move_cursor` module, which manipulates the corners of the quadrilateral when the user wishes to manually adjust the correction. The `bram` module implements a very simple 2 port block memory on the FPGA. Two instances of this are created, one called `ntsc_buf` storing the camera input, and one called `vga_buf` containing the processed output (from the perspective transformation). Essentially, we have a pipeline formed: input camera writes to `ntsc_buf`, perspective transformation reads from `ntsc_buf` and writes to `vga_buf`, and the output VGA signal is formed by looping over the `vga_buf`.

3.4 Audio system

The audio system is responsible for receiving, saving, and playing back audio. This requires interacting with several different pieces of hardware and external modules. The top-level module of this functional component is `audioManager`. When triggered by a switch setting, `audioManager` prepares itself for receiving the audio tracks as a continuous stream of data from the external USB input hardware. Because of the difficult-to-implement timing requirements of the external hardware, we use the 6.111 staff provided `usb_input` module to simplify receiving data over USB. As audio data is being received over USB, it is written to the labkit's on-board flash memory. Flash memory is also difficult to use, since a time-sensitive start up sequence needs to be followed to enable proper use of the flash hardware. We made use of the staff provided `flash_manager`, `test_fsm`, and `flash_int` modules. This set of modules greatly abstracts away the

low-level details of working with the flash. When the audio is triggered via an external button input, `audioManager` handles queuing up the appropriate set of tracks for sequential playback. The percentage of pixels used is an external input to `audioManager` from the `pixels_kept` module.

4 Design Decisions

There are several noteworthy design decisions.

4.1 Use of 640×480 @60 Hz VGA

The first of them is the use of a 640×480 @60 Hz VGA display. The use of 640×480 as opposed to the more common higher resolutions found today is due to the memory and computational limitations of our FPGA kit. Initially, we hoped that we could get away with generic code that does not explicitly tie in heavily with the specific screen resolution. Unfortunately, this is not easy, since the choice of resolution influences many different aspects. Chief among these are the sizes of memory vs available memory trade-off, the bit widths of x and y coordinates, the size of the `accel_lut` ROM, and the bit widths of numerous other quantities such as the dividers and the multipliers in `pixel_map` and `perspective_params` respectively. The choice of 60 Hz was made on the basis of its almost guaranteed availability: almost all VGA displays support this refresh rate.

4.2 Use of NTSC Camera as Input

Also related to input/output is our decision to use an NTSC camera feed as the input to the FPGA. Ideally, we would have liked to hook up a computer's VGA signal to the FPGA, so that we can demonstrate the correction system in a more realistic setting. Unfortunately, the labkit in this course has only a single VGA port, ruling out this option.

4.3 Choice of Memory Architecture

Another major design decision made is the choice of memory architecture. We initially planned on using the available 36 Mbits of ZBT memory spread across 2 banks. This would allow us to store at least 4 frames at full 640×480 resolution and 24 bit color (8 bit R, 8 bit G, 8 bit B). Unfortunately, it turns out that ZBT memory is not dual-ported, so read and write on the same bank can't be achieved simultaneously. Since the perspective transform (operating pixel by pixel) turns out to be a huge computational bottleneck, we did not want to waste additional time waiting for read/write on ZBT. Moreover, coordination between the memory banks storing processed and unprocessed data would require some sort of arbiter, adding complexity to our project. Thus, we chose instead to go with the block memory on the FPGA. This can be made into true dual-port

memory, and has a single read/write cycle latency, as opposed to the multi-cycle latency of ZBT memory. However, the amount of block memory available on the FPGA is only 2.5 Mbits. This required sacrifice on image quality. We chose a combination of image down-sampling and color depth reduction. More specifically, we chose a 320x240 sized memory, with 12 bits per line (4 bit R, 4 bit G, 4 bit B). This results in approximately 1 Mbit per buffer, and we use 2 buffers, leaving us with nearly 512k for the accel_lut ROM. That is more than sufficient for our accel_lut. It is certainly possible to squeeze some more color depth, e.g a 14 bit asymmetrical division among R, G, and B. This should help the image quality, but our visual tests indicated no substantial improvement, and hence we omitted this.

4.4 Choice of clocks

A technically noteworthy design decision is our choice of clocks. To avoid clock domain issues, as much as possible we used multiples of a common system clock. Using the Xilinx DCM (Digital Clock Manager), we synthesized a 50 MHz clock from the labkit's standard 27 MHz clock. We used this as a global sys_clk. 640×480 @60Hz needs to be driven at nearly 25 MHz, and was thus obtained by multiplying the time period of sys_clk by 2. The NTSC camera must be driven at around 27 MHz, and the appropriate clock is already generated by the labkit - clock_27mhz. To avoid change of the perspective parameters mid-frame, we also needed a slow_clk signal, i.e one who's time period is on the order of seconds. This can't be accomplished through the use of DCM that easily. Moreover, since actual timing violations for this signal are not really that important, we implemented a simple "flip clock on reaching a count" style "clock divider".

The audio system used the native clock_27mhz for all logic excluding providing data to the AC97 audio codec hardware. The operation of providing audio samples to the AC97 operated on the AC97's external clock. This is required because the AC97 plays back audio at a 48kHz rate, a common audio sampling rate. On spare cycles between the AC97's external clock and the native clock, the system performed all other operations, including track queueing and track switching.

4.5 User interface considerations

An important factor in the quality of the projected image is the percentage of pixels used. The more correction is applied to the image, the more image quality suffers and the dimmer the resulting projection. We wanted to provide the user of the system this information. They could act on this feedback by further adjusting the projector if possible. We realized only power users would be able to make use of this feedback, so we wanted to provide it in a non-intrusive manner - audio was the most natural candidate. The decision to use audio was not without drawbacks. Hearing-impaired users would unfortunately not be able to make use of the feedback. Since the labkit's on-board display

was already used for the manual correction system, there was no other way to display the percentage of pixels used without showing it on the projected screen, which we had earlier eliminated considering it was only useful to power users.

The manual adjustment functionality relies on the user being able to select the corner to be manipulated via a set of switches. While this binary representation would be natural to computer engineers and mathematicians, it would not be self-explanatory to non-technical users. We were again constrained by the labkit's limited hardware input devices from making a more natural interface. Of course, in theory we could have hooked up an external keyboard to another one of the labkit's ports, but this would have increased the complexity of our project.

5 Module Descriptions

In this section, we give a more detailed description of each module, along with the respective contributors.

5.1 `par_to_ser` (James)

This module converts a word of some size W (e.g. `reg[W-1:0]`) to a stream of bits, one at each clock cycle, to be sent to the accelerometer. The higher-order bits are produced first. This module is necessitated by the SPI communication protocol of the accelerometer.

5.2 `ser_to_par` (James)

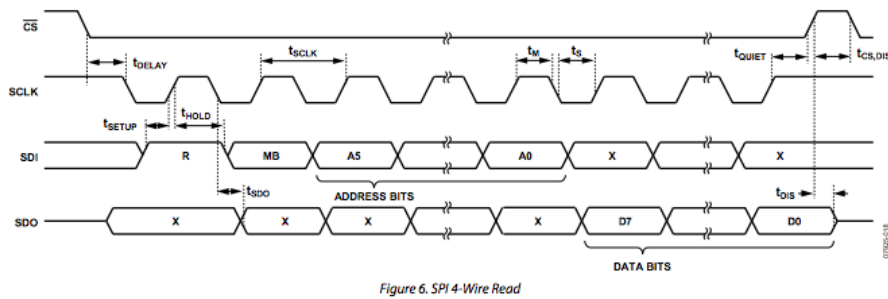
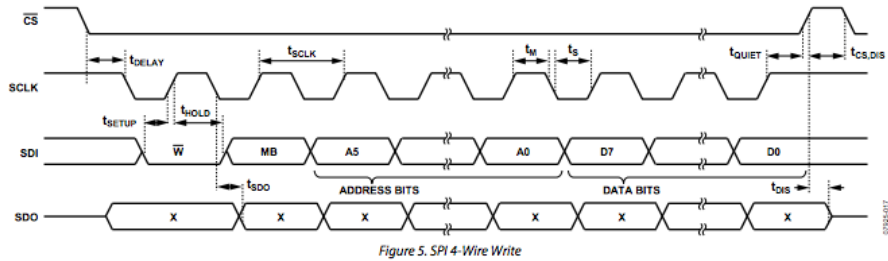
This module converts a stream of bits received from the accelerometer, one per clock cycle, to a word of desired width. It expects that the higher-order bits will be received first.

5.3 `moving_avg` (James)

This module uses a ring buffer to store the last 32 acceleration readings received from the accelerometer for a particular dimension. When a new sample is received, the oldest sample is removed, the sum of the last 32 samples is adjusted accordingly, and the average is recomputed. Since we compute the average over 32 (2^5) samples, we do not need dividers and can use bit shifts for the needed divisions. This module is meant to smooth acceleration readings, which are expected to fluctuate slightly even when the accelerometer is held fixed.

5.4 `acc` (James)

The timing diagram for 4-wire SPI interaction with the accelerometer is shown below:



This module first initializes the accelerometer and then reads x and y acceleration values from it in a loop. The clock used for accelerometer communications is our 50 MHz system clock slowed by a factor of 20, which meets the the max clock frequency spec of 5 MHz. Most of the default configurations for the accelerometer are acceptable for our purposes; the only initialization we need to do is set the measure bit of the POWER_CTL register (each register contains a single byte). Since this register is at address 0x2D, and since we want to perform a single-byte write to it, we first drive the SDI pin with the serial bit stream 8'b00101101 (see Figure 5 in the timing diagram upper two bits for mode and lower six bits for register address). This specifies the mode and register address, and can then be used to drive it with the data to be stored in the register. By default, the accelerometer can measure acceleration values in the range of $\pm 2g$. This is sufficient for our purposes since we care only about static acceleration (which is in the range $\pm 1g$).

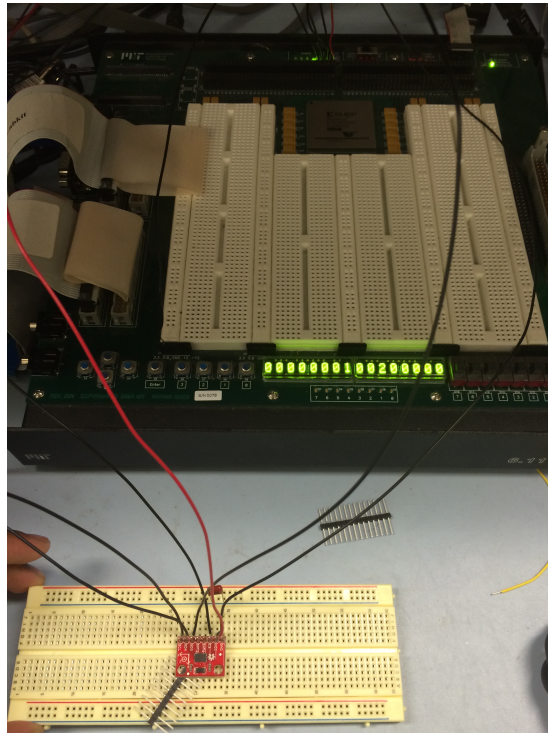
After 16 accelerometer clock cycles have passed, we know that we have sent all of the necessary initialization data to the accelerometer. We then transition our accelerometer state machine to the x-y read loop. Since the x and y acceleration readings are 16 bits (two registers' worth, since the accelerometer gives us 10 bits of precision), we want to perform a multi-byte read of the x and y registers. This can be done in the manner shown in Figure 6 of the timing diagram. We simply assert the R and MB bits, continue driving SDI with the address bits for the first x or first y register (the second registers for x and y immediately follow the first, which is why the multi-byte read of contiguous

registers is acceptable), and then wait 16 additional cycles for all of our data to be produced on the SDO line. We rotate between a state for reading the x registers and a state for reading the y registers. We feed the produced x and y readings into instantiations of the moving_avg filter. The output of the acc module is the output of these filters.

One challenge we faced was getting the accelerometer clock period correct. Even though the spec states that the maximum clock frequency is 5 MHz, if we use a 5 MHz clock and then use a single clock cycle to implement the $t_{CS,DIS}$ waiting period shown in Figures 5 and 6 of the timing diagram, we will not meet the minimum $t_{CS,DIS}$ spec of 250ns. So we had to pick a clock frequency lower than $\frac{1s}{250ns} = 4$ MHz to use a one-cycle wait to meet the $t_{CS,DIS}$ spec. This bug took quite a while to find.

The picture below shows the accelerometer module working in isolation, with averaged accelerometer readings displayed on the hex display:

Figure 3: Accelerometer readings on hex display



5.5 accel_lut (Ganesh)

The `accel_lut` module provides a look-up table from the accelerometer readings in two directions to the four corners of the quadrilateral. The output is synchronized to the global `sys_clk`. The module is essentially implemented as a giant case statement, from which Xilinx’s tools are able to infer a ROM of the appropriate size. Although the data coming out of the accelerometer has 10 bits of precision in each axis, using the full precision would require too much space for the ROM. Instead, we use 6 bits for each of the 2 axes, for a total of 12 bits as an index into the ROM. Each word of the ROM is 76 bits wide. This is because each x coordinate is 10 bits wide (0 to 639), each y coordinate is 9 bits wide (0 to 479), and there are 4 corners. Since each entry corresponds to a choice of quadrilateral corners, and there are $2^{12} = 4096$ entries, manual entry of the look-up table is infeasible. Instead, we wrote a code generator `accel_lut.jl` in the Julia programming language (julia-lang.org). Essentially, the code generator accepts an input CSV (comma separated values) file containing some entries of the desired look-up table obtained via manual testing. The code generator then does a mathematical interpolation to obtain the remaining values of the look-up table, and writes out the `accel_lut.v` file. Of course, it may be argued that the interpolation is not sufficient to get fine grained accuracy. Indeed, our demonstration did show that our look-up table could use some refinement. However, we believe that this is not an inherent deficiency of the approach; it just means that we need more data points and sufficient polynomial degree of the interpolator. Also related to this is another merit to this method: as the user obtains more and more entries, he/she can simply populate the CSV file with them. The quality of interpolation can then only improve.

5.6 pixels_kept (Ganesh)

The `pixels_kept` module accepts the four corners of the quadrilateral, computes its area, and expresses it as a percentage of the total area. The area formula of a quadrilateral in terms of the coordinates of its four corners is a simple determinant expansion, and may be found easily in elementary geometry references. For a quadrilateral with coordinates $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$, its area A is given by the expression:

$$2A = |(x_1 - x_3)(y_2 - y_4) - (y_1 - y_3)(x_2 - x_4)|. \quad (1)$$

The division by 640×480 , and subsequent multiplication by 100 is accomplished without doing either a hardware division or multiplication for efficiency reasons. The key observation here is that $64 \times 48 = 2^{10} \times 3$. Thus, it suffices to figure out how to divide by 3 without performing a hardware division. For this, we approximate 3 by $\frac{21}{64}$, giving us the result correct to the nearest percent. The output is then a 7 bit value expressing the percentage from 0 to 100.

5.7 bram (Ganesh)

The bram module is a very simple true dual port memory module, with exactly enough storage for a down-sampled frame (320x240) with 12 bit color depth per pixel. By convention, the first port is always a write port, triggered by a WE (write enable) signal. The second port is a read port. The ports have their own individual address lines and clock signals in order to achieve the dual ported-ness. The advantages of a dual port memory over a single port memory were clear in our case, and have already been outlined. One more advantage of a dual port memory is that the two ports can be driven at different clocks. The only place where the dual ported abstraction breaks down is when one tries to read and write to the same address simultaneously. It is clear that there is no reasonable behavior in that case. Our design takes this into account, and guarantees that at most one pixel in each frame stores a garbage value. Given the large fraction of displays with at least one defective pixel, this is an extremely minor issue. We initially used Xilinx's proprietary IP Coregen application to synthesize the BRAM (block random access memory) module. However, in order to maximize portability and minimize the use of proprietary files, we studied this a little more and found out that with the appropriate Verilog code, Xilinx's tools will automatically infer the presence of a BRAM. This allowed us to use our own very simple BRAM implementation. It also gave us one additional benefit: Coregen modules often take significantly longer to synthesize as compared to inferred ones.

5.8 addr_map (Ganesh)

The addr_map is a critical single-line module that abstracts away memory address computations, allowing one to address the correct location in memory by providing the x coordinate and y coordinate locations. This is a pure combinational logic module, with no synchronization to any clock. Essentially, it takes in an x coordinate in $[0, 639]$ and a y coordinate in $[0, 479]$, and computes the memory address in the 320x240 line BRAM buffer. This is achieved by taking the integer part of a division by 2 of the x and y coordinates, followed by a standard mapping of a 2 dimensional matrix address to a flat array, an exact analog of manipulating a matrix that has been heap allocated in the C programming language. We unfortunately had a somewhat subtle bug in our memory address computation. At its core, it boils down to the fact that integer division by two followed by a multiplication is not the same as multiplication followed by integer division by two. Fortunately, we tested this early on, and noticed a checkerboard pattern with horizontal stripes on the VGA display instead of a neat checkerboard. The credit for finding that bug goes to our TA José.

5.9 slow_clk (Ganesh, James)

The slow_clk is a simple module that takes in a high frequency clock signal (on the order of MHz) and generates a signal with a much lower frequency.

For us, this was of use in generating a clock with a frequency on the order of a Hz for synchronizing the quadrilateral corner locations and perspective transform parameters. It was also of use in generating a clock signal that met the manufacturer specs for the accelerometer (see the acc section). It is implemented by having a looping counter that would result in an inversion of the clock signal every time the counter hit a certain number of “ticks”. Note that this method is not a robust way of generating an in-phase clock with no skew. It is, however, quite well suited for the task of generating much lower frequency clocks, such as a 1000 times lower frequency. For other applications, such as dividing a clock frequency by a factor of 2, we used Xilinx’s DCM (Digital Clock Manager) instead.

5.10 move_cursor (Ganesh)

The move_cursor module is a user interface module for manipulating the four corners of a quadrilateral. It allows one to move the four corners of a quadrilateral by pressing the four arrow keys and selecting the corner index through the switches on the labkit. To avoid accidental manual adjustment of keystone correction, the movement can only be done when an override switch is on. This module is very similar to the controls used to move the paddle in the Pong game that we designed for Lab 3, and no tricky debugging was needed for this.

5.11 perspective_params (Ganesh)

The perspective_params module lies at the very heart of this project. It accepts a list of the four corners of the quadrilateral $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ and finds the parameters $p_i, 1 \leq i \leq 9$ such that the general perspective transformation:

$$(x, y) \rightarrow \left(\frac{p_1x + p_2y + p_3}{p_7x + p_8y + p_9}, \frac{p_4x + p_5y + p_6}{p_7x + p_8y + p_9} \right) \quad (2)$$

maps the outer coordinates of the screen $(0, 0), (0, 480), (640, 480), (640, 0)$ onto $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ respectively. The module also computes the coefficients $pinv_i$ of the inverse mapping, which is also a perspective transformation (due to the algebraic group structure of perspective transformations which can be checked by direct calculation). The reason for computing the $pinv_i$ is mainly due to our initial goal of having a “memory-less” output transformation, in the sense that given the desired VGA coordinate, we can compute the pre-image of that point. This in turn was due to an inadequate understanding of the FPGA memory and VGA behavior. For instance, we assumed that the VGA frame rate can be controlled simply by adjusting the VGA clock, an assumption that is incorrect. This lack of understanding at one point almost threatened the successful completion of our project, and thus we strongly recommend future 6.111 students to think and discuss very hard the hardware limitations before starting actual implementation work. We were fortunately able to come up with the BRAM design, which although suffers from poorer image quality, nevertheless demonstrates the soundness of our algorithm. It

turns out that the BRAM design can work equally well with either p_i or $pinv_i$. However, at this point of the project, we had already started using $pinv_i$ and did not want to take the risk of going back to p_i . Computing both p_i and $pinv_i$ takes up approximately 80 % of the 144 available 18x18 bit signed multipliers on the FPGA. This may be easily reduced to around 20 % by eliminating the computation of $pinv_i$, something we would have done with additional time for the project.

Note that the p_i (or $pinv_i$) may be scaled by an arbitrary constant, so without loss of generality, we assumed that $p_3 = 1$. However, to avoid needless divisions, in the actual solutions implemented on the FPGA, we scale all parameters to ensure that they are all integers. The closed-form solution to the set of equations (2) (as implemented on the FPGA) is given below:

$$p_7 = 3[(x_1 - x_4)(y_2 - y_3) + (y_1 - y_4)(x_3 - x_2)] \quad (3)$$

$$p_8 = 4[(x_1 - x_2)(y_3 - y_4) + (x_4 - x_3)(y_1 - y_2)] \quad (4)$$

$$d = x_4(y_2 - y_3) + x_2(y_3 - y_4) + x_3(y_4 - y_2) \quad (5)$$

$$p_9 = 1920d \quad (6)$$

$$p_3 = 1920x_1d \quad (7)$$

$$p_6 = 1920y_1d \quad (8)$$

$$p_1 = x_4p_7 + 3(x_4 - x_1)d \quad (9)$$

$$p_2 = x_2p_8 + 4(x_2 - x_1)d \quad (10)$$

$$p_4 = y_4p_7 + 3(y_4 - y_1)d \quad (11)$$

$$p_5 = y_2p_8 + 4(y_2 - y_1)d \quad (12)$$

$$pinv_1 = p_6p_8 - p_5p_9 \quad (13)$$

$$pinv_2 = p_2p_9 - p_3p_8 \quad (14)$$

$$pinv_3 = p_3p_5 - p_2p_6 \quad (15)$$

$$pinv_4 = p_4p_9 - p_6p_7 \quad (16)$$

$$pinv_5 = p_3p_7 - p_1p_9 \quad (17)$$

$$pinv_6 = p_1p_6 - p_3p_4 \quad (18)$$

$$pinv_7 = p_5p_7 - p_4p_8 \quad (19)$$

$$pinv_8 = p_1p_8 - p_2p_7 \quad (20)$$

$$pinv_9 = p_2p_4 - p_1p_5 \quad (21)$$

$$d_x = 639pinv_1 \quad (22)$$

$$d_y = 639pinv_4 \quad (23)$$

$$d_d = 639pinv_7 \quad (24)$$

d_x, d_y, d_d are a couple of parameters used to avoid multiplications in the actual mapping described by the perspective transformation (2). More precisely, they allow clients of this module (such as `pixel_map` in our case) to simply execute a two dimensional loop over the image, incrementing/decrementing the numerator and denominator on each iteration as opposed to performing a fresh

multiplication. Essentially, they correspond to the decrements needed at the end of each horizontal scan line. Note the use of 480, 640 (i.e. $\frac{1920}{4}$ and $\frac{1920}{4}$) as opposed to the more precise 479, 639. The reason for this is that 480, 640 are multiples of sizable powers of two, reducing the bit width needed for certain multiplications required in the solution. Moreover, the difference caused by this slight error can be ignored, due to the continuity of the perspective transformation. Quite crucial to the ease of solving this set of equations symbolically by hand was a good choice of basis. We were fortunate that the “natural basis” in terms of the screen coordinates is actually a pretty good one in that sense.

The outputs of this module are synchronized on a `slow_clk` signal. The rationale for this is two fold:

1. Due to large number of multiplications, even with pipe-lining, it is unlikely that we can meet timing requirements of `sys_clk` (50 MHz).
2. It is undesirable to change the parameters mid-frame anyway, and having a huge latency in changing these parameters may thus in fact be desirable.

5.12 pixel_map (Ganesh)

The `pixel_map` module uses the parameters $pinv_i$ and d_x, d_y, d_d obtained in the `perspective_params` module to do the actual perspective transformation pixel by pixel. At a high level, it is doing a simple loop through x and y through $[0, 639]$ and $[0, 479]$ respectively, computing

$$\left(\frac{pinv_1x + pinv_2y + pinv_3}{pinv_7x + pinv_8y + pinv_9}, \frac{pinv_4x + pinv_5y + pinv_6}{pinv_7x + pinv_8y + pinv_9} \right)$$

at each x and y . It then uses the results to obtain the memory address via `addr_map` in the `ntsc` buffer, and uses another `addr_map` to write the data found in the `ntsc` buffer to the `VGA` buffer. When the coordinates of the inverse transformation are out of range, the module writes a black pixel. A state machine is used to start the divider and keep track of the general state of the computation. The divider used is based on the restoring division algorithm, and was provided by the staff. There was a subtle bug in the staff-provided divider, resulting in incomplete divisions being written out. For us, that translated into always reading an address of 0 of the `ntsc` buffer, leading to a uniform color over the whole frame. Essentially, the bug boiled down to an insufficient width for a counter register in the divider. Tracking down the bug was pretty difficult, since it first required verifying that our `pixel_map` module was providing the correct numerator and denominator. A test bench (runnable under the free software Icarus Verilog simulator) confirmed that there was something wrong with the divider. Another test bench then verified that the results of the division were indeed incorrect. Finally, a close examination of the divider code resulted in the identification of the bug. Once this bug was fixed, the module worked correctly. Unfortunately, the divider needs a width of 79 bits, and the staff provided divider is not pipe-lined. Xilinx’s IP Coregen provides pipe-lined dividers up to a

width of 32 bits, which is insufficient for our needs. Furthermore, they discourage creation of pipe-lined dividers of greater width due to the high area cost. As such, we require 80 clock cycles per pixel, resulting in a frame rate of 1-2 frames per second. Note that with sufficient hardware resources, this may be easily converted into a real-time system.

5.13 ntsc_to_bram (James)

Most of the staff code from the zbt_6111 example was kept intact, including the ADV7185 initialization module and the NTSC decoding module. `ntsc_to_zbt` was converted to `ntsc_to_bram`, which involved a few changes. First off, `ROW_START` and `COL_START` were changed to be 0, since we want our image to fill the entire screen (as opposed to the staff example, where the upper left corner of the image does not start at pixel (0,0)). We expanded all of the registers holding NTSC data to have widths of 30 bits (as opposed to 8), since we want 12-bit color and need the full YCrCb data. We maintained the original approach of reading only scan lines from field 0 and alternately writing them to even and odd screen rows. Since we stored pixels in BRAM, which has a maximum capacity of 2.5 Mbits, and we needed to store a frame from the camera and another frame with the transformed output in addition to the look-up table (discussed above), we only had space to store a 320 by 240 frame with 12 bits of color per pixel. So we cropped the NTSC input by taking only the upper left 320 by 240 rectangle.

Since we now store one pixel per memory location as opposed to four (we were able to size our BRAM so that each location was 12 bits), we write to BRAM whenever the x address is less than 320 and the y address is less than 240 and we are at a write enable positive edge, given by the `we_edge` signal in the code. (In the staff code, we wrote to ZBT only when the x address was a multiple of 4 since we stored 4 pixels per memory location). We determined the BRAM address from the x and y coordinates by treating BRAM as a 2D array with 240 rows and 320 columns laid out in row-major order (row 1 in the first 320 locations, row 2 in the next 320, etc.). So we simply multiplied the y coordinate by 320 and added the x coordinate to get the BRAM address. Finally, since we needed to convert the YCrCb data to RGB using the staff-provided module, which takes 3 clock cycles, we needed to delay the BRAM address and BRAM write enable signals by three clock cycles as well using the staff-provided synchronize module so that we wrote the right data to the right addresses.

Since our BRAM has both read and write ports, we were able to write camera data to the BRAM at the same time that our transformation code fetched pixels from this BRAM, eliminating the need for any coordination mechanisms.

One major challenge in designing this module was eliminating the appearance of randomly colored pixels in the video output. We expected the video to be grainy because we were scaling a 320 by 240 image to 640 by 480 resolution for projection, but the random pixels were unexplained until Gim pointed out that we were writing the BRAM at a different clock rate (the system clock rate)

than the NTSC data was coming to us (the tv_in_line_clock1 rate). This was causing some setup times at the BRAM to be violated when clocks were out of phase and the BRAM to be written with unstable values.

Below is an image of the ntsc_to_bram code working in isolation:

Figure 4: camera output using bram



5.14 audioManager (Shantanu)

The audioManager module is the core module in the audio playback component. It handles the whole life-cycle of audio, from receiving over USB, to storing into flash, queuing tracks, and playback. It interfaces with three external hardware components - the flash chip, the USB interface, and the AC97 audio codec hardware.

The module is centered around its interaction with the flash hardware, since the flash memory is manipulated any time the module is active. Depending on the input of the switches, it either sets the flash memory to “read mode” or “write mode.” To prevent the module from resetting the flash memory and

deleting its contents, which the labkit automatically does when it is powered on, the module also has a “reset disable” switch.

When in “write mode”, the labkit prepares itself to receive data from the usb hardware, via the `usb_input` module provided by the staff. Whenever `usb_input` indicates a new data sample is available, the “dowrite” input to the staff provided `flash_manager` module is asserted to be true, in order to trigger the write operation. The system continues in this cycle until the user’s computer indicates all data has been transmitted, at which time he must exit write mode by manipulating the switches to “read mode.” Due to the limitations of `flash_manager`, data must be written in one session - the user does not have the ability to leave “write mode,” analyze the data on the flash memory, and re-enter “write mode” to continue writing, without first resetting the flash and deleting all the data it contains. As a result, this module inherits these limitations, and so a user may have to manually disable the “reset disable” switch to trigger a reset if his data transfer was to fail. When in “read mode” the flash memory’s contents can be retrieved one location at a time, by setting the read address parameter “`raddr`”. We keep this value at address 0, until we receive the signal to begin audio playback. Based on the percentage of pixels kept, an input signal to the module, the sequence of audio tracks to be played is decided. For example, if eighty-nine percent of pixels are used, four audio tracks are queued for playback: “Eighty,” “Nine,” “Percent,” “Used.” In particular, the number eighty nine is constructed out of two separate sequential tracks instead of a single recording. This system allows us to save memory, reducing transfer time and accumulated error, at the expense of additional complexity in the module.

Having decided what tracks are to be played back, playback begins by calculating the address of the currently playing track, and retrieving the contents of that memory location. While many schemes may be used to calculate the location of the track, we made our recording so that every track is exactly one second long. We then used a simple multiplication of one second of samples by a track index to calculate the track address. Based on the AC97’s external 48kHz clock, we provided a new audio sample to the AC97 to achieve audio playback. After one second of samples had been played back, we calculated the address of the next track queued and began playback. A special end of playback marker called “`UNUSED_TRACK`” halted playback.

5.15 BCD (Shantanu)

This module converts from a binary representation to a decimal representation. It was used within the `audioManager` module to convert the percentage of pixels kept from binary to a decimal representation in order to assign the tracks to be played. This was necessary since digital systems naturally use binary representations, but our audio output was necessarily in decimal.

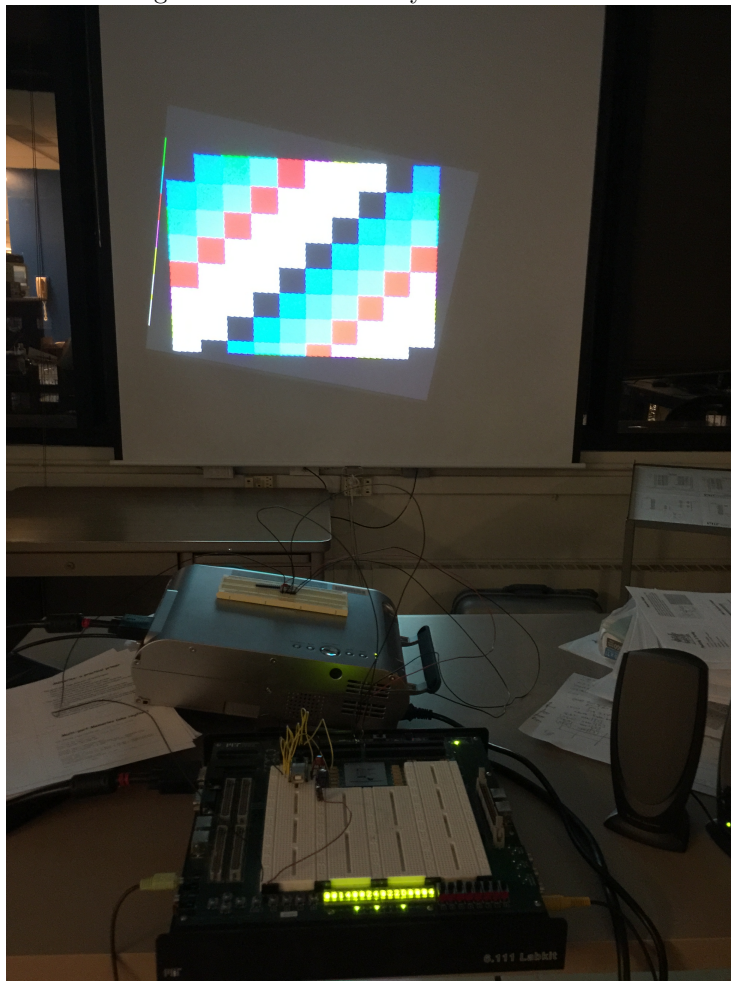
Technically, we opted to implement this module as a look-up table that assigned the decimal ones and tens place based on the input value. The alternative was to use a computational algorithm implemented in Verilog, which was widely available. We used the look-up table because we thought it provided more clar-

ity to the user than the seemingly-opaque efficient algorithms, at the expense of utilizing more of the scarce look-up tables on the hardware.

6 Final Product and User Interface

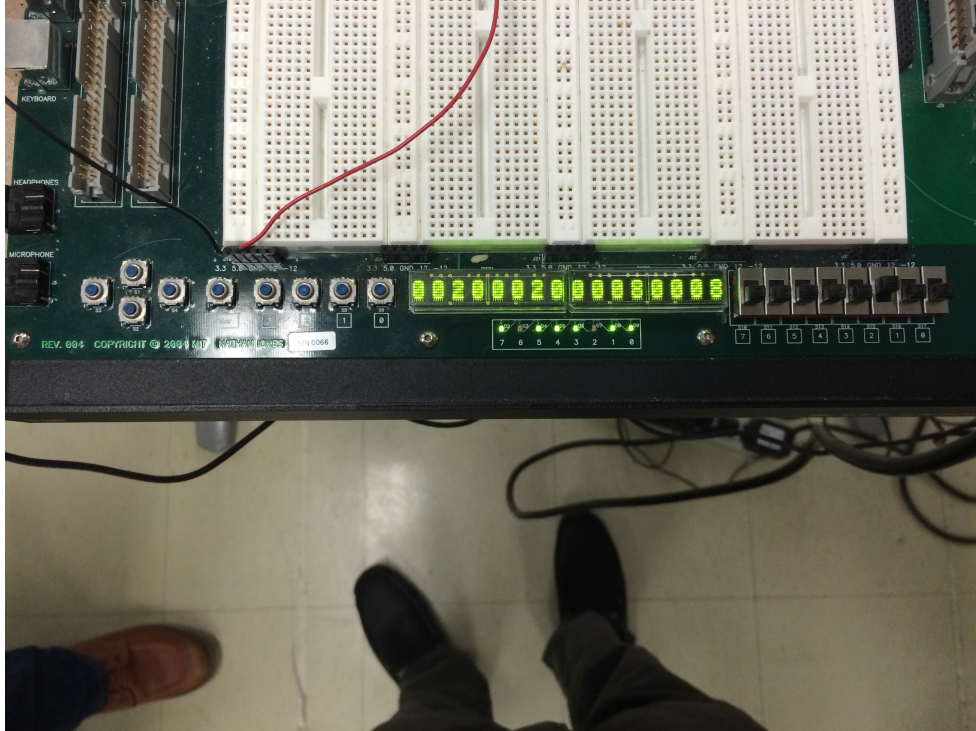
Below is an image of our working system automatically correcting the input to a tilted projector so that it appears rectangular:

Figure 5: Automatic Keystone Correction



Below is an image of the user interface (UI) to our correction system:

Figure 6: User Interface



When pressed, button 0 triggers the audio system, which announces the percentage of screen pixels currently used for the corrected image. LEDs 7-1 show the binary representation of this value. Buttons 2 and 3 adjust the volume of the audio system up or down. The first four digits on the hex display show the current x acceleration value, the next four show the current y acceleration value, the next four show the x coordinate of the corner of the quadrilateral selected by switches 0 and 1, and the last four show the y coordinate of this corner. If switch 7 is high, we are in manual correction mode, and the accelerometer readings are ignored. In this mode, we can manually adjust the corner of the quadrilateral selected by switches 0 and 1 by pressing the up and down buttons (to adjust the x coordinate) and the left and right buttons (to adjust the y coordinate). When switch 5 is high, we display a checkerboard on the screen instead of the NTSC camera input.

7 Testing And Debugging

Testing and debugging Verilog-defined hardware is complicated due to the large synthesis times. Unlike software, where one can change a single line of code, and

trigger an incremental build that can complete on the order of seconds, synthesis of FPGA hardware is a lengthy process, taking on the order of minutes or even hours depending on the amount of logic. Our complete avoidance of Coregen modules definitely served us well in this respect. We also consciously often commented out unnecessary modules. This provided dual benefits:

- Much shorter synthesis times
- Ensuring that the bug is being isolated as much as possible.

Nevertheless, these two steps are certainly not sufficient in efficient debugging. As such, we adopted a multi-pronged testing and debugging approach. Broadly speaking, we used the following methods:

- Icarus Verilog test benches/verification
- Xilinx ModelSim test benches/verification
- Julia implementations/tests of algorithms
- Labkit I/O, e.g led's, hex display, logic analyzer probes
- Staring at code/data-sheets
- Integration tests

7.1 Icarus Verilog test benches/verification

By far the most effective and efficient way of testing simple modules in our experience was the use of Icarus Verilog. We are extremely grateful to a fellow student Andres for posting a note on our online discussion forum (Piazza) regarding the benefits and use of Icarus Verilog. Defining and using test benches through Icarus Verilog is extremely easy and efficient. In our experience, this was most useful for verifying long and complicated chains of combinational logic, such as mathematical algorithms. It was also very useful for checking the syntax of our Verilog code. As an example of its utility, we found a bug with perspective_params that originated from incorrect mixed use of signed and unsigned arithmetic. The main drawbacks of Icarus Verilog are its lack of visualizations of waveforms when invoked on the command line, and its decreasing utility with complex state machines and other sequential logic. The first of these drawbacks is addressed quite well by Xilinx ModelSim.

7.2 Xilinx ModelSim test benches/verification

ModelSim helped us to discover a bug in the staff-provided divider module. We instantiated a 79-bit divider and were not getting correct results. When we examined waveforms in ModelSim, we saw that the ready bit was being asserted far fewer than 79 cycles after we started the division. Upon closer examination of the module, it turned out that the counter the module was using to keep

track of the current bit was only 6 bits wide, which supports only operand bit widths up to 63. We extended the width of this counter to 7 bits to solve this issue.

ModelSim was also used to verify the functionality of the BCD module. After uncovering a bug in the percentage of pixels kept during integration, the first area of investigation was the BCD module, since it was the least tested module. It was trivial to write a test bench for the module and verify the output was correct using ModelSim.

7.3 Julia implementations/tests of algorithms

Prior to implementing the perspective transforms on the FPGA, we definitely wanted to verify the soundness of our approach in software first. We settled on the use of the Julia programming language for this purpose, though in all likelihood MATLAB or Python(+numpy/scipy/matplotlib) would have served just as well. This was by far the most effective means of testing whether an algorithm is correct or not, since one can rely on the benefits of fast debug/iterate cycles in software. This is particularly true in the case of languages featuring a REPL (read, evaluate, print loop) based interpreter, such as Julia. Correcting code is as simple as making some changes, re-including the file in the REPL, and rerunning. This helped us particularly in the verification of the `pixels_kept` and `perspective_params` module. As an aside, to avoid using too many languages for software implementations, we used Julia for our code generator for the `accel_lut` as well. As a concrete example of a serious bug caught using the help of our software implementations, we were able to track and correct an error in the computation of p_1 and p_2 using this method. Essentially, the bug was an interchange of two terms in the closed form expressions that occurred while transcribing the closed form solutions we found for p_1 and p_2 from paper to the computer code.

7.4 Labkit I/O

We wired up the logic analyzer to the accelerometer pins to verify that the `acc` module was correctly initializing the accelerometer (e.g. the `par_to_ser` module was producing the correct serial bit stream for the register address and data). It turned out that the bug in the accelerometer was elsewhere (see the next section).

7.5 Staring at code/data-sheets

This is definitely a questionable inclusion, and we do not recommend this method in general. Nevertheless, it is often effective when done correctly and in the right spirit, since it forces one to rethink and step through the logic again, questioning all assumptions. For instance, once we confirmed that the divider implementation was incorrect, we had to examine the divider code line by line.

It did not take us too long to realize the mistake in the implementation, and looking back we do not see any other way of correcting the mistake.

As described in the section on the acc module, taking a second pass through the data-sheet for the accelerometer helped us to identify an issue with a violated timing constraint ($t_{CS,DIS}$).

7.6 USB input to flash memory issues

The most challenging issue in the audio component of the project was receiving data over USB and storing the samples into flash memory. When we received data over USB, we wrote it immediately to the flash memory. However, after a data transfer, we noticed that many of the bytes were missing.

The flash memory is already difficult to work with because of its tight timing constraints. It is also an old piece of hardware, with read and write times considered slow at the time of manufacture of the labkit over ten years ago. We theorized that the flash memory was too slow to be able to write data at the rate it was being received.

To isolate the device at fault, a series of unit tests were developed. One test took data from a look-up table coded into the device via Verilog and placed it into the flash memory. When we analyzed the flash memory contents, we found it was exactly the same as the look-up table. From this test we were able to conclude that the flash memory was functional. We developed another test that counted the number of bytes received over the USB module before writing them to flash. We found that while the number of bytes received was exactly the number of bytes transmitted from the computer, the number of bytes in memory was still off by a factor between two and three. Even while buffering the data from the USB input device, itself also a buffer, we could not improve the performance of the system.

Despite repeatedly consulting with instructor Gim Hom, teaching assistant Luis Fernandez (who previously implemented a working audio system), and the author of the staff-provided flash_manager module, and in spite of our best efforts to understand and modify the code, we could not improve the performance enough to record all samples. We did improve our system so that the audio was intelligible. With Gim's approval, we closed this issue as not resolvable in the scope of our project and moved forward.

7.7 Integration tests

In addition to testing modules in isolation, we obviously also needed to test whether the modules work together or not. Fortunately, the audio system could be quite easily separated from the rest of the system, so we anticipated easy integration here. Ironically, we faced a strange problem here, which we never understood. The issue was with the percentage value computed by pixels_kept. Initially, pixels_kept was actually named pixels_lost, and was computing 100 minus the percentage of pixels_kept. The audio module was written for percentage of pixels kept however, so initially the audio module did a second subtraction

from 100 to get the correct value. The playback of the percentage was often incorrect for some reason (never identified). Initially, we were keen on fixing the issue from the audio end, since `pixels_lost` was a known correct module. However, we had no luck fixing it from the audio end, so instead we renamed `pixels_lost` to `pixels_kept` and removed the subtraction from 100. This somehow fixed the issue.

Much easier was the integration with the accelerometer. This went flawlessly after we agreed upon the clock frequencies used by the accelerometer and the system. We made sure that the clocks used by the accelerometer and the `accel_lut` (to which the accelerometer readings are sent) were integer multiples of one another to prevent any setup or hold time violations when clocks were out of phase.

Integration of the transformation logic with the memory interface was also extremely smooth, apart from the subtle bug in address computation uncovered by our TA José.

One of the most tedious aspects was the collection of data values for the lookup table. It took around 2 hours of fiddling with the manual correction to get 12 readings for doing the interpolation. Lacking the appropriate measurement equipment, it is difficult to verify that the vertical edges of the checkerboard are exactly parallel to the vertical edges of the screen. Furthermore, we also needed to ensure that the 4:3 aspect ratio was not distorted.

8 Future Work

Although our projector tilt compensation system met most of our initial goals, there are a wide variety of extensions and modifications to the project that we would have liked to implement.

On the image side, we would like to take some steps to improve the quality of the projected image. As noted earlier, one of our key design decisions was the use of reduced resolution (320×240) and reduced color depth (12 bit as opposed to 24 bit). This can be achieved with either a labkit with more dual-ported memory, or our existing labkit with a sophisticated ZBT based memory design (with an arbiter). Another improvement that can be made to the images is the use of some sort of anti-aliasing filter. For instance, when we compute the coordinates in the `pixel_map` module, we only look at the integer part of the division (corresponding to a single pixel index). By looking at the fractional part, we could interpolate the color values at neighboring pixels and use that to reduce the “jagged edges” associated with aliasing in images.

Another significant unknown is how well the accelerometer approach works when the projector is not kept at a fixed distance from the screen. We have not tested whether this would affect the quality of the automatic correction. If this does affect quality, we are confident that by adding an additional distance sensor, we could simply incorporate the distance values into the lookup table and make the automatic correction usable again. Of course, the lookup table could become very large if we do this. There are two avenues around this:

- Store the look-up table in a larger memory, e.g ZBT.
- A more scalable approach is to start doing interpolation in the hardware itself. This will require additional hardware resources and more logic, but the size of the logic will only grow with the number of data points, as opposed to the number of bits coming from the sensors.

Within the audio component, there are many avenues for substantial improvement. The major unresolved issue was the loss of data samples when transferring data over USB and recording it to flash memory. The preferred alternative would be a more modern hardware platform with an updated flash hardware component, as well as an on-board USB module and reference code supplied from the manufacturer. This would likely solve whatever bottleneck stopped us. Another alternative would be to use the serial port instead of the USB hardware, since the labkit already has a serial port built in. This would require a moderately complex Verilog module to implement the serial protocol, although Xilinx may provide some reference code.

Another opportunity for improvement would be variable length tracks. Every track was one second long, which resulted in artificial sounding two digit numbers because some portion of the one second track was silence. A variable length track would reduce the length of silence and sound overall more natural. Our implementation barred us from pursuing this improvement as it requires an end-of-track marker for the track. Since we were randomly losing data samples, we could not reliably determine when the track ended, and so were forced into the fixed-length option.

Another useful feature which we unfortunately did not have time to implement is persistent storage of a user's manual correction settings. After all, no matter how good the `accel_lut` is, it is extremely likely that with certain extreme orientations, the settings obtained through the look-up table are unsatisfactory. One option for dealing with this would be saving manual override settings into some non-volatile storage such as flash, and using them on the next power cycle. Given our issues with compact flash in the audio domain, progress on this front would likely be negligible. Hopefully, with an improved labkit, we would not run into the flash issues.

9 Conclusion

Our projector tilt compensation system used digital logic to successfully perform a perspective transform on an image. This enables fully general manual correction of distortions of the projected image on the screen. Moreover, we also created an automatic correction system for correction of distortion resulting from 2 axes of tilt via the usage of an accelerometer that would sense the angle of tilt along these two axes. We also implemented a useful audio features regarding the percentage of pixels kept by our correction system.

We believe that there were a number of factors that allowed us to succeed with this project. First and foremost, we started early and tackled potential

issues as early as possible, with a consistent time commitment every week. Second, we spent a lot of time doing software simulations/verifications before diving in and writing Verilog. The rationale behind this is that correction and testing in software is much faster than synthesis on the FPGA. Third, we had regular meetings among ourselves to ensure that issues some member of our team had would be addressed collectively.

Overall, we are satisfied with the quality of this system, especially considering the time limitations/scope of this project. We are confident that with some additional work, this system could truly rival commercial offerings in this space.

References

- Baoxin Li and I. Sezan. Automatic keystone correction for smart projectors with embedded camera. In *Image Processing, 2004. ICIP '04. 2004 International Conference on*, volume 4, pages 2829–2832 Vol. 4, Oct 2004. doi: 10.1109/ICIP.2004.1421693.
- Ramesh Raskar and Paul Beardsley. A self-correcting projector. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–504. IEEE, 2001.
- Rahul Sukthankar, Robert G Stockton, and Matthew D Mullin. Smarter presentations: Exploiting homography in camera-projector systems. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 247–253. IEEE, 2001.

A Source Code

Source code for the project may be obtained on GitHub: https://github.com/gajjanag/6111_Project. For completeness, we include all source code here as well. For ease of browsing through the code, we have divided the modules into roughly three categories:

- Staff Modules: modules that are essentially the same as staff provided modules, with minor modifications for our specific needs.
- Labkit: top level labkit module with general instantiations, including clock generators, and ucf file
- Our Modules: modules created by us for our project

A.1 Staff Modules

A.1.1 debounce.v

```
1 // Switch Debounce Module
2 // use your system clock for the clock input
3 // to produce a synchronous, debounced output
4 module debounce #(parameter DELAY=270000) // .01 sec with a 27Mhz clock
5     (input reset, clock, noisy,
6      output reg clean);
7
8     reg [18:0] count;
9     reg new;
10
11     always @(posedge clock)
12         if (reset)
13             begin
14                 count <= 0;
15                 new <= noisy;
16                 clean <= noisy;
17             end
18         else if (noisy != new)
19             begin
20                 new <= noisy;
21                 count <= 0;
22             end
23         else if (count == DELAY)
24             clean <= new;
25         else
26             count <= count+1;
27
28     endmodule
```

A.1.2 delay.v

```
1  `timescale 1ns / 1ps
2  ///////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    23:13:26 12/02/2014
7  // Design Name:
8  // Module Name:    delay
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21 // pulse synchronizer
22 module synchronize #(parameter NSYNC = 2, parameter W = 1) // number of sync flops. must b
23     (input clk, input [W-1:0] in,
24     output reg [W-1:0] out);
25
26     reg [(NSYNC-1)*W-1:0] sync;
27
28     always @ (posedge clk)
29     begin
30         {out, sync} <= {sync[(NSYNC-1)*W-1:0], in};
31     end
32 endmodule
```

A.1.3 display_16hex.v

```
1  ///////////////////////////////////////////////////////////////////
2  //
3  // 6.111 FPGA Labkit -- Hex display driver
4  //
5  // File:    display_16hex.v
6  // Date:    24-Sep-05
7  //
8  // Created: April 27, 2004
9  // Author:  Nathan Ickes
```

```

10 //
11 // 24-Sep-05 Ike: updated to use new reset-once state machine, remove clear
12 // 28-Nov-06 CJT: fixed race condition between CE and RS (thanks Javier!)
13 //
14 // This verilog module drives the labkit hex dot matrix displays, and puts
15 // up 16 hexadecimal digits (8 bytes). These are passed to the module
16 // through a 64 bit wire ("data"), asynchronously.
17 //
18 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
19
20 module display_16hex (reset, clock_27mhz, data,
21                     disp_blank, disp_clock, disp_rs, disp_ce_b,
22                     disp_reset_b, disp_data_out);
23
24     input reset, clock_27mhz;    // clock and reset (active high reset)
25     input [63:0] data;          // 16 hex nibbles to display
26
27     output disp_blank, disp_clock, disp_data_out, disp_rs, disp_ce_b,
28            disp_reset_b;
29
30     reg disp_data_out, disp_rs, disp_ce_b, disp_reset_b;
31
32     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
33     //
34     // Display Clock
35     //
36     // Generate a 500kHz clock for driving the displays.
37     //
38     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
39
40     reg [4:0] count;
41     reg [7:0] reset_count;
42     reg clock;
43     wire dreset;
44
45     always @(posedge clock_27mhz)
46     begin
47         if (reset)
48             begin
49                 count = 0;
50                 clock = 0;
51             end
52         else if (count == 26)
53             begin
54                 clock = ~clock;
55                 count = 5'h00;

```



```

102         dot_index <= 0;
103         state <= state+1;
104     end
105
106 8'h01:
107     begin
108         // End reset
109         disp_reset_b <= 1'b1;
110         state <= state+1;
111     end
112
113 8'h02:
114     begin
115         // Initialize dot register (set all dots to zero)
116         disp_ce_b <= 1'b0;
117         disp_data_out <= 1'b0; // dot_index[0];
118         if (dot_index == 639)
119             state <= state+1;
120         else
121             dot_index <= dot_index+1;
122     end
123
124 8'h03:
125     begin
126         // Latch dot data
127         disp_ce_b <= 1'b1;
128         dot_index <= 31; // re-purpose to init ctrl reg
129         disp_rs <= 1'b1; // Select the control register
130         state <= state+1;
131     end
132
133 8'h04:
134     begin
135         // Setup the control register
136         disp_ce_b <= 1'b0;
137         disp_data_out <= control[31];
138         control <= {control[30:0], 1'b0}; // shift left
139         if (dot_index == 0)
140             state <= state+1;
141         else
142             dot_index <= dot_index-1;
143     end
144
145 8'h05:
146     begin
147         // Latch the control register data / dot data

```

```

148         disp_ce_b <= 1'b1;
149         dot_index <= 39;           // init for single char
150         char_index <= 15;        // start with MS char
151         state <= state+1;
152         disp_rs <= 1'b0;         // Select the dot register
153     end
154
155     8'h06:
156     begin
157         // Load the user's dot data into the dot reg, char by char
158         disp_ce_b <= 1'b0;
159         disp_data_out <= dots[dot_index]; // dot data from msb
160         if (dot_index == 0)
161             if (char_index == 0)
162                 state <= 5;           // all done, latch data
163             else
164                 begin
165                     char_index <= char_index - 1; // goto next char
166                     dot_index <= 39;
167                 end
168             else
169                 dot_index <= dot_index-1; // else loop thru all dots
170         end
171     endcase
172
173     always @ (data or char_index)
174     case (char_index)
175         4'h0: nibble <= data[3:0];
176         4'h1: nibble <= data[7:4];
177         4'h2: nibble <= data[11:8];
178         4'h3: nibble <= data[15:12];
179         4'h4: nibble <= data[19:16];
180         4'h5: nibble <= data[23:20];
181         4'h6: nibble <= data[27:24];
182         4'h7: nibble <= data[31:28];
183         4'h8: nibble <= data[35:32];
184         4'h9: nibble <= data[39:36];
185         4'hA: nibble <= data[43:40];
186         4'hB: nibble <= data[47:44];
187         4'hC: nibble <= data[51:48];
188         4'hD: nibble <= data[55:52];
189         4'hE: nibble <= data[59:56];
190         4'hF: nibble <= data[63:60];
191     endcase
192
193

```



```

69         blank <= next_vblank | (next_hblank & ~hreset);
70     end
71 endmodule
72

```

A.1.5 divider.v

```

1  // The divider module divides one number by another. It
2  // produces a signal named "ready" when the quotient output
3  // is ready, and takes a signal named "start" to indicate
4  // the the input dividend and divider is ready.
5  // sign -- 0 for unsigned, 1 for twos complement
6
7  // It uses a simple restoring divide algorithm.
8  // http://en.wikipedia.org/wiki/Division\_\(digital\)#Restoring\_division
9
10 module divider #(parameter WIDTH = 8)
11     (input clk, sign, start,
12      input [WIDTH-1:0] dividend,
13      input [WIDTH-1:0] divider,
14      output reg [WIDTH-1:0] quotient,
15      output [WIDTH-1:0] remainder,
16      output ready);
17
18     reg [WIDTH-1:0] quotient_temp;
19     reg [WIDTH*2-1:0] dividend_copy, divider_copy, diff;
20     reg negative_output;
21
22     assign remainder = (!negative_output) ?
23         dividend_copy[WIDTH-1:0] : ~dividend_copy[WIDTH-1:0] + 1'b1;
24
25     reg [6:0] bit;
26     reg del_ready = 1;
27     assign ready = (!bit) & ~del_ready;
28
29     wire [WIDTH-2:0] zeros = 0;
30     initial bit = 0;
31     initial negative_output = 0;
32     always @( posedge clk ) begin
33         del_ready <= !bit;
34         if( start ) begin
35
36             bit = WIDTH;
37             quotient = 0;
38             quotient_temp = 0;
39             dividend_copy = (!sign || !dividend[WIDTH-1]) ?

```

```

40         {1'b0,zeros,dividend} :
41         {1'b0,zeros,~dividend + 1'b1};
42     divider_copy = (!sign || !divider[WIDTH-1]) ?
43         {1'b0,divider,zeros} :
44         {1'b0,~divider + 1'b1,zeros};
45
46     negative_output = sign &&
47         ((divider[WIDTH-1] && !dividend[WIDTH-1])
48         ||(!divider[WIDTH-1] && dividend[WIDTH-1]));
49
50     end
51     else if ( bit > 0 ) begin
52         diff = dividend_copy - divider_copy;
53         quotient_temp = quotient_temp << 1;
54         if( !diff[WIDTH*2-1] ) begin
55             dividend_copy = diff;
56             quotient_temp[0] = 1'd1;
57         end
58         quotient = (!negative_output) ?
59             quotient_temp :
60             ~quotient_temp + 1'b1;
61         divider_copy = divider_copy >> 1;
62         bit = bit - 1'b1;
63     end
64 endmodule

```

A.1.6 ycrb2rgb.v

```

1  'timescale 1ns / 1ps
2  ///////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    18:05:46 12/02/2014
7  // Design Name:
8  // Module Name:    ycrb2rgb
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:

```

```

19 //
20 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
21 module ycrcb2rgb ( R, G, B, clk, rst, Y, Cr, Cb );
22
23 output [7:0] R, G, B;
24
25 input clk,rst;
26 input[9:0] Y, Cr, Cb;
27
28 wire [7:0] R,G,B;
29 reg [20:0] R_int,G_int,B_int,X_int,A_int,B1_int,B2_int,C_int;
30 reg [9:0] const1,const2,const3,const4,const5;
31 reg[9:0] Y_reg, Cr_reg, Cb_reg;
32
33 //registering constants
34 always @ (posedge clk)
35 begin
36     const1 = 10'b 0100101010; //1.164 = 01.00101010
37     const2 = 10'b 0110011000; //1.596 = 01.10011000
38     const3 = 10'b 0011010000; //0.813 = 00.11010000
39     const4 = 10'b 0001100100; //0.392 = 00.01100100
40     const5 = 10'b 1000000100; //2.017 = 10.00000100
41 end
42
43 always @ (posedge clk or posedge rst)
44     if (rst)
45         begin
46             Y_reg <= 0; Cr_reg <= 0; Cb_reg <= 0;
47         end
48     else
49         begin
50             Y_reg <= Y; Cr_reg <= Cr; Cb_reg <= Cb;
51         end
52
53 always @ (posedge clk or posedge rst)
54     if (rst)
55         begin
56             A_int <= 0; B1_int <= 0; B2_int <= 0; C_int <= 0; X_int <= 0;
57         end
58     else
59         begin
60             X_int <= (const1 * (Y_reg - 'd64)) ;
61             A_int <= (const2 * (Cr_reg - 'd512));
62             B1_int <= (const3 * (Cr_reg - 'd512));
63             B2_int <= (const4 * (Cb_reg - 'd512));
64             C_int <= (const5 * (Cb_reg - 'd512));

```



```

65     end
66
67 always @ (posedge clk or posedge rst)
68     if (rst)
69         begin
70             R_int <= 0; G_int <= 0; B_int <= 0;
71         end
72     else
73         begin
74             R_int <= X_int + A_int;
75             G_int <= X_int - B1_int - B2_int;
76             B_int <= X_int + C_int;
77         end
78
79
80
81 /*always @ (posedge clk or posedge rst)
82     if (rst)
83         begin
84             R_int <= 0; G_int <= 0; B_int <= 0;
85         end
86     else
87         begin
88             X_int <= (const1 * (Y_reg - 'd64)) ;
89             R_int <= X_int + (const2 * (Cr_reg - 'd512));
90             G_int <= X_int - (const3 * (Cr_reg - 'd512)) - (const4 * (Cb_reg - 'd512));
91             B_int <= X_int + (const5 * (Cb_reg - 'd512));
92         end
93
94 */
95 /* limit output to 0 - 4095, <0 equals 0 and >4095 equals 4095 */
96 assign R = (R_int[20]) ? 0 : (R_int[19:18] == 2'b0) ? R_int[17:10] : 8'b11111111;
97 assign G = (G_int[20]) ? 0 : (G_int[19:18] == 2'b0) ? G_int[17:10] : 8'b11111111;
98 assign B = (B_int[20]) ? 0 : (B_int[19:18] == 2'b0) ? B_int[17:10] : 8'b11111111;
99
100 endmodule

```

A.1.7 ntsc2zbt.v

```

1  /**
2  NOTE: Code borrowed heavily from Pranav Kaundinya, et. al. (2012).
3  */
4
5  module ntsc_to_bram(clk, vclk, fvh, dv, din, ntsc_addr, ntsc_data, ntsc_we, sw);
6
7      input          clk;          // system clock

```

```

8     input          vclk;          // video clock from camera
9     input [2:0]    fvh;
10    input          dv;
11    input [29:0]   din;
12    input          sw;
13    output reg [16:0] ntsc_addr;
14    output reg [11:0] ntsc_data;
15    output          ntsc_we;      // write enable for NTSC data
16    parameter      COL_START = 10'd0;
17    parameter      ROW_START = 10'd0;
18
19    reg [9:0]       col = 0;
20    reg [9:0]       row = 0;
21    reg [29:0]     vdata = 0;
22    reg            vwe;
23    reg            old_dv;
24    reg            old_frame;    // frames are even / odd interlaced
25    reg            even_odd;     // decode interlaced frame to this wire
26
27    wire           frame = fvh[2];
28    wire           frame_edge = frame & ~old_frame;
29
30    always @ (posedge vclk) begin // LLC1 is reference
31
32        old_dv <= dv;
33        vwe <= dv && !fvh[2] & ~old_dv; // if data valid, write it
34
35        old_frame <= frame;
36        even_odd = frame_edge ? ~even_odd : even_odd;
37
38        if (!fvh[2]) begin
39            col <= fvh[0] ? COL_START :
40                (!fvh[2] && !fvh[1] && dv && (col < 1024)) ? col + 1 : col;
41            row <= fvh[1] ? ROW_START :
42                (!fvh[2] && fvh[0] && (row < 768)) ? row + 1 : row;
43            vdata <= (dv && !fvh[2]) ? din : vdata;
44        end
45    end
46
47    // synchronize with system clock
48
49    reg [9:0] x[1:0], y[1:0];
50    reg [29:0] data[1:0];
51    reg        we[1:0];
52    reg        eo[1:0];
53

```

```

54         always @(posedge clk)begin
55
56             {x[1],x[0]} <= {x[0],col};
57             {y[1],y[0]} <= {y[0],row};
58             {data[1],data[0]} <= {data[0],vdata};
59             {we[1],we[0]} <= {we[0],vwe};
60             {eo[1],eo[0]} <= {eo[0],even_odd};
61         end
62
63         // edge detection on write enable signal
64
65         reg old_we;
66         wire we_edge = we[1] & ~old_we;
67         always @(posedge clk) old_we <= we[1];
68
69         // shift each set of four bytes into a large register for the ZBT
70
71         // compute address to store data in
72         wire [9:0] y_addr = {y[1][8:0], eo[1]};
73         wire [9:0] x_addr = x[1];
74
75         wire [7:0] R, G, B;
76         ycrcb2rgb conv( R, G, B, clk, 1'b0, data[1][29:20],
77                     data[1][19:10], data[1][9:0] );
78
79         wire [16:0] myaddr_o = (y_addr[7:0] << 8) + (y_addr[7:0] << 6) + x_addr[8:0];
80         wire [16:0] myaddr;
81         synchronize #(.NSYNC(3), .W(17)) myaddr_sync(clk, myaddr_o, myaddr);
82         // update the output address and data only when four bytes ready
83
84         wire ntsc_we_o = (x_addr < COL_START + 10'd320 && y_addr < ROW_START + 10'd240) &
85
86         synchronize #(.NSYNC(3)) we_sync(clk, ntsc_we_o, ntsc_we);
87         always @(posedge clk)
88             if ( ntsc_we ) begin
89                 ntsc_addr <= myaddr;           // normal and expanded modes
90                 ntsc_data <= ~sw ? {R[7:4], G[7:4], B[7:4]} :
91                     {x_addr[9], 3'b0, x_addr[8], 3'b0, x_addr[7], 3'b0};
92             end
93
94     endmodule // ntsc_to_zbt

```

A.1.8 video_decoder.v

```

1 //
2 // File: video_decoder.v

```

```

3 // Date: 31-Oct-05
4 // Author: J. Castro (MIT 6.111, fall 2005)
5 //
6 // This file contains the ntsc_decode and adv7185init modules
7 //
8 // These modules are used to grab input NTSC video data from the RCA
9 // phono jack on the right hand side of the 6.111 labkit (connect
10 // the camera to the LOWER jack).
11 //
12
13 ///////////////////////////////////////////////////////////////////
14 //
15 // NTSC decode - 16-bit CCIR656 decoder
16 // By Javier Castro
17 // This module takes a stream of LLC data from the adv7185
18 // NTSC/PAL video decoder and generates the corresponding pixels,
19 // that are encoded within the stream, in YCrCb format.
20
21 // Make sure that the adv7185 is set to run in 16-bit LLC2 mode.
22
23 module ntsc_decode(clk, reset, tv_in_ycrbc, ycrbc, f, v, h, data_valid);
24
25 // clk - line-locked clock (in this case, LLC1 which runs at 27Mhz)
26 // reset - system reset
27 // tv_in_ycrbc - 10-bit input from chip. should map to pins [19:10]
28 // ycrbc - 24 bit luminance and chrominance (8 bits each)
29 // f - field: 1 indicates an even field, 0 an odd field
30 // v - vertical sync: 1 means vertical sync
31 // h - horizontal sync: 1 means horizontal sync
32
33 input clk;
34 input reset;
35 input [9:0] tv_in_ycrbc; // modified for 10 bit input - should be P[19:10]
36 output [29:0] ycrbc;
37 output f;
38 output v;
39 output h;
40 output data_valid;
41 // output [4:0] state;
42
43 parameter SYNC_1 = 0;
44 parameter SYNC_2 = 1;
45 parameter SYNC_3 = 2;
46 parameter SAV_f1_cb0 = 3;
47 parameter SAV_f1_y0 = 4;
48 parameter SAV_f1_cr1 = 5;

```

```

49     parameter          SAV_f1_y1 = 6;
50     parameter          EAV_f1 = 7;
51     parameter          SAV_VBI_f1 = 8;
52     parameter          EAV_VBI_f1 = 9;
53     parameter          SAV_f2_cb0 = 10;
54     parameter          SAV_f2_y0 = 11;
55     parameter          SAV_f2_cr1 = 12;
56     parameter          SAV_f2_y1 = 13;
57     parameter          EAV_f2 = 14;
58     parameter          SAV_VBI_f2 = 15;
59     parameter          EAV_VBI_f2 = 16;
60
61
62
63
64     // In the start state, the module doesn't know where
65     // in the sequence of pixels, it is looking.
66
67     // Once we determine where to start, the FSM goes through a normal
68     // sequence of SAV process_YCrCb EAV... repeat
69
70     // The data stream looks as follows
71     // SAV_FF | SAV_00 | SAV_00 | SAV_XY | Cb0 | Y0 | Cr1 | Y1 | Cb2 | Y2 | ... | EAV sequenc
72     // There are two things we need to do:
73     // 1. Find the two SAV blocks (stands for Start Active Video perhaps?)
74     // 2. Decode the subsequent data
75
76     reg [4:0]          current_state = 5'h00;
77     reg [9:0]          y = 10'h000; // luminance
78     reg [9:0]          cr = 10'h000; // chrominance
79     reg [9:0]          cb = 10'h000; // more chrominance
80
81     assign             state = current_state;
82
83     always @ (posedge clk)
84     begin
85         if (reset)
86             begin
87
88                 end
89             else
90                 begin
91                     // these states don't do much except allow us to know where we are in the stream
92                     // whenever the synchronization code is seen, go back to the sync_state before
93                     // transitioning to the new state
94                     case (current_state)

```

```

95         SYNC_1: current_state <= (tv_in_ycrcb == 10'h000) ? SYNC_2 : SYNC_1;
96         SYNC_2: current_state <= (tv_in_ycrcb == 10'h000) ? SYNC_3 : SYNC_1;
97         SYNC_3: current_state <= (tv_in_ycrcb == 10'h200) ? SAV_f1_cb0 :
98             (tv_in_ycrcb == 10'h274) ? EAV_f1 :
99             (tv_in_ycrcb == 10'h2ac) ? SAV_VBI_f1 :
100            (tv_in_ycrcb == 10'h2d8) ? EAV_VBI_f1 :
101            (tv_in_ycrcb == 10'h31c) ? SAV_f2_cb0 :
102            (tv_in_ycrcb == 10'h368) ? EAV_f2 :
103            (tv_in_ycrcb == 10'h3b0) ? SAV_VBI_f2 :
104            (tv_in_ycrcb == 10'h3c4) ? EAV_VBI_f2 : SYNC_1;
105
106         SAV_f1_cb0: current_state <= (tv_in_ycrcb == 10'h3ff) ? SYNC_1 : SAV_f1_y0;
107         SAV_f1_y0: current_state <= (tv_in_ycrcb == 10'h3ff) ? SYNC_1 : SAV_f1_cr1;
108         SAV_f1_cr1: current_state <= (tv_in_ycrcb == 10'h3ff) ? SYNC_1 : SAV_f1_y1;
109         SAV_f1_y1: current_state <= (tv_in_ycrcb == 10'h3ff) ? SYNC_1 : SAV_f1_cb0;
110
111         SAV_f2_cb0: current_state <= (tv_in_ycrcb == 10'h3ff) ? SYNC_1 : SAV_f2_y0;
112         SAV_f2_y0: current_state <= (tv_in_ycrcb == 10'h3ff) ? SYNC_1 : SAV_f2_cr1;
113         SAV_f2_cr1: current_state <= (tv_in_ycrcb == 10'h3ff) ? SYNC_1 : SAV_f2_y1;
114         SAV_f2_y1: current_state <= (tv_in_ycrcb == 10'h3ff) ? SYNC_1 : SAV_f2_cb0;
115
116         // These states are here in the event that we want to cover these signals
117         // in the future. For now, they just send the state machine back to SYNC_1
118         EAV_f1: current_state <= SYNC_1;
119         SAV_VBI_f1: current_state <= SYNC_1;
120         EAV_VBI_f1: current_state <= SYNC_1;
121         EAV_f2: current_state <= SYNC_1;
122         SAV_VBI_f2: current_state <= SYNC_1;
123         EAV_VBI_f2: current_state <= SYNC_1;
124
125         endcase
126     end
127 end // always @ (posedge clk)
128
129 // implement our decoding mechanism
130
131 wire y_enable;
132 wire cr_enable;
133 wire cb_enable;
134
135 // if y is coming in, enable the register
136 // likewise for cr and cb
137 assign y_enable = (current_state == SAV_f1_y0) ||
138                 (current_state == SAV_f1_y1) ||
139                 (current_state == SAV_f2_y0) ||
140                 (current_state == SAV_f2_y1);

```



```

233
234 'define Y_PEAKING_FILTER 3'h4
235 // 0: Composite = 4.5dB, s-video = 9.25dB
236 // 1: Composite = 4.5dB, s-video = 9.25dB
237 // 2: Composite = 4.5dB, s-video = 5.75dB
238 // 3: Composite = 1.25dB, s-video = 3.3dB
239 // 4: Composite = 0.0dB, s-video = 0.0dB
240 // 5: Composite = -1.25dB, s-video = -3.0dB
241 // 6: Composite = -1.75dB, s-video = -8.0dB
242 // 7: Composite = -3.0dB, s-video = -8.0dB
243 'define CORING 2'h0
244 // 0: No coring
245 // 1: Truncate if Y < black+8
246 // 2: Truncate if Y < black+16
247 // 3: Truncate if Y < black+32
248
249 'define ADV7185_REGISTER_2 {3'b000, 'CORING, 'Y_PEAKING_FILTER}
250
251 ////////////////////////////////////
252 // Register 3
253 ////////////////////////////////////
254
255 'define INTERFACE_SELECT 2'h0
256 // 0: Philips-compatible
257 // 1: Broktree API A-compatible
258 // 2: Broktree API B-compatible
259 // 3: [Not valid]
260 'define OUTPUT_FORMAT 4'h0
261 // 0: 10-bit @ LLC, 4:2:2 CCIR656
262 // 1: 20-bit @ LLC, 4:2:2 CCIR656
263 // 2: 16-bit @ LLC, 4:2:2 CCIR656
264 // 3: 8-bit @ LLC, 4:2:2 CCIR656
265 // 4: 12-bit @ LLC, 4:1:1
266 // 5-F: [Not valid]
267 // (Note that the 6.111 labkit hardware provides only a 10-bit interface to
268 // the ADV7185.)
269 'define TRISTATE_OUTPUT_DRIVERS 1'b0
270 // 0: Drivers tristated when ~OE is high
271 // 1: Drivers always tristated
272 'define VBI_ENABLE 1'b0
273 // 0: Decode lines during vertical blanking interval
274 // 1: Decode only active video regions
275
276 'define ADV7185_REGISTER_3 {'VBI_ENABLE, 'TRISTATE_OUTPUT_DRIVERS, 'OUTPUT_FORMAT, 'INTERFA
277
278 ////////////////////////////////////

```

```

279 // Register 4
280 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
281
282 'define OUTPUT_DATA_RANGE                1'b0
283 // 0: Output values restricted to CCIR-compliant range
284 // 1: Use full output range
285 'define BT656_TYPE                       1'b0
286 // 0: BT656-3-compatible
287 // 1: BT656-4-compatible
288
289 'define ADV7185_REGISTER_4 {'BT656_TYPE, 3'b000, 3'b110, 'OUTPUT_DATA_RANGE}
290
291 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
292 // Register 5
293 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
294
295
296 'define GENERAL_PURPOSE_OUTPUTS          4'b0000
297 'define GPO_0_1_ENABLE                  1'b0
298 // 0: General purpose outputs 0 and 1 tristated
299 // 1: General purpose outputs 0 and 1 enabled
300 'define GPO_2_3_ENABLE                   1'b0
301 // 0: General purpose outputs 2 and 3 tristated
302 // 1: General purpose outputs 2 and 3 enabled
303 'define BLANK_CHROMA_IN_VBI              1'b1
304 // 0: Chroma decoded and output during vertical blanking
305 // 1: Chroma blanked during vertical blanking
306 'define HLOCK_ENABLE                     1'b0
307 // 0: GPO 0 is a general purpose output
308 // 1: GPO 0 shows HLOCK status
309
310 'define ADV7185_REGISTER_5 {'HLOCK_ENABLE, 'BLANK_CHROMA_IN_VBI, 'GPO_2_3_ENABLE, 'GPO_0_1_
311
312 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
313 // Register 7
314 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
315
316 'define FIFO_FLAG_MARGIN                  5'h10
317 // Sets the locations where FIFO almost-full and almost-empty flags are set
318 'define FIFO_RESET                       1'b0
319 // 0: Normal operation
320 // 1: Reset FIFO. This bit is automatically cleared
321 'define AUTOMATIC_FIFO_RESET            1'b0
322 // 0: No automatic reset
323 // 1: FIFO is automatically reset at the end of each video field
324 'define FIFO_FLAG_SELF_TIME              1'b1

```

```

325 // 0: FIFO flags are synchronized to CLKIN
326 // 1: FIFO flags are synchronized to internal 27MHz clock
327
328 'define ADV7185_REGISTER_7 {'FIFO_FLAG_SELF_TIME, 'AUTOMATIC_FIFO_RESET, 'FIFO_RESET, 'FIFO
329
330 ////////////////////////////////////////////////////
331 // Register 8
332 ////////////////////////////////////////////////////
333
334 'define INPUT_CONTRAST_ADJUST                                8'h80
335
336 'define ADV7185_REGISTER_8 {'INPUT_CONTRAST_ADJUST}
337
338 ////////////////////////////////////////////////////
339 // Register 9
340 ////////////////////////////////////////////////////
341
342 'define INPUT_SATURATION_ADJUST                            8'h8C
343
344 'define ADV7185_REGISTER_9 {'INPUT_SATURATION_ADJUST}
345
346 ////////////////////////////////////////////////////
347 // Register A
348 ////////////////////////////////////////////////////
349
350 'define INPUT_BRIGHTNESS_ADJUST                          8'h00
351
352 'define ADV7185_REGISTER_A {'INPUT_BRIGHTNESS_ADJUST}
353
354 ////////////////////////////////////////////////////
355 // Register B
356 ////////////////////////////////////////////////////
357
358 'define INPUT_HUE_ADJUST                                  8'h00
359
360 'define ADV7185_REGISTER_B {'INPUT_HUE_ADJUST}
361
362 ////////////////////////////////////////////////////
363 // Register C
364 ////////////////////////////////////////////////////
365
366 'define DEFAULT_VALUE_ENABLE                              1'b0
367 // 0: Use programmed Y, Cr, and Cb values
368 // 1: Use default values
369 'define DEFAULT_VALUE_AUTOMATIC_ENABLE                   1'b0
370 // 0: Use programmed Y, Cr, and Cb values

```

```

371 // 1: Use default values if lock is lost
372 #define DEFAULT_Y_VALUE 6'h0C
373 // Default Y value
374
375 #define ADV7185_REGISTER_C {DEFAULT_Y_VALUE, DEFAULT_VALUE_AUTOMATIC_ENABLE, DEFAULT_VAL
376
377 ////////////////////////////////////////////////////
378 // Register D
379 ////////////////////////////////////////////////////
380
381 #define DEFAULT_CR_VALUE 4'h8
382 // Most-significant four bits of default Cr value
383 #define DEFAULT_CB_VALUE 4'h8
384 // Most-significant four bits of default Cb value
385
386 #define ADV7185_REGISTER_D {DEFAULT_CB_VALUE, DEFAULT_CR_VALUE}
387
388 ////////////////////////////////////////////////////
389 // Register E
390 ////////////////////////////////////////////////////
391
392 #define TEMPORAL_DECIMATION_ENABLE 1'b0
393 // 0: Disable
394 // 1: Enable
395 #define TEMPORAL_DECIMATION_CONTROL 2'h0
396 // 0: Suppress frames, start with even field
397 // 1: Suppress frames, start with odd field
398 // 2: Suppress even fields only
399 // 3: Suppress odd fields only
400 #define TEMPORAL_DECIMATION_RATE 4'h0
401 // 0-F: Number of fields/frames to skip
402
403 #define ADV7185_REGISTER_E {1'b0, TEMPORAL_DECIMATION_RATE, TEMPORAL_DECIMATION_CONTROL,
404
405 ////////////////////////////////////////////////////
406 // Register F
407 ////////////////////////////////////////////////////
408
409 #define POWER_SAVE_CONTROL 2'h0
410 // 0: Full operation
411 // 1: CVBS only
412 // 2: Digital only
413 // 3: Power save mode
414 #define POWER_DOWN_SOURCE_PRIORITY 1'b0
415 // 0: Power-down pin has priority
416 // 1: Power-down control bit has priority

```



```
463 'define ADV7185_REGISTER_14 8'h18
464 'define ADV7185_REGISTER_15 8'h60
465 'define ADV7185_REGISTER_16 8'h00
466 'define ADV7185_REGISTER_17 8'h01
467 'define ADV7185_REGISTER_18 8'h00
468 'define ADV7185_REGISTER_19 8'h10
469 'define ADV7185_REGISTER_1A 8'h10
470 'define ADV7185_REGISTER_1B 8'hF0
471 'define ADV7185_REGISTER_1C 8'h16
472 'define ADV7185_REGISTER_1D 8'h01
473 'define ADV7185_REGISTER_1E 8'h00
474 'define ADV7185_REGISTER_1F 8'h3D
475 'define ADV7185_REGISTER_20 8'hD0
476 'define ADV7185_REGISTER_21 8'h09
477 'define ADV7185_REGISTER_22 8'h8C
478 'define ADV7185_REGISTER_23 8'hE2
479 'define ADV7185_REGISTER_24 8'h1F
480 'define ADV7185_REGISTER_25 8'h07
481 'define ADV7185_REGISTER_26 8'hC2
482 'define ADV7185_REGISTER_27 8'h58
483 'define ADV7185_REGISTER_28 8'h3C
484 'define ADV7185_REGISTER_29 8'h00
485 'define ADV7185_REGISTER_2A 8'h00
486 'define ADV7185_REGISTER_2B 8'hA0
487 'define ADV7185_REGISTER_2C 8'hCE
488 'define ADV7185_REGISTER_2D 8'hF0
489 'define ADV7185_REGISTER_2E 8'h00
490 'define ADV7185_REGISTER_2F 8'hF0
491 'define ADV7185_REGISTER_30 8'h00
492 'define ADV7185_REGISTER_31 8'h70
493 'define ADV7185_REGISTER_32 8'h00
494 'define ADV7185_REGISTER_34 8'h0F
495 'define ADV7185_REGISTER_35 8'h01
496 'define ADV7185_REGISTER_36 8'h00
497 'define ADV7185_REGISTER_37 8'h00
498 'define ADV7185_REGISTER_38 8'h00
499 'define ADV7185_REGISTER_39 8'h00
500 'define ADV7185_REGISTER_3A 8'h00
501 'define ADV7185_REGISTER_3B 8'h00
502
503 'define ADV7185_REGISTER_44 8'h41
504 'define ADV7185_REGISTER_45 8'hBB
505
506 'define ADV7185_REGISTER_F1 8'hEF
507 'define ADV7185_REGISTER_F2 8'h80
508
```

```

509
510 module adv7185init (reset, clock_27mhz, source, tv_in_reset_b,
511                    tv_in_i2c_clock, tv_in_i2c_data);
512
513     input reset;
514     input clock_27mhz;
515     output tv_in_reset_b; // Reset signal to ADV7185
516     output tv_in_i2c_clock; // I2C clock output to ADV7185
517     output tv_in_i2c_data; // I2C data line to ADV7185
518     input source; // 0: composite, 1: s-video
519
520     initial begin
521         $display("ADV7185 Initialization values:");
522         $display(" Register 0: 0x%X", 'ADV7185_REGISTER_0);
523         $display(" Register 1: 0x%X", 'ADV7185_REGISTER_1);
524         $display(" Register 2: 0x%X", 'ADV7185_REGISTER_2);
525         $display(" Register 3: 0x%X", 'ADV7185_REGISTER_3);
526         $display(" Register 4: 0x%X", 'ADV7185_REGISTER_4);
527         $display(" Register 5: 0x%X", 'ADV7185_REGISTER_5);
528         $display(" Register 7: 0x%X", 'ADV7185_REGISTER_7);
529         $display(" Register 8: 0x%X", 'ADV7185_REGISTER_8);
530         $display(" Register 9: 0x%X", 'ADV7185_REGISTER_9);
531         $display(" Register A: 0x%X", 'ADV7185_REGISTER_A);
532         $display(" Register B: 0x%X", 'ADV7185_REGISTER_B);
533         $display(" Register C: 0x%X", 'ADV7185_REGISTER_C);
534         $display(" Register D: 0x%X", 'ADV7185_REGISTER_D);
535         $display(" Register E: 0x%X", 'ADV7185_REGISTER_E);
536         $display(" Register F: 0x%X", 'ADV7185_REGISTER_F);
537         $display(" Register 33: 0x%X", 'ADV7185_REGISTER_33);
538     end
539
540     //
541     // Generate a 1MHz for the I2C driver (resulting I2C clock rate is 250kHz)
542     //
543
544     reg [7:0] clk_div_count, reset_count;
545     reg clock_slow;
546     wire reset_slow;
547
548     initial
549     begin
550         clk_div_count <= 8'h00;
551         // synthesis attribute init of clk_div_count is "00";
552         clock_slow <= 1'b0;
553         // synthesis attribute init of clock_slow is "0";
554     end

```

```

555
556 always @(posedge clock_27mhz)
557     if (clk_div_count == 26)
558         begin
559             clock_slow <= ~clock_slow;
560             clk_div_count <= 0;
561         end
562     else
563         clk_div_count <= clk_div_count+1;
564
565 always @(posedge clock_27mhz)
566     if (reset)
567         reset_count <= 100;
568     else
569         reset_count <= (reset_count==0) ? 0 : reset_count-1;
570
571 assign reset_slow = reset_count != 0;
572
573 //
574 // I2C driver
575 //
576
577 reg load;
578 reg [7:0] data;
579 wire ack, idle;
580
581 i2c i2c(.reset(reset_slow), .clock4x(clock_slow), .data(data), .load(load),
582         .ack(ack), .idle(idle), .scl(tv_in_i2c_clock),
583         .sda(tv_in_i2c_data));
584
585 //
586 // State machine
587 //
588
589 reg [7:0] state;
590 reg tv_in_reset_b;
591 reg old_source;
592
593 always @(posedge clock_slow)
594     if (reset_slow)
595         begin
596             state <= 0;
597             load <= 0;
598             tv_in_reset_b <= 0;
599             old_source <= 0;
600         end

```



```

601     else
602         case (state)
603             8'h00:
604                 begin
605                     // Assert reset
606                     load <= 1'b0;
607                     tv_in_reset_b <= 1'b0;
608                     if (!ack)
609                         state <= state+1;
610                 end
611             8'h01:
612                 state <= state+1;
613             8'h02:
614                 begin
615                     // Release reset
616                     tv_in_reset_b <= 1'b1;
617                     state <= state+1;
618                 end
619             8'h03:
620                 begin
621                     // Send ADV7185 address
622                     data <= 8'h8A;
623                     load <= 1'b1;
624                     if (ack)
625                         state <= state+1;
626                 end
627             8'h04:
628                 begin
629                     // Send subaddress of first register
630                     data <= 8'h00;
631                     if (ack)
632                         state <= state+1;
633                 end
634             8'h05:
635                 begin
636                     // Write to register 0
637                     data <= 'ADV7185_REGISTER_0 | {5'h00, {3{source}}};
638                     if (ack)
639                         state <= state+1;
640                 end
641             8'h06:
642                 begin
643                     // Write to register 1
644                     data <= 'ADV7185_REGISTER_1;
645                     if (ack)
646                         state <= state+1;

```

```

647         end
648     8'h07:
649         begin
650             // Write to register 2
651             data <= 'ADV7185_REGISTER_2;
652             if (ack)
653                 state <= state+1;
654         end
655     8'h08:
656         begin
657             // Write to register 3
658             data <= 'ADV7185_REGISTER_3;
659             if (ack)
660                 state <= state+1;
661         end
662     8'h09:
663         begin
664             // Write to register 4
665             data <= 'ADV7185_REGISTER_4;
666             if (ack)
667                 state <= state+1;
668         end
669     8'h0A:
670         begin
671             // Write to register 5
672             data <= 'ADV7185_REGISTER_5;
673             if (ack)
674                 state <= state+1;
675         end
676     8'h0B:
677         begin
678             // Write to register 6
679             data <= 8'h00; // Reserved register, write all zeros
680             if (ack)
681                 state <= state+1;
682         end
683     8'h0C:
684         begin
685             // Write to register 7
686             data <= 'ADV7185_REGISTER_7;
687             if (ack)
688                 state <= state+1;
689         end
690     8'h0D:
691         begin
692             // Write to register 8

```

```

693         data <= 'ADV7185_REGISTER_8;
694         if (ack)
695             state <= state+1;
696     end
697 8'h0E:
698     begin
699         // Write to register 9
700         data <= 'ADV7185_REGISTER_9;
701         if (ack)
702             state <= state+1;
703     end
704 8'h0F: begin
705         // Write to register A
706         data <= 'ADV7185_REGISTER_A;
707         if (ack)
708             state <= state+1;
709     end
710 8'h10:
711     begin
712         // Write to register B
713         data <= 'ADV7185_REGISTER_B;
714         if (ack)
715             state <= state+1;
716     end
717 8'h11:
718     begin
719         // Write to register C
720         data <= 'ADV7185_REGISTER_C;
721         if (ack)
722             state <= state+1;
723     end
724 8'h12:
725     begin
726         // Write to register D
727         data <= 'ADV7185_REGISTER_D;
728         if (ack)
729             state <= state+1;
730     end
731 8'h13:
732     begin
733         // Write to register E
734         data <= 'ADV7185_REGISTER_E;
735         if (ack)
736             state <= state+1;
737     end
738 8'h14:

```

```

739     begin
740         // Write to register F
741         data <= 'ADV7185_REGISTER_F;
742         if (ack)
743             state <= state+1;
744     end
745 8'h15:
746     begin
747         // Wait for I2C transmitter to finish
748         load <= 1'b0;
749         if (idle)
750             state <= state+1;
751     end
752 8'h16:
753     begin
754         // Write address
755         data <= 8'h8A;
756         load <= 1'b1;
757         if (ack)
758             state <= state+1;
759     end
760 8'h17:
761     begin
762         data <= 8'h33;
763         if (ack)
764             state <= state+1;
765     end
766 8'h18:
767     begin
768         data <= 'ADV7185_REGISTER_33;
769         if (ack)
770             state <= state+1;
771     end
772 8'h19:
773     begin
774         load <= 1'b0;
775         if (idle)
776             state <= state+1;
777     end
778
779 8'h1A: begin
780     data <= 8'h8A;
781     load <= 1'b1;
782     if (ack)
783         state <= state+1;
784 end

```

```

785     8'h1B:
786         begin
787             data <= 8'h33;
788             if (ack)
789                 state <= state+1;
790         end
791     8'h1C:
792         begin
793             load <= 1'b0;
794             if (idle)
795                 state <= state+1;
796         end
797     8'h1D:
798         begin
799             load <= 1'b1;
800             data <= 8'h8B;
801             if (ack)
802                 state <= state+1;
803         end
804     8'h1E:
805         begin
806             data <= 8'hFF;
807             if (ack)
808                 state <= state+1;
809         end
810     8'h1F:
811         begin
812             load <= 1'b0;
813             if (idle)
814                 state <= state+1;
815         end
816     8'h20:
817         begin
818             // Idle
819             if (old_source != source) state <= state+1;
820             old_source <= source;
821         end
822     8'h21: begin
823         // Send ADV7185 address
824         data <= 8'h8A;
825         load <= 1'b1;
826         if (ack) state <= state+1;
827     end
828     8'h22: begin
829         // Send subaddress of register 0
830         data <= 8'h00;

```

```

831         if (ack) state <= state+1;
832     end
833     8'h23: begin
834         // Write to register 0
835         data <= 'ADV7185_REGISTER_0 | {5'h00, {3{source}}};
836         if (ack) state <= state+1;
837     end
838     8'h24: begin
839         // Wait for I2C transmitter to finish
840         load <= 1'b0;
841         if (idle) state <= 8'h20;
842     end
843     endcase
844
845 endmodule
846
847 // i2c module for use with the ADV7185
848
849 module i2c (reset, clock4x, data, load, idle, ack, scl, sda);
850
851     input reset;
852     input clock4x;
853     input [7:0] data;
854     input load;
855     output ack;
856     output idle;
857     output scl;
858     output sda;
859
860     reg [7:0] ldata;
861     reg ack, idle;
862     reg scl;
863     reg sdai;
864
865     reg [7:0] state;
866
867     assign sda = sdai ? 1'bZ : 1'b0;
868
869     always @(posedge clock4x)
870         if (reset)
871             begin
872                 state <= 0;
873                 ack <= 0;
874             end
875         else
876             case (state)

```

```

877      8'h00: // idle
878          begin
879              scl <= 1'b1;
880              sdai <= 1'b1;
881              ack <= 1'b0;
882              idle <= 1'b1;
883              if (load)
884                  begin
885                      ldata <= data;
886                      ack <= 1'b1;
887                      state <= state+1;
888                  end
889              end
890      8'h01: // Start
891          begin
892              ack <= 1'b0;
893              idle <= 1'b0;
894              sdai <= 1'b0;
895              state <= state+1;
896          end
897      8'h02:
898          begin
899              scl <= 1'b0;
900              state <= state+1;
901          end
902      8'h03: // Send bit 7
903          begin
904              ack <= 1'b0;
905              sdai <= ldata[7];
906              state <= state+1;
907          end
908      8'h04:
909          begin
910              scl <= 1'b1;
911              state <= state+1;
912          end
913      8'h05:
914          begin
915              state <= state+1;
916          end
917      8'h06:
918          begin
919              scl <= 1'b0;
920              state <= state+1;
921          end
922      8'h07:

```

```

923         begin
924             sdai <= ldata[6];
925             state <= state+1;
926         end
927     8'h08:
928         begin
929             scl <= 1'b1;
930             state <= state+1;
931         end
932     8'h09:
933         begin
934             state <= state+1;
935         end
936     8'h0A:
937         begin
938             scl <= 1'b0;
939             state <= state+1;
940         end
941     8'h0B:
942         begin
943             sdai <= ldata[5];
944             state <= state+1;
945         end
946     8'h0C:
947         begin
948             scl <= 1'b1;
949             state <= state+1;
950         end
951     8'h0D:
952         begin
953             state <= state+1;
954         end
955     8'h0E:
956         begin
957             scl <= 1'b0;
958             state <= state+1;
959         end
960     8'h0F:
961         begin
962             sdai <= ldata[4];
963             state <= state+1;
964         end
965     8'h10:
966         begin
967             scl <= 1'b1;
968             state <= state+1;

```



```

969         end
970     8'h11:
971         begin
972             state <= state+1;
973         end
974     8'h12:
975         begin
976             scl <= 1'b0;
977             state <= state+1;
978         end
979     8'h13:
980         begin
981             sdai <= ldata[3];
982             state <= state+1;
983         end
984     8'h14:
985         begin
986             scl <= 1'b1;
987             state <= state+1;
988         end
989     8'h15:
990         begin
991             state <= state+1;
992         end
993     8'h16:
994         begin
995             scl <= 1'b0;
996             state <= state+1;
997         end
998     8'h17:
999         begin
1000             sdai <= ldata[2];
1001             state <= state+1;
1002         end
1003     8'h18:
1004         begin
1005             scl <= 1'b1;
1006             state <= state+1;
1007         end
1008     8'h19:
1009         begin
1010             state <= state+1;
1011         end
1012     8'h1A:
1013         begin
1014             scl <= 1'b0;

```

```

1015         state <= state+1;
1016     end
1017 8'h1B:
1018     begin
1019         sdai <= ldata[1];
1020         state <= state+1;
1021     end
1022 8'h1C:
1023     begin
1024         scl <= 1'b1;
1025         state <= state+1;
1026     end
1027 8'h1D:
1028     begin
1029         state <= state+1;
1030     end
1031 8'h1E:
1032     begin
1033         scl <= 1'b0;
1034         state <= state+1;
1035     end
1036 8'h1F:
1037     begin
1038         sdai <= ldata[0];
1039         state <= state+1;
1040     end
1041 8'h20:
1042     begin
1043         scl <= 1'b1;
1044         state <= state+1;
1045     end
1046 8'h21:
1047     begin
1048         state <= state+1;
1049     end
1050 8'h22:
1051     begin
1052         scl <= 1'b0;
1053         state <= state+1;
1054     end
1055 8'h23: // Acknowledge bit
1056     begin
1057         state <= state+1;
1058     end
1059 8'h24:
1060     begin

```

```

1061         scl <= 1'b1;
1062         state <= state+1;
1063     end
1064 8'h25:
1065     begin
1066         state <= state+1;
1067     end
1068 8'h26:
1069     begin
1070         scl <= 1'b0;
1071         if (load)
1072             begin
1073                 ldata <= data;
1074                 ack <= 1'b1;
1075                 state <= 3;
1076             end
1077         else
1078             state <= state+1;
1079         end
1080 8'h27:
1081     begin
1082         sdai <= 1'b0;
1083         state <= state+1;
1084     end
1085 8'h28:
1086     begin
1087         scl <= 1'b1;
1088         state <= state+1;
1089     end
1090 8'h29:
1091     begin
1092         sdai <= 1'b1;
1093         state <= 0;
1094     end
1095 endcase
1096
1097 endmodule
1098
1099

```

A.1.9 flash_int.v

```

1 //flash interface
2 module flash_int(reset, clock, op, address, wdata, rdata, busy, flash_data,
3                 flash_address, flash_ce_b, flash_oe_b, flash_we_b,
4                 flash_reset_b, flash_sts, flash_byte_b);

```

```

5
6     parameter access_cycles = 5;
7     parameter reset_assert_cycles = 1000;
8     parameter reset_recovery_cycles = 30;
9
10    input reset, clock; // Reset and clock for the flash interface
11    input [1:0] op; // Flash operation select (read, write, idle)
12    input [22:0] address;
13    input [15:0] wdata;
14    output [15:0] rdata;
15    output busy;
16    inout [15:0] flash_data;
17    output [23:0] flash_address;
18    output flash_ce_b, flash_oe_b, flash_we_b;
19    output flash_reset_b, flash_byte_b;
20    input flash_sts;
21
22    reg [1:0] lop;
23    reg [15:0] rdata;
24    reg busy;
25    reg [15:0] flash_wdata;
26    reg flash_ddata;
27    reg [23:0] flash_address;
28    reg flash_oe_b, flash_we_b, flash_reset_b;
29
30    assign flash_ce_b = flash_oe_b && flash_we_b;
31    assign flash_byte_b = 1; // 1 = 16-bit mode (A0 ignored)
32
33    assign flash_data = flash_ddata ? flash_wdata : 16'hZ;
34
35    initial
36        flash_reset_b <= 1'b1;
37
38    reg [9:0] state;
39
40    always @(posedge clock)
41        if (reset)
42            begin
43                state <= 0;
44                flash_reset_b <= 0;
45                flash_we_b <= 1;
46                flash_oe_b <= 1;
47                flash_ddata <= 0;
48                busy <= 1;
49            end
50    else if (flash_reset_b == 0)

```

```

51     if (state == reset_assert_cycles)
52         begin
53             flash_reset_b <= 1;
54             state <= 1023-reset_recovery_cycles;
55         end
56     else
57         state <= state+1;
58         else if ((state == 0) && !busy)
59             // The flash chip and this state machine are both idle. Latch the user's
60             // address and write data inputs. Deassert OE and WE, and stop driving
61             // the data buss ourselves. If a flash operation (read or write) is
62             // requested, move to the next state.
63         begin
64             flash_address <= {address, 1'b0};
65             flash_we_b <= 1;
66             flash_oe_b <= 1;
67             flash_ddata <= 0;
68             flash_wdata <= wdata;
69             lop <= op;
70             if (op != 'FLASHOP_IDLE)
71                 begin
72                     busy <= 1;
73                     state <= state+1;
74                 end
75             else
76                 busy <= 0;
77             end
78             else if ((state==0) && flash_sts)
79                 busy <= 0;
80             else if (state == 1)
81                 // The first stage of a flash operation. The address bus is already set,
82                 // so, if this is a read, we assert OE. For a write, we start driving
83                 // the user's data onto the flash databus (the value was latched in the
84                 // previous state.
85                 begin
86                     if (lop == 'FLASHOP_WRITE)
87                         flash_ddata <= 1;
88                     else if (lop == 'FLASHOP_READ)
89                         flash_oe_b <= 0;
90                     state <= state+1;
91                 end
92             else if (state == 2)
93                 // The second stage of a flash operation. Nothing to do for a read. For
94                 // a write, we assert WE.
95                 begin
96                     if (lop == 'FLASHOP_WRITE)

```

```

97             flash_we_b <= 0;
98             state <= state+1;
99         end
100     else if (state == access_cycles+1)
101         // The third stage of a flash operation. For a read, we latch the data
102         // from the flash chip. For a write, we deassert WE.
103         begin
104             if (lop == 'FLASHOP_WRITE)
105                 flash_we_b <= 1;
106             if (lop == 'FLASHOP_READ)
107                 rdata <= flash_data;
108             state <= 0;
109         end
110     else
111         begin
112             if (!flash_sts)
113                 busy <= 1;
114             state <= state+1;
115         end
116     end
117 endmodule

```

A.1.10 flash_manager.v

```

1 //manages all the stuff needed to read and write to the flash ROM
2 module flash_manager(
3     clock, reset,
4     dots,
5     writemode,
6     wdata,
7     dowrite,
8     raddr,
9     frdata,
10    doread,
11    busy,
12    flash_data,
13    flash_address,
14    flash_ce_b,
15    flash_oe_b,
16    flash_we_b,
17    flash_reset_b,
18    flash_sts,
19    flash_byte_b,
20    fsmstate);
21
22    input reset, clock; //clock and reset

```

```

23     output [639:0] dots; //outputs to dot-matrix to help debug flash, no
24     input writemode; //if true then we're in write mode, else we
25     input [15:0] wdata; //data to be written
26     input dowrite; //putting this high tells the manager
27     input [22:0] raddr; //address to read from
28     output[15:0] frdata; //data being read
29     reg[15:0] rdata;
30     input doread; //putting this high tells the manager
31     output busy; //and an output to tell folks we're sti
32     reg busy;
33
34     inout [15:0] flash_data; //direct passthrough
35     output [23:0] flash_address;
36     output flash_ce_b, flash_oe_b, flash_we_b;
37     output flash_reset_b, flash_byte_b;
38     input flash_sts;
39
40     wire flash_busy; //except these, which are internal to the interface
41     wire[15:0] fwdata;
42     wire[15:0] frdata;
43     wire[22:0] address;
44     wire [1:0] op;
45
46     reg [1:0] mode;
47     wire fsm_busy;
48
49     reg[2:0] state; //210
50
51     output[11:0] fsmstate;
52     wire [7:0] fsmstateinv;
53     assign fsmstate = {state,flash_busy,fsm_busy,fsmstateinv[4:0],mode}; //for d
54
55 //this guy
56 flash_int flash(reset, clock, op, address, fwdata, frdata, flash_busy, flash_data, f
57 //and this g
58 test_fsm fsm (reset, clock, op, address, fwdata, frdata, flash_busy, dots, mode, f
59
60 parameter MODE_IDLE = 0;
61 parameter MODE_INIT = 1;
62 parameter MODE_WRITE = 2;
63 parameter MODE_READ = 3;
64
65 parameter HOME = 3'd0;
66 parameter MEM_INIT = 3'd1;
67 parameter MEM_WAIT = 3'd2;
68 parameter WRITE_READY= 3'd3;

```

```

69     parameter WRITE_WAIT          = 3'd4;
70     parameter READ_READY         = 3'd5;
71     parameter READ_WAIT          = 3'd6;
72
73     always @ (posedge clock)
74         if(reset)
75             begin
76                 busy <= 1;
77                 state <= HOME;
78                 mode <= MODE_IDLE;
79             end
80     else begin
81         case(state)
82             HOME://0                                     //we always start h
83                 if(!fsm_busy)
84                     begin
85                         busy <= 0;
86                         if(writemode)
87                             begin
88                                 busy <= 1;
89                                 state <= MEM_INIT;
90                             end
91                         else
92                             begin
93                                 busy <= 1;
94                                 state <= READ_READY;
95                             end
96                         end
97                 else
98                     mode <= MODE_IDLE;
99
100            MEM_INIT://1
101                begin
102                    busy <= 1;
103                    mode <= MODE_INIT;
104                    if(fsm_busy)
105                        state <= MEM_WAIT;
106                end
107
108            MEM_WAIT://2
109                if(!fsm_busy)
110                    begin
111                        busy <= 0;
112                        state <= WRITE_READY;
113                    end
114                else

```



```

115         mode <= MODE_IDLE;
116
117     WRITE_READY://3                                     //wa
118         if(dowrite)
119             begin
120                 busy <= 1;
121                 mode <= MODE_WRITE;
122             end
123         else if(busy)
124             state <= WRITE_WAIT;
125         else if(!writemode)
126             state <= READ_READY;
127
128     WRITE_WAIT://4                                     //waiting for
129         if(!fsm_busy)
130             begin
131                 busy <= 0;
132                 state <= WRITE_READY;
133             end
134         else
135             mode <= MODE_IDLE;
136
137     READ_READY://5                                     //ready to re
138         if(doread)
139             begin
140                 busy <= 1;
141                 mode <= MODE_READ;
142                 if(busy)                                     //l
143                     state <= READ_WAIT;
144             end
145         else
146             busy <= 0;
147
148     READ_WAIT://6                                     //waiting for flash to
149         if(!fsm_busy)
150             begin
151                 busy <= 0;
152                 state <= READ_READY;
153             end
154         else
155             mode <= MODE_IDLE;
156
157     default: begin                                     //should never happen...
158         state <= 3'd7;
159     end
160 endcase

```

```
161         end
162     endmodule
```

A.1.11 test_fsm.v

```
1  'define STATUS_RESET          4'h0
2  'define STATUS_READ_ID       4'h1
3  'define STATUS_CLEAR_LOCKS   4'h2
4  'define STATUS_ERASING       4'h3
5  'define STATUS_WRITING       4'h4
6  'define STATUS_READING       4'h5
7  'define STATUS_SUCCESS       4'h6
8  'define STATUS_BAD_MANUFACTURER 4'h7
9  'define STATUS_BAD_SIZE      4'h8
10 'define STATUS_LOCK_BIT_ERROR 4'h9
11 'define STATUS_ERASE_BLOCK_ERROR 4'hA
12 'define STATUS_WRITE_ERROR    4'hB
13 'define STATUS_READ_WRONG_DATA 4'hC
14
15 'define NUM_BLOCKS 128
16 'define BLOCK_SIZE 64*1024
17 'define LAST_BLOCK_ADDRESS ((NUM_BLOCKS-1)*BLOCK_SIZE)
18 'define LAST_ADDRESS (NUM_BLOCKS*BLOCK_SIZE-1)
19
20 'define FLASHOP_IDLE 2'b00
21 'define FLASHOP_READ 2'b01
22 'define FLASHOP_WRITE 2'b10
23
24 module test_fsm (reset, clock, fop, faddress, fwdata, frdata, fbusy, dots, mode, busy, datain, dataout);
25     input reset, clock;
26     output [1:0] fop;
27     output [22:0] faddress;
28     output [15:0] fwdata;
29     input [15:0] frdata;
30     input fbusy;
31     output [639:0] dots;
32     input [1:0] mode;
33     output busy;
34     input [15:0] datain;
35     input [22:0] addrin;
36     output state;
37
38     reg [1:0] fop;
39     reg [22:0] faddress;
40     reg [15:0] fwdata;
41     reg [639:0] dots;
```

```

42     reg busy;
43     reg [15:0] data_to_store;
44
45     ///////////////////////////////////////////////////////////////////
46     //
47     // State Machine
48     //
49     ///////////////////////////////////////////////////////////////////
50
51     reg [7:0] state;
52     reg [3:0] status;
53
54     parameter MODE_IDLE      = 0;
55     parameter MODE_INIT     = 1;
56     parameter MODE_WRITE    = 2;
57     parameter MODE_READ     = 3;
58
59     parameter MAX_ADDRESS = 23'h200000;
60
61     parameter HOME = 8'h12;
62
63
64     always @(posedge clock)
65         if (reset)
66             begin
67
68                 state <= HOME;
69                 status <= 'STATUS_RESET;
70                 faddress <= 0;
71                 fop <= 'FLASHOP_IDLE;
72                 busy <= 1;
73
74             end
75     else if (!fbusy && (fop == 'FLASHOP_IDLE))
76         case (state)
77
78             HOME://12
79                 case(mode)
80                     MODE_INIT: begin
81                         state <= 8'h00;
82                         busy <= 1;
83                     end
84
85                     MODE_WRITE: begin
86                         state <= 8'h0C;
87                         busy <= 1;
88                     end
89
90             end

```

```

88             MODE_READ: begin
89                 busy <= 1;
90                 if(status == 'STATUS_READING)
91                     state <= 8'h11;
92                 else
93                     state <= 8'h10;
94             end
95
96             default: begin
97                 state <= HOME;
98                 busy <= 0;
99             end
100         endcase
101
102         //////////////////////////////////////
103         // Wipe It
104         //////////////////////////////////////
105         8'h00:
106             begin
107                 // Issue "read id codes" command
108                 status <= 'STATUS_READ_ID;
109                 faddress <= 0;
110                 fwdata <= 16'h0090;
111                 fop <= 'FLASHOP_WRITE;
112                 state <= state+1;
113             end
114
115         8'h01:
116             begin
117                 // Read manufacturer code
118                 faddress <= 0;
119                 fop <= 'FLASHOP_READ;
120                 state <= state+1;
121             end
122
123         8'h02:
124             if (frdata != 16'h0089) // 16'h0089 = Intel
125                 status <= 'STATUS_BAD_MANUFACTURER;
126             else
127                 begin
128                     // Read the device size code
129                     faddress <= 1;
130                     fop <= 'FLASHOP_READ;
131                     state <= state+1;
132                 end
133

```

```

134      8'h03:
135          if (frdata != 16'h0018) // 16'h0018 = 128Mbit
136              status <= 'STATUS_BAD_SIZE;
137          else
138              begin
139                  faddress <= 0;
140                  fwdata <= 16'hFF;
141                  fop <= 'FLASHOP_WRITE;
142                  state <= state+1;
143              end
144
145      8'h04:
146          begin
147              // Issue "clear lock bits" command
148              status <= 'STATUS_CLEAR_LOCKS;
149              faddress <= 0;
150              fwdata <= 16'h60;
151              fop <= 'FLASHOP_WRITE;
152              state <= state+1;
153          end
154
155      8'h05:
156          begin
157              // Issue "confirm clear lock bits" command
158              faddress <= 0;
159              fwdata <= 16'hD0;
160              fop <= 'FLASHOP_WRITE;
161              state <= state+1;
162          end
163
164      8'h06:
165          begin
166              // Read status
167              faddress <= 0;
168              fop <= 'FLASHOP_READ;
169              state <= state+1;
170          end
171
172      8'h07:
173          if (frdata[7] == 1) // Done clearing lock bits
174              if (frdata[6:1] == 0) // No errors
175                  begin
176                      faddress <= 0;
177                      fop <= 'FLASHOP_IDLE;
178                      state <= state+1;
179                  end

```

```

180         else
181             status <= 'STATUS_LOCK_BIT_ERROR;
182     else // Still busy, reread status register
183         begin
184             faddress <= 0;
185             fop <= 'FLASHOP_READ;
186         end
187
188     //////////////////////////////////////
189     // Block Erase Sequence
190     //////////////////////////////////////
191     8'h08:
192         begin
193             status <= 'STATUS_ERASING;
194             fwdata <= 16'h20; // Issue "erase block" command
195             fop <= 'FLASHOP_WRITE;
196             state <= state+1;
197         end
198
199     8'h09:
200         begin
201             fwdata <= 16'hD0; // Issue "confirm erase" command
202             fop <= 'FLASHOP_WRITE;
203             state <= state+1;
204         end
205     8'h0A:
206         begin
207             fop <= 'FLASHOP_READ;
208             state <= state+1;
209         end
210     8'h0B:
211     if (frdata[7] == 1) // Done erasing block
212     if (frdata[6:1] == 0) // No errors
213     if (faddress != MAX_ADDRESS) // 'LAST_BLOCK_ADDRESS)
214     begin
215         faddress <= faddress+'BLOCK_SIZE;
216         fop <= 'FLASHOP_IDLE;
217         state <= state-3;
218     end
219     else
220     begin
221         faddress <= 0;
222         fop <= 'FLASHOP_IDLE;
223         state <= HOME; //done erasing, go home
224     end
225     else // Erase error detected

```

```

226         status <= 'STATUS_ERASE_BLOCK_ERROR;
227     else // Still busy
228         fop <= 'FLASHOP_READ;
229
230     //////////////////////////////////////
231     // Write Mode
232     //////////////////////////////////////
233     8'h0C:
234         begin
235             data_to_store <= datain;
236             status <= 'STATUS_WRITING;
237             fwdata <= 16'h40; // Issue "setup write" command
238             fop <= 'FLASHOP_WRITE;
239             state <= state+1;
240         end
241
242     8'h0D:
243         begin
244             fwdata <= data_to_store; // Finish write
245             fop <= 'FLASHOP_WRITE;
246             state <= state+1;
247         end
248     8'h0E:
249         begin
250             // Read status register
251             fop <= 'FLASHOP_READ;
252             state <= state+1;
253         end
254     8'h0F:
255         if (frdata[7] == 1) // Done writing
256             if (frdata[6:1] == 0) // No errors
257                 if (faddress != 23'h7FFFFFF) // 'LAST_ADDRESS)
258                     begin
259                         faddress <= faddress+1;
260                         fop <= 'FLASHOP_IDLE;
261                         state <= HOME;
262                     end
263                 else
264                     status <= 'STATUS_WRITE_ERROR;
265             else // Write error detected
266                 status <= 'STATUS_WRITE_ERROR;
267             else // Still busy
268                 fop <= 'FLASHOP_READ;
269
270     //////////////////////////////////////
271     // Read Mode INIT

```



```

318         4'hB: nib2char = 40'b01111111_01001001_01001001_01001001_00110110;
319         4'hC: nib2char = 40'b00111110_01000001_01000001_01000001_00100010;
320         4'hD: nib2char = 40'b01111111_01000001_01000001_01000001_00111110;
321         4'hE: nib2char = 40'b01111111_01001001_01001001_01001001_01000001;
322         4'hF: nib2char = 40'b01111111_00001001_00001001_00001001_00000001;
323     endcase
324 end
325 endfunction
326
327 wire [159:0] data_dots;
328 assign data_dots = {nib2char(frdata[15:12]), nib2char(frdata[11:8]),
329                   nib2char(frdata[7:4]), nib2char(frdata[3:0])};
330
331 wire [239:0] address_dots;
332 assign address_dots = {nib2char({ 1'b0, faddress[22:20]}),
333                       nib2char(faddress[19:16]),
334                       nib2char(faddress[15:12]),
335                       nib2char(faddress[11:8]),
336                       nib2char(faddress[7:4]),
337                       nib2char(faddress[3:0])};
338
339 always @(status or address_dots or data_dots)
340     case (status)
341     'STATUS_RESET:
342         dots <= {40'b01111111_00001001_00011001_00101001_01000110, // R
343                 40'b01111111_01001001_01001001_01001001_01000001, // E
344                 40'b00100110_01001001_01001001_01001001_00110010, // S
345                 40'b01111111_01001001_01001001_01001001_01000001, // E
346                 40'b00000001_00000001_01111111_00000001_00000001, // T
347                 40'b00000000_00000000_00000000_00000000_00000000, //
348                 40'b00000000_00000000_00000000_00000000_00000000, //
349                 40'b00000000_00000000_00000000_00000000_00000000, //
350                 40'b00000000_00000000_00000000_00000000_00000000, //
351                 40'b00000000_00000000_00000000_00000000_00000000, //
352                 40'b00001000_00001000_00001000_00001000_00001000, // -
353                 40'b00001000_00001000_00001000_00001000_00001000, // -
354                 40'b00001000_00001000_00001000_00001000_00001000, // -
355                 40'b00001000_00001000_00001000_00001000_00001000, // -
356                 40'b00001000_00001000_00001000_00001000_00001000, // -
357                 40'b00001000_00001000_00001000_00001000_00001000}; // -
358     'STATUS_READ_ID:
359         dots <= {40'b01111111_00001001_00011001_00101001_01000110, // R
360                 40'b01111111_01001001_01001001_01001001_01000001, // E
361                 40'b01111110_00001001_00001001_00001001_01111110, // A
362                 40'b01111111_01000001_01000001_01000001_00111110, // D
363                 40'b00000000_00000000_00000000_00000000_00000000, //

```

```

364         40'b00000000_01000001_01111111_01000001_00000000, // I
365         40'b01111111_01000001_01000001_01000001_00111110, // D
366         40'b00000000_00000000_00000000_00000000_00000000, //
367         40'b00000000_00000000_00000000_00000000_00000000, //
368         40'b00000000_00000000_00000000_00000000_00000000, //
369         address_dots};
370
371     'STATUS_CLEAR_LOCKS:
372     dots <= {40'b00111110_01000001_01000001_01000001_00100010, // C
373             40'b01111111_01000000_01000000_01000000_01000000, // L
374             40'b01111111_00001001_00011001_00101001_01000110, // R
375             40'b00000000_00000000_00000000_00000000_00000000, //
376             40'b01111111_01000000_01000000_01000000_01000000, // L
377             40'b00111110_01000001_01000001_01000001_00111110, // O
378             40'b00111110_01000001_01000001_01000001_00100010, // C
379             40'b01111111_00001000_00010100_00100010_01000001, // K
380             40'b00100110_01001001_01001001_01001001_00110010, // S
381             40'b00000000_00000000_00000000_00000000_00000000, //
382             address_dots};
383
384     'STATUS_ERASING:
385     dots <= {40'b01111111_01001001_01001001_01001001_01000001, // E
386             40'b01111111_00001001_00011001_00101001_01000110, // R
387             40'b01111110_00001001_00001001_00001001_01111110, // A
388             40'b00100110_01001001_01001001_01001001_00110010, // S
389             40'b00000000_01000001_01111111_01000001_00000000, // I
390             40'b01111111_00000010_00000100_00001000_01111111, // N
391             40'b00111110_01000001_01001001_01001001_00111010, // G
392             40'b00000000_00000000_00000000_00000000_00000000, //
393             40'b00000000_00000000_00000000_00000000_00000000, //
394             40'b00000000_00000000_00000000_00000000_00000000, //
395             address_dots};
396
397     'STATUS_WRITING:
398     dots <= {40'b01111111_00100000_00011000_00100000_01111111, // W
399             40'b01111111_00001001_00011001_00101001_01000110, // R
400             40'b00000000_01000001_01111111_01000001_00000000, // I
401             40'b00000001_00000001_01111111_00000001_00000001, // T
402             40'b00000000_01000001_01111111_01000001_00000000, // I
403             40'b01111111_00000010_00000100_00001000_01111111, // N
404             40'b00111110_01000001_01001001_01001001_00111010, // G
405             40'b00000000_00000000_00000000_00000000_00000000, //
406             40'b00000000_00000000_00000000_00000000_00000000, //
407             40'b00000000_00000000_00000000_00000000_00000000, //
408             40'b00000000_00000000_00000000_00000000_00000000, //
409             address_dots};
410
411     'STATUS_READING:
412     dots <= {40'b01111111_00001001_00011001_00101001_01000110, // R
413             40'b01111111_01001001_01001001_01001001_01000001, // E
414             40'b01111110_00001001_00001001_00001001_01111110, // A

```

```

410         40'b01111111_01000001_01000001_01000001_00111110, // D
411         40'b00000000_01000001_01111111_01000001_00000000, // I
412         40'b01111111_00000010_00000100_00001000_01111111, // N
413         40'b00111110_01000001_01001001_01001001_00111010, // G
414         40'b00000000_00000000_00000000_00000000_00000000, //
415         40'b00000000_00000000_00000000_00000000_00000000, //
416         40'b00000000_00000000_00000000_00000000_00000000, //
417         address_dots};
418
419     'STATUS_SUCCESS:
420         dots <= {40'b00000000_00000000_00000000_00000000_00000000, //
421                40'b00101010_00011100_01111111_00011100_00101010, // *
422                40'b00101010_00011100_01111111_00011100_00101010, // *
423                40'b00101010_00011100_01111111_00011100_00101010, // *
424                40'b00000000_00000000_00000000_00000000_00000000, //
425                40'b01111111_00001001_00001001_00001001_00000110, // P
426                40'b01111110_00001001_00001001_00001001_01111110, // A
427                40'b00100110_01001001_01001001_01001001_00110010, // S
428                40'b00100110_01001001_01001001_01001001_00110010, // S
429                40'b01111111_01001001_01001001_01001001_01000001, // E
430                40'b01111111_01000001_01000001_01000001_00111110, // D
431                40'b00000000_00000000_00000000_00000000_00000000, //
432                40'b00101010_00011100_01111111_00011100_00101010, // *
433                40'b00101010_00011100_01111111_00011100_00101010, // *
434                40'b00101010_00011100_01111111_00011100_00101010, // *
435                40'b00000000_00000000_00000000_00000000_00000000}; //
436
437     'STATUS_BAD_MANUFACTURER:
438         dots <= {40'b01111111_01001001_01001001_01001001_01000001, // E
439                40'b01111111_00001001_00011001_00101001_01000110, // R
440                40'b01111111_00001001_00011001_00101001_01000110, // R
441                40'b00000000_00110110_00110110_00000000_00000000, // :
442                40'b00000000_00000000_00000000_00000000_00000000, //
443                40'b01111111_00000010_00001100_00000010_01111111, // M
444                40'b01111110_00001001_00001001_00001001_01111110, // A
445                40'b01111111_00000010_00000100_00001000_01111111, // N
446                40'b01111111_00001001_00001001_00001001_00000001, // U
447                40'b01111111_00001001_00001001_00001001_00000001, // F
448                40'b00000000_00000000_00000000_00000000_00000000, //
449                40'b00000000_00000000_00000000_00000000_00000000, //
450                data_dots};
451
452     'STATUS_BAD_SIZE:
453         dots <= {40'b01111111_01001001_01001001_01001001_01000001, // E
454                40'b01111111_00001001_00011001_00101001_01000110, // R
455                40'b01111111_00001001_00011001_00101001_01000110, // R
456                40'b00000000_00110110_00110110_00000000_00000000, // :
457                40'b00000000_00000000_00000000_00000000_00000000, //
458                40'b00100110_01001001_01001001_01001001_00110010, // S

```

```

456         40'b00000000_01000001_01111111_01000001_00000000, // I
457         40'b01100001_01010001_01001001_01000101_01000011, // Z
458         40'b01111111_01001001_01001001_01001001_01000001, // E
459         40'b00000000_00000000_00000000_00000000_00000000,
460         40'b00000000_00000000_00000000_00000000_00000000,
461         40'b00000000_00000000_00000000_00000000_00000000,
462         data_dots};
463     'STATUS_LOCK_BIT_ERROR:
464         dots <= {40'b01111111_01001001_01001001_01001001_01000001, // E
465         40'b01111111_00001001_00011001_00101001_01000110, // R
466         40'b01111111_00001001_00011001_00101001_01000110, // R
467         40'b00000000_00110110_00110110_00000000_00000000, // :
468         40'b00000000_00000000_00000000_00000000_00000000, //
469         40'b01111111_01000000_01000000_01000000_01000000, // L
470         40'b00111110_01000001_01000001_01000001_00111110, // O
471         40'b00111110_01000001_01000001_01000001_00100010, // C
472         40'b01111111_00001000_00010100_00100010_01000001, // K
473         40'b00100110_01001001_01001001_01001001_00110010, // S
474         40'b00000000_00000000_00000000_00000000_00000000,
475         40'b00000000_00000000_00000000_00000000_00000000,
476         data_dots};
477     'STATUS_ERASE_BLOCK_ERROR:
478         dots <= {40'b01111111_01001001_01001001_01001001_01000001, // E
479         40'b01111111_00001001_00011001_00101001_01000110, // R
480         40'b01111111_00001001_00011001_00101001_01000110, // R
481         40'b00000000_00110110_00110110_00000000_00000000, // :
482         40'b00000000_00000000_00000000_00000000_00000000, //
483         40'b01111111_01001001_01001001_01001001_01000001, // E
484         40'b01111111_00001001_00011001_00101001_01000110, // R
485         40'b01111110_00001001_00001001_00001001_01111110, // A
486         40'b00100110_01001001_01001001_01001001_00110010, // S
487         40'b01111111_01001001_01001001_01001001_01000001, // E
488         40'b00000000_00000000_00000000_00000000_00000000,
489         40'b00000000_00000000_00000000_00000000_00000000,
490         data_dots};
491     'STATUS_WRITE_ERROR:
492         dots <= {40'b01111111_01001001_01001001_01001001_01000001, // E
493         40'b01111111_00001001_00011001_00101001_01000110, // R
494         40'b01111111_00001001_00011001_00101001_01000110, // R
495         40'b00000000_00110110_00110110_00000000_00000000, // :
496         40'b00000000_00000000_00000000_00000000_00000000, //
497         40'b01111111_00100000_00011000_00100000_01111111, // W
498         40'b01111111_00001001_00011001_00101001_01000110, // R
499         40'b00000000_01000001_01111111_01000001_00000000, // I
500         40'b00000001_00000001_01111111_00000001_00000001, // T
501         40'b01111111_01001001_01001001_01001001_01000001, // E

```

```

502         40'b00000000_00000000_00000000_00000000_00000000,
503         40'b00000000_00000000_00000000_00000000_00000000,
504         data_dots};
505     'STATUS_READ_WRONG_DATA:
506         dots <= {40'b01111111_01001001_01001001_01001001_01000001, // E
507                 40'b01111111_00001001_00011001_00101001_01000110, // R
508                 40'b01111111_00001001_00011001_00101001_01000110, // R
509                 40'b00000000_00110110_00110110_00000000_00000000, // :
510                 40'b00000000_00000000_00000000_00000000_00000000,
511                 address_dots,
512                 40'b00000000_00000000_00000000_00000000_00000000,
513                 data_dots};
514     default:
515         dots <= {16{40'b01010101_00101010_01010101_00101010_01010101}};
516 endcase
517
518 endmodule

```

A.1.12 usb_input.v

```

1  //reads data and puts it on out
2  module usb_input(clk,reset,data,rd,rxf,out,newout,hold,state);
3      input clk, reset;           //clock and reset
4      input [7:0] data;          //the data pins from the USB fifo
5      input rxf;                 //the rxf pin from the USB fifo
6      output rd;                 //the rd pin from the USB fifo
7      reg rd;
8
9      output[7:0] out;           //this is where data goes when it has been read from the fi
10     reg[7:0] out;
11     output newout;              //when this is high, out contains a new chunk of data
12     reg newout;
13     input hold;                 //as long as hold is high, this module sits
14                                 //still module and will not accept new data
15
16     output state;               //for debugging purposes
17     reg[3:0] state;
18
19     parameter RESET              = 0;           //state data
20     parameter WAIT                = 1;
21     parameter WAIT2               = 2;
22     parameter WAIT3               = 3;
23     parameter DATA_COMING        = 4;
24     parameter DATA_COMING_2      = 5;
25     parameter DATA_COMING_3      = 6;
26     parameter DATA_COMING_4      = 7;

```

```

27     parameter DATA_COMING_5      = 8;
28     parameter DATA_HERE          = 9;
29     parameter DATA_LEAVING       =10;
30     parameter DATA_LEAVING_2=11;
31     parameter DATA_LEAVING_3=12;
32     parameter DATA_LEAVING_4=13;
33     parameter DATA_LEAVING_5=14;
34     parameter DATA_LEAVING_6=15;
35
36     initial
37         state <= WAIT;
38
39     always @ (posedge clk)
40         if(reset)
41             begin
42                 newout <= 0;
43                 rd <= 1;                                     //we can't read data
44                 state <= WAIT;
45             end
46         else
47             if(~hold)
48                 begin
49                     newout <= 0;
50                     case(state)
51                         WAIT:
52                             if(~rxf)                       //if rxf is low and
53                                 begin
54                                     rd <= 1;
55                                     state <= WAIT2;         //and
56                                 end
57                             WAIT2:
58                                 if(~rxf)                   //double check
59                                     begin
60                                         rd <= 1;
61                                         state <= WAIT3;
62                                     end
63                                 else
64                                     state <= WAIT;
65                             WAIT3:
66                                 if(~rxf)                   //and triple check
67                                     begin
68                                         rd <= 0;
69                                         state <= DATA_COMING;
70                                     end
71                                 end
72

```

```

73         else
74             state <= WAIT;
75
76     DATA_COMING:           //once rd goes low we go
77         state <= DATA_COMING_2;
78
79     DATA_COMING_2:
80         state <= DATA_COMING_3;
81
82     DATA_COMING_3:
83         state <= DATA_HERE;
84
85     DATA_HERE:
86     begin
87         out <= data;           //the data is v
88         state <= DATA_LEAVING;
89         newout <= 1;         //let folks know
90     end
91
92     DATA_LEAVING:           //wait a cycle
93     begin
94         //rd <= 1; // ORIGINAL
95         state <= DATA_LEAVING_2;
96         newout <= 0;         //let folks know
97     end
98
99     DATA_LEAVING_2:           //wait another cycle
100    state <= DATA_LEAVING_3;
101
102    DATA_LEAVING_3:           //wait another cycle
103    state <= DATA_LEAVING_4;
104
105    DATA_LEAVING_4:           //wait another cycle
106    state <= DATA_LEAVING_5;
107
108    DATA_LEAVING_5:           //wait another cycle
109    state <= DATA_LEAVING_6;
110
111    DATA_LEAVING_6:           //wait another cycle
112    begin
113        state <= WAIT;
114        rd <= 1;
115    end
116    default:
117        state <= WAIT;
118    endcase

```

```
119         end
120     endmodule
```

A.1.13 usb_transfer_script.py

```
1  #!/usr/bin/env python
2
3  import serial
4  import wave
5  import struct
6  import scipy
7
8
9  '''
10 6.111 USB transfer script
11 Luis Fernandez
12
13 Script runs through a coe file (basically row after row of 8 bit values) and
14 sends line by line.
15 '''
16
17 ser = serial.Serial(port='/dev/tty.usbserial-FTDHKA57')
18
19 a = open('audio_convert/Fa48k8bit.coe', 'r')
20
21 for line in a:
22
23     line = line.rstrip()[0:-1]
24     line = int(line, base=2)
25
26     b = struct.pack("<H", line)
27
28     r = ser.write(b[0])
29
30 ser.close()
```

A.2 Labkit

A.2.1 labkit.v

```
1  'default_nettype none
2  //////////////////////////////////////
3  //
4  // 6.111 FPGA Labkit -- Template Toplevel Module
5  //
6  // For Labkit Revision 004
```



```

7 //
8 //
9 // Created: October 31, 2004, from revision 003 file
10 // Author: Nathan Ickes
11 //
12 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
13 //
14 // CHANGES FOR BOARD REVISION 004
15 //
16 // 1) Added signals for logic analyzer pods 2-4.
17 // 2) Expanded "tv_in_ycrcb" to 20 bits.
18 // 3) Renamed "tv_out_data" to "tv_out_i2c_data" and "tv_out_sclk" to
19 //     "tv_out_i2c_clock".
20 // 4) Reversed disp_data_in and disp_data_out signals, so that "out" is an
21 //     output of the FPGA, and "in" is an input.
22 //
23 // CHANGES FOR BOARD REVISION 003
24 //
25 // 1) Combined flash chip enables into a single signal, flash_ce_b.
26 //
27 // CHANGES FOR BOARD REVISION 002
28 //
29 // 1) Added SRAM clock feedback path input and output
30 // 2) Renamed "mousedata" to "mouse_data"
31 // 3) Renamed some ZBT memory signals. Parity bits are now incorporated into
32 //     the data bus, and the byte write enables have been combined into the
33 //     4-bit ram#_bwe_b bus.
34 // 4) Removed the "systemace_clock" net, since the SystemACE clock is now
35 //     hardwired on the PCB to the oscillator.
36 //
37 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
38 //
39 // Complete change history (including bug fixes)
40 //
41 // 2006-Mar-08: Corrected default assignments to "vga_out_red", "vga_out_green"
42 //              and "vga_out_blue". (Was 10'h0, now 8'h0.)
43 //
44 // 2005-Sep-09: Added missing default assignments to "ac97_sdata_out",
45 //              "disp_data_out", "analyzer[2-3]_clock" and
46 //              "analyzer[2-3]_data".
47 //
48 // 2005-Jan-23: Reduced flash address bus to 24 bits, to match 128Mb devices
49 //              actually populated on the boards. (The boards support up to
50 //              256Mb devices, with 25 address lines.)
51 //
52 // 2004-Oct-31: Adapted to new revision 004 board.

```

```

53 //
54 // 2004-May-01: Changed "disp_data_in" to be an output, and gave it a default
55 //             value. (Previous versions of this file declared this port to
56 //             be an input.)
57 //
58 // 2004-Apr-29: Reduced SRAM address busses to 19 bits, to match 18Mb devices
59 //             actually populated on the boards. (The boards support up to
60 //             72Mb devices, with 21 address lines.)
61 //
62 // 2004-Apr-29: Change history started
63 //
64 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
65
66 module labkit (beep, audio_reset_b, ac97_sdata_out, ac97_sdata_in, ac97_sync,
67               ac97_bit_clock,
68
69               vga_out_red, vga_out_green, vga_out_blue, vga_out_sync_b,
70               vga_out_blank_b, vga_out_pixel_clock, vga_out_hsync,
71               vga_out_vsync,
72
73               tv_out_ycrfb, tv_out_reset_b, tv_out_clock, tv_out_i2c_clock,
74               tv_out_i2c_data, tv_out_pal_ntsc, tv_out_hsync_b,
75               tv_out_vsync_b, tv_out_blank_b, tv_out_subcar_reset,
76
77               tv_in_ycrfb, tv_in_data_valid, tv_in_line_clock1,
78               tv_in_line_clock2, tv_in_aef, tv_in_hff, tv_in_aff,
79               tv_in_i2c_clock, tv_in_i2c_data, tv_in_fifo_read,
80               tv_in_fifo_clock, tv_in_iso, tv_in_reset_b, tv_in_clock,
81
82               ram0_data, ram0_address, ram0_adv_ld, ram0_clk, ram0_cen_b,
83               ram0_ce_b, ram0_oe_b, ram0_we_b, ram0_bwe_b,
84
85               ram1_data, ram1_address, ram1_adv_ld, ram1_clk, ram1_cen_b,
86               ram1_ce_b, ram1_oe_b, ram1_we_b, ram1_bwe_b,
87
88               clock_feedback_out, clock_feedback_in,
89
90               flash_data, flash_address, flash_ce_b, flash_oe_b, flash_we_b,
91               flash_reset_b, flash_sts, flash_byte_b,
92
93               rs232_txd, rs232_rxd, rs232_rts, rs232_cts,
94
95               mouse_clock, mouse_data, keyboard_clock, keyboard_data,
96
97               clock_27mhz, clock1, clock2,
98

```

```

99         disp_blank, disp_data_out, disp_clock, disp_rs, disp_ce_b,
100         disp_reset_b, disp_data_in,
101
102         button0, button1, button2, button3, button_enter, button_right,
103         button_left, button_down, button_up,
104
105         switch,
106
107         led,
108
109         user1, user2, user3, user4,
110
111         daughtercard,
112
113         systemace_data, systemace_address, systemace_ce_b,
114         systemace_we_b, systemace_oe_b, systemace_irq, systemace_mpbrdy,
115
116         analyzer1_data, analyzer1_clock,
117         analyzer2_data, analyzer2_clock,
118         analyzer3_data, analyzer3_clock,
119         analyzer4_data, analyzer4_clock);
120
121     output beep, audio_reset_b, ac97_synch, ac97_sdata_out;
122     input  ac97_bit_clock, ac97_sdata_in;
123
124     output [7:0] vga_out_red, vga_out_green, vga_out_blue;
125     output vga_out_sync_b, vga_out_blank_b, vga_out_pixel_clock,
126            vga_out_hsync, vga_out_vsync;
127
128     output [9:0] tv_out_ycrcb;
129     output tv_out_reset_b, tv_out_clock, tv_out_i2c_clock, tv_out_i2c_data,
130            tv_out_pal_ntsc, tv_out_hsync_b, tv_out_vsync_b, tv_out_blank_b,
131            tv_out_subcar_reset;
132
133     input  [19:0] tv_in_ycrcb;
134     input  tv_in_data_valid, tv_in_line_clock1, tv_in_line_clock2, tv_in_aef,
135            tv_in_hff, tv_in_aff;
136     output tv_in_i2c_clock, tv_in_fifo_read, tv_in_fifo_clock, tv_in_iso,
137            tv_in_reset_b, tv_in_clock;
138     inout  tv_in_i2c_data;
139
140     inout  [35:0] ram0_data;
141     output [18:0] ram0_address;
142     output ram0_adv_ld, ram0_clk, ram0_cen_b, ram0_ce_b, ram0_oe_b, ram0_we_b;
143     output [3:0] ram0_bwe_b;
144

```

```

145     inout  [35:0] ram1_data;
146     output [18:0] ram1_address;
147     output ram1_adv_ld, ram1_clk, ram1_cen_b, ram1_ce_b, ram1_oe_b, ram1_we_b;
148     output [3:0] ram1_bwe_b;
149
150     input  clock_feedback_in;
151     output clock_feedback_out;
152
153     inout  [15:0] flash_data;
154     output [23:0] flash_address;
155     output flash_ce_b, flash_oe_b, flash_we_b, flash_reset_b, flash_byte_b;
156     input  flash_sts;
157
158     output rs232_txd, rs232_rts;
159     input  rs232_rxd, rs232_cts;
160
161     input  mouse_clock, mouse_data, keyboard_clock, keyboard_data;
162
163     input  clock_27mhz, clock1, clock2;
164
165     output disp_blank, disp_clock, disp_rs, disp_ce_b, disp_reset_b;
166     input  disp_data_in;
167     output disp_data_out;
168
169     input button0, button1, button2, button3, button_enter, button_right,
170           button_left, button_down, button_up;
171     input [7:0] switch;
172     output [7:0] led;
173
174     inout [31:0] user1, user2, user3, user4;
175
176     inout [43:0] daughtercard;
177
178     inout [15:0] systemace_data;
179     output [6:0] systemace_address;
180     output systemace_ce_b, systemace_we_b, systemace_oe_b;
181     input  systemace_irq, systemace_mpbrdy;
182
183     output [15:0] analyzer1_data, analyzer2_data, analyzer3_data,
184               analyzer4_data;
185     output analyzer1_clock, analyzer2_clock, analyzer3_clock, analyzer4_clock;
186
187     //////////////////////////////////////
188     //
189     // I/O Assignments
190     //

```



```

237     assign ram1_cen_b = 1'b1;
238     assign ram1_ce_b = 1'b1;
239     assign ram1_oe_b = 1'b1;
240     assign ram1_we_b = 1'b1;
241     assign ram1_bwe_b = 4'hF;
242     assign clock_feedback_out = 1'b0;
243     // clock_feedback_in is an input
244
245     // Flash ROM
246     // assign flash_data = 16'hZ;
247     // assign flash_address = 24'h0;
248     // assign flash_ce_b = 1'b1;
249     // assign flash_oe_b = 1'b1;
250     // assign flash_we_b = 1'b1;
251     // assign flash_reset_b = 1'b0;
252     // assign flash_byte_b = 1'b1;
253     // // flash_sts is an input
254
255     // RS-232 Interface
256     assign rs232_txd = 1'b1;
257     assign rs232_rts = 1'b1;
258     // rs232_rxd and rs232_cts are inputs
259
260     // PS/2 Ports
261     // mouse_clock, mouse_data, keyboard_clock, and keyboard_data are inputs
262
263
264     // Buttons, Switches, and Individual LEDs
265     // assign led = 8'hFF;
266     // button0, button1, button2, button3, button_enter, button_right,
267     // button_left, button_down, button_up, and switches are inputs
268
269     // User I/Os
270     assign user1[21:4] = 28'hZ;
271     assign user2 = 32'hZ;
272     assign user3 = 32'hZ;
273     assign user4 = 32'hZ;
274
275     // Daughtercard Connectors
276     assign daughtercard = 44'hZ;
277
278     // SystemACE Microprocessor Port
279     assign systemace_data = 16'hZ;
280     assign systemace_address = 7'h0;
281     assign systemace_ce_b = 1'b1;
282     assign systemace_we_b = 1'b1;

```

```

283     assign systemace_oe_b = 1'b1;
284     // systemace_irq and systemace_mpbrdy are inputs
285
286     // Logic Analyzer
287     assign analyzer1_data = 16'h0;
288     assign analyzer1_clock = 1'b1;
289     assign analyzer2_data = 16'h0;
290     assign analyzer2_clock = 1'b1;
291     assign analyzer3_data = 16'h0;
292     assign analyzer3_clock = 1'b1;
293     assign analyzer4_data = 16'h0;
294     assign analyzer4_clock = 1'b1;
295
296     ////////////////////////////////////////////////////////////////////
297     //
298     // Reset Generation
299     //
300     // A shift register primitive is used to generate an active-high reset
301     // signal that remains high for 16 clock cycles after configuration finishes
302     // and the FPGA's internal clocks begin toggling.
303     //
304     ////////////////////////////////////////////////////////////////////
305     wire reset;
306     SRL16 reset_sr(.D(1'b0), .CLK(clock_27mhz), .Q(reset),
307                 .A0(1'b1), .A1(1'b1), .A2(1'b1), .A3(1'b1));
308     defparam reset_sr.INIT = 16'hFFFF;
309
310     ////////////////////////////////////////////////////////////////////
311     // create clocks
312     // use FPGA's digital clock manager to produce a 50 MHz clock
313     // this clock is our system clock
314     // to drive VGA at 640x480 (60 Hz), we need a 25 MHz vga clock
315     // credits to Jose for computing the required clock values
316     // and use of ramclock module
317     ////////////////////////////////////////////////////////////////////
318     wire sys_clk_unbuf, sys_clk, vga_clk, vga_clk_unbuf;
319     wire slow_clk;
320     wire clk_locked;
321     DCM vclk1(.CLKIN(clock_27mhz), .CLKFX(sys_clk_unbuf));
322     // synthesis attribute CLKFX_DIVIDE of vclk1 is 15
323     // synthesis attribute CLKFX_MULTIPLY of vclk1 is 28
324     // synthesis attribute CLK_FEEDBACK of vclk1 is NONE
325     // synthesis attribute CLKIN_PERIOD of vclk1 is 37
326     BUFG vclk2(.O(sys_clk), .I(sys_clk_unbuf));
327     DCM int_dcm(.CLKIN(sys_clk), .CLKFX(vga_clk_unbuf), .LOCKED(clk_locked));
328     // synthesis attribute CLKFX_DIVIDE of int_dcm is 4

```



```

559 ntsc_to_bram n2b (tv_in_line_clock1, tv_in_line_clock1, fvh, dv,
560                 ycrb, ntsc_addr, ntsc_data, ntsc_we, switch[6]);
561
562 // dump a checkerboard into "ntsc" buffer
563 reg[9:0] cur_x = 0;
564 reg[9:0] cur_y = 0;
565 wire[2:0] checkerboard;
566 assign checkerboard = cur_x[7:5] + cur_y[7:5];
567 assign ntsc_cb_din = {{4{checkerboard[2]}}, {4{checkerboard[1]}}, {4{checkerboard[0]}}};
568 always @(posedge tv_in_line_clock1) begin
569     ntsc_cb_in_addr <= (ntsc_cb_in_addr < 76799) ? (ntsc_cb_in_addr + 1) : 0;
570     cur_x <= (cur_x < 319) ? (cur_x + 1) : 0;
571     if ((cur_x == 319) && (cur_y == 239)) begin
572         cur_y <= 0;
573     end
574     else if (cur_x == 319) begin
575         cur_y <= cur_y + 1;
576     end
577 end
578
579 // instantiate the pixel_map module
580 pixel_map pixel_map(.clk(sys_clk),
581                   .p1_inv(p1_inv),
582                   .p2_inv(p2_inv),
583                   .p3_inv(p3_inv),
584                   .p4_inv(p4_inv),
585                   .p5_inv(p5_inv),
586                   .p6_inv(p6_inv),
587                   .p7_inv(p7_inv),
588                   .p8_inv(p8_inv),
589                   .p9_inv(p9_inv),
590                   .dec_numx_horiz(dec_numx_horiz),
591                   .dec_namy_horiz(dec_namy_horiz),
592                   .dec_denom_horiz(dec_denom_horiz),
593                   .pixel_in(ntsc_dout),
594                   .pixel_out(vga_din),
595                   .ntsc_out_addr(ntsc_out_addr),
596                   .vga_in_wr(vga_in_wr),
597                   .vga_in_addr(vga_in_addr));
598
599 // read from vga buffer for display
600 addr_map addr_map(.hcount(hcount),
601                  .vcount(vcount),
602                  .addr(vga_out_addr));
603
604 /*always @(posedge sys_clk) begin

```

```

605     vga_in_addr <= (vga_in_addr < 76799) ? (vga_in_addr + 1) : 0;
606     ntsc_out_addr <= (ntsc_out_addr < 76799) ? (ntsc_out_addr + 1) : 0;
607     vga_in_wr <= 1;
608 end*/
609
610 // create the brams
611 bram ntsc_buf(.a_clk(tv_in_line_clock1),
612             .a_wr(switch[5] ? ntsc_cb_in_wr : ntsc_we),
613             .a_addr(switch[5] ? ntsc_cb_in_addr : ntsc_addr),
614             .a_din(switch[5] ? ntsc_cb_din : ntsc_data),
615             .b_clk(sys_clk),
616             .b_addr(ntsc_out_addr),
617             .b_dout(ntsc_dout));
618
619 bram vga_buf(.a_clk(sys_clk),
620            .a_wr(vga_in_wr),
621            .a_addr(vga_in_addr),
622            .a_din(vga_din),
623            .b_clk(vga_clk),
624            .b_addr(vga_out_addr),
625            .b_dout(vga_dout));
626
627 ////////////////////////////////////////////////////
628 // Create VGA output signals
629 // In order to meet the setup and hold times of AD7125, we send it ~vga_clk
630 ////////////////////////////////////////////////////
631 assign vga_out_red = {vga_dout[11:8], 4'b0};
632 assign vga_out_green = {vga_dout[7:4], 4'b0};
633 assign vga_out_blue = {vga_dout[3:0], 4'b0};
634 assign vga_out_sync_b = 1'b1; // not used
635 assign vga_out_blank_b = ~blank;
636 assign vga_out_pixel_clock = ~vga_clk;
637 assign vga_out_hsync = hsync;
638 assign vga_out_vsync = vsync;
639
640 ////////////////////////////////////////////////////
641 // instantiate hex display
642 ////////////////////////////////////////////////////
643 wire[63:0] hex_disp_data;
644 // lower 32 bits, keep nice separator of 0 between x, y
645 assign hex_disp_data[8:0] = display_y;
646 assign hex_disp_data[15:9] = 7'd0;
647 assign hex_disp_data[25:16] = display_x;
648 assign hex_disp_data[31:26] = 6'd0;
649 // higher bits, put the percent_kept
650

```

```

651 assign hex_disp_data[63:32] = {10'b0, accel_val[11:6], 10'b0, accel_val[5:0]};
652 display_16hex display_16hex(.reset(reset),
653     .clock_27mhz(clock_27mhz),
654     .data(hex_disp_data),
655     .disp_blank(disp_blank),
656     .disp_clock(disp_clock),
657     .disp_data_out(disp_data_out),
658     .disp_rs(disp_rs),
659     .disp_ce_b(disp_ce_b),
660     .disp_reset_b(disp_reset_b));
661
662 // AC97
663
664 wire [7:0] from_ac97_data, to_ac97_data;
665 wire ready;
666
667 // allow user to adjust volume
668 wire vup,vdown;
669 reg old_vup,old_vdown;
670 debounce bup(.reset(reset),.clock(clock_27mhz),.noisy(~button3),.clean(vup));
671 debounce bdown(.reset(reset),.clock(clock_27mhz),.noisy(~button_down),.clean(vdown));
672 reg [4:0] volume;
673 always @ (posedge clock_27mhz) begin
674     if (reset) volume <= 5'd8;
675     else begin
676         if (vup & ~old_vup & volume != 5'd31) volume <= volume+1;
677         if (vdown & ~old_vdown & volume != 5'd0) volume <= volume-1;
678     end
679     old_vup <= vup;
680     old_vdown <= vdown;
681 end
682
683 // AC97 driver
684 lab5audio labaudio(clock_27mhz, reset, volume, from_ac97_data, to_ac97_data, ready,
685     audio_reset_b, ac97_sdata_out, ac97_sdata_in,
686     ac97_synch, ac97_bit_clock);
687
688 // writeSwitch UP to write, DOWN to read
689 wire writeSwitch;
690 debounce sw7(.reset(reset),.clock(clock_27mhz),.noisy(switch[3]),.clean(writeSwitch));
691
692 wire startSwitch;
693 debounce sw6(.reset(reset),.clock(clock_27mhz),.noisy(switch[2]),.clean(startSwitch));
694
695 wire memReset;
696 debounce benter(.reset(reset),.clock(clock_27mhz),.noisy(~button_enter),.clean(memReset));

```

```

697
698 wire audioTrigger;
699 debounce b3(.reset(reset),.clock(clock_27mhz),.noisy(~button0),.clean(audioTrigger));
700
701 wire [63:0] hexdisp;
702
703 // Receive and Playback module
704 audioManager management(
705     .clock(clock_27mhz),
706     .reset(memReset),
707
708     // User I/O
709     .startSwitch(startSwitch),
710     .audioSelector(percent_kept),
711     .writeSwitch(writeSwitch),
712     // .hexdisp(hexdisp),
713     .audioTrigger(audioTrigger),
714
715     // AC97 I/O
716     .ready(ready),
717     .from_ac97_data(from_ac97_data),
718     .to_ac97_data(to_ac97_data),
719
720     // Flash I/O
721     .flash_data(flash_data),
722     .flash_address(flash_address),
723     .flash_ce_b(flash_ce_b),
724     .flash_oe_b(flash_oe_b),
725     .flash_we_b(flash_we_b),
726     .flash_reset_b(flash_reset_b),
727     .flash_byte_b(flash_byte_b),
728     .flash_sts(flash_sts),
729     .busy(busy),
730
731     // USB I/O
732     .data(user1[31:24]), //the data pins from the USB fifo
733     .rxf(user1[23]), //the rxf pin from the USB fifo
734     .rd(user1[22]) //the rd pin TO the USB FIFO (OUTPUT)
735 );
736
737 endmodule

```

A.2.2 labkit.ucf

```

1 #####
2 #

```



```

3 # 6.111 FPGA Labkit -- Constraints File
4 #
5 # For Labkit Revision 004
6 #
7 #
8 # Created: Oct 30, 2004 from revision 003 constraints file
9 # Author: Nathan Ickes and Isaac Cambron
10 #
11 #####
12 #
13 # CHANGES FOR BOARD REVISION 004
14 #
15 # 1) Added signals for logic analyzer pods 2-4.
16 # 2) Expanded tv_in_ycrcy bus to 20 bits.
17 # 3) Renamed tv_out_sclk to tv_out_i2c_clock for consistency
18 # 4) Renamed tv_out_data to tv_out_i2c_data for consistency
19 # 5) Reversed disp_data_in and disp_data_out signals, so that "out" is an
20 #     output of the FPGA, and "in" is an input.
21 #
22 # CHANGES FOR BOARD REVISION 003
23 #
24 # 1) Combined flash chip enables into a single signal, flash_ce_b.
25 # 2) Moved SRAM feedback clock loop to FPGA pins AL28 (out) and AJ16 (in).
26 # 3) Moved rs232_rts to FPGA pin R3.
27 # 4) Moved flash_address<1> to AE14.
28 #
29 # CHANGES FOR BOARD REVISION 002
30 #
31 # 1) Moved ZBT_BANK1_CLK signal to pin Y9.
32 # 2) Moved user1<30> to J14.
33 # 3) Moved user3<29> to J13.
34 # 4) Added SRAM clock feedback loop between D15 and H15.
35 # 5) Renamed ram#_parity and ram#_we#_b signals.
36 # 6) Removed the constraint on "systemace_clock", since this net no longer
37 #     exists. The SystemACE clock is now hardwired to the 27MHz oscillator
38 #     on the PCB.
39 #
40 #####
41 #
42 # Complete change history (including bug fixes)
43 #
44 # 2007-Aug-16: Fixed revision history. (flash_address<1> was actually changed
45 #             to AE14 for revision 003.)
46 #
47 # 2005-Sep-09: Added missing IOSTANDARD attribute to "disp_data_out".
48 #

```

```

49 # 2005-Jan-23: Added a pullup to FLASH_STS
50 #
51 # 2005-Jan-23: Reduced flash address bus to 24 bits, to match 128Mb devices
52 #         actually populated on the boards. (The boards support up to
53 #         256Mb devices, with 25 address lines.)
54 #
55 # 2005-Jan-23: Change history started.
56 #
57 #####
58
59 #
60 # Audio CODEC
61 #
62
63 NET "beep"          LOC="ac19" | IOSTANDARD=LVDCI_33;
64 NET "audio_reset_b" LOC="ae18" | IOSTANDARD=LVTTTL;
65 NET "ac97_sdata_out" LOC="ac18" | IOSTANDARD=LVDCI_33;
66 NET "ac97_sdata_in"  LOC="aj24";
67 NET "ac97_synch"     LOC="ac17" | IOSTANDARD=LVDCI_33;
68 NET "ac97_bit_clock" LOC="ah24";
69
70 NET "sys_clk" TNM_NET = sys_clk;
71 TIMESPEC TS_sys_clk = PERIOD "sys_clk" 20 ns HIGH 50%;
72
73 NET "vga_clk" TNM_NET = vga_clk;
74 TIMESPEC TS_vga_clk = PERIOD "vga_clk" 40 ns HIGH 50%;
75 #
76 # VGA Output
77 #
78
79 NET "vga_out_red<7>" LOC="ae9" | IOSTANDARD=LVTTTL;
80 NET "vga_out_red<6>" LOC="ae8" | IOSTANDARD=LVTTTL;
81 NET "vga_out_red<5>" LOC="ad12" | IOSTANDARD=LVTTTL;
82 NET "vga_out_red<4>" LOC="af8" | IOSTANDARD=LVTTTL;
83 NET "vga_out_red<3>" LOC="af9" | IOSTANDARD=LVTTTL;
84 NET "vga_out_red<2>" LOC="ag9" | IOSTANDARD=LVTTTL;
85 NET "vga_out_red<1>" LOC="ag10" | IOSTANDARD=LVTTTL;
86 NET "vga_out_red<0>" LOC="af11" | IOSTANDARD=LVTTTL;
87
88 NET "vga_out_green<7>" LOC="ah8" | IOSTANDARD=LVTTTL;
89 NET "vga_out_green<6>" LOC="ah7" | IOSTANDARD=LVTTTL;
90 NET "vga_out_green<5>" LOC="aj6" | IOSTANDARD=LVTTTL;
91 NET "vga_out_green<4>" LOC="ah6" | IOSTANDARD=LVTTTL;
92 NET "vga_out_green<3>" LOC="ad15" | IOSTANDARD=LVTTTL;
93 NET "vga_out_green<2>" LOC="ac14" | IOSTANDARD=LVTTTL;
94 NET "vga_out_green<1>" LOC="ag8" | IOSTANDARD=LVTTTL;

```

```

95 NET "vga_out_green<0>" LOC="ac12" | IOSTANDARD=LVTTL;
96
97 NET "vga_out_blue<7>" LOC="ag15" | IOSTANDARD=LVTTL;
98 NET "vga_out_blue<6>" LOC="ag14" | IOSTANDARD=LVTTL;
99 NET "vga_out_blue<5>" LOC="ag13" | IOSTANDARD=LVTTL;
100 NET "vga_out_blue<4>" LOC="ag12" | IOSTANDARD=LVTTL;
101 NET "vga_out_blue<3>" LOC="aj11" | IOSTANDARD=LVTTL;
102 NET "vga_out_blue<2>" LOC="ah11" | IOSTANDARD=LVTTL;
103 NET "vga_out_blue<1>" LOC="aj10" | IOSTANDARD=LVTTL;
104 NET "vga_out_blue<0>" LOC="ah9" | IOSTANDARD=LVTTL;
105
106 NET "vga_out_sync_b" LOC="aj9" | IOSTANDARD=LVTTL;
107 NET "vga_out_blank_b" LOC="aj8" | IOSTANDARD=LVTTL;
108 NET "vga_out_pixel_clock" LOC="ac10" | IOSTANDARD=LVTTL;
109 NET "vga_out_hsync" LOC="ac13" | IOSTANDARD=LVTTL;
110 NET "vga_out_vsync" LOC="ac11" | IOSTANDARD=LVTTL;
111
112 #
113 # Video Output
114 #
115
116 NET "tv_out_ycrbc<9>" LOC="p27" | IOSTANDARD=LVDCI_33;
117 NET "tv_out_ycrbc<8>" LOC="r27" | IOSTANDARD=LVDCI_33;
118 NET "tv_out_ycrbc<7>" LOC="t29" | IOSTANDARD=LVDCI_33;
119 NET "tv_out_ycrbc<6>" LOC="h26" | IOSTANDARD=LVDCI_33;
120 NET "tv_out_ycrbc<5>" LOC="j26" | IOSTANDARD=LVDCI_33;
121 NET "tv_out_ycrbc<4>" LOC="l26" | IOSTANDARD=LVDCI_33;
122 NET "tv_out_ycrbc<3>" LOC="m26" | IOSTANDARD=LVDCI_33;
123 NET "tv_out_ycrbc<2>" LOC="n26" | IOSTANDARD=LVDCI_33;
124 NET "tv_out_ycrbc<1>" LOC="p26" | IOSTANDARD=LVDCI_33;
125 NET "tv_out_ycrbc<0>" LOC="r26" | IOSTANDARD=LVDCI_33;
126
127 NET "tv_out_reset_b" LOC="g27" | IOSTANDARD=LVDCI_33;
128 NET "tv_out_clock" LOC="l27" | IOSTANDARD=LVDCI_33;
129 NET "tv_out_i2c_clock" LOC="j27" | IOSTANDARD=LVDCI_33;
130 NET "tv_out_i2c_data" LOC="h27" | IOSTANDARD=LVDCI_33;
131 NET "tv_out_pal_ntsc" LOC="j25" | IOSTANDARD=LVDCI_33;
132 NET "tv_out_hsync_b" LOC="n27" | IOSTANDARD=LVDCI_33;
133 NET "tv_out_vsync_b" LOC="m27" | IOSTANDARD=LVDCI_33;
134 NET "tv_out_blank_b" LOC="h25" | IOSTANDARD=LVDCI_33;
135 NET "tv_out_subcar_reset" LOC="k27" | IOSTANDARD=LVDCI_33;
136
137 #
138 # Video Input
139 #
140

```

```

141 NET "tv_in_ycrbc<19>" LOC="ag17";
142 NET "tv_in_ycrbc<18>" LOC="ag18";
143 NET "tv_in_ycrbc<17>" LOC="ag19";
144 NET "tv_in_ycrbc<16>" LOC="ag20";
145 NET "tv_in_ycrbc<15>" LOC="ae20";
146 NET "tv_in_ycrbc<14>" LOC="af21";
147 NET "tv_in_ycrbc<13>" LOC="ad20";
148 NET "tv_in_ycrbc<12>" LOC="ag23";
149 NET "tv_in_ycrbc<11>" LOC="aj26";
150 NET "tv_in_ycrbc<10>" LOC="ah26";
151 NET "tv_in_ycrbc<9>" LOC="w23";
152 NET "tv_in_ycrbc<8>" LOC="v23";
153 NET "tv_in_ycrbc<7>" LOC="u23";
154 NET "tv_in_ycrbc<6>" LOC="t23";
155 NET "tv_in_ycrbc<5>" LOC="t26";
156 NET "tv_in_ycrbc<4>" LOC="t24";
157 NET "tv_in_ycrbc<3>" LOC="r25";
158 NET "tv_in_ycrbc<2>" LOC="l30";
159 NET "tv_in_ycrbc<1>" LOC="m31";
160 NET "tv_in_ycrbc<0>" LOC="m30";
161
162 NET "tv_in_data_valid" LOC="ah25";
163 NET "tv_in_line_clock1" LOC="ad16" | IOSTANDARD=LVDCI_33;
164 NET "tv_in_line_clock2" LOC="ad17" | IOSTANDARD=LVDCI_33;
165 NET "tv_in_aef" LOC="aj23";
166 NET "tv_in_hff" LOC="ah23";
167 NET "tv_in_aff" LOC="aj22";
168 NET "tv_in_i2c_clock" LOC="ad21" | IOSTANDARD=LVDCI_33;
169 NET "tv_in_i2c_data" LOC="ad19" | IOSTANDARD=LVDCI_33;
170 NET "tv_in_fifo_read" LOC="ac22" | IOSTANDARD=LVDCI_33;
171 NET "tv_in_fifo_clock" LOC="ag22" | IOSTANDARD=LVDCI_33;
172 NET "tv_in_iso" LOC="aj27" | IOSTANDARD=LVDCI_33;
173 NET "tv_in_reset_b" LOC="ag25" | IOSTANDARD=LVDCI_33;
174 NET "tv_in_clock" LOC="ab21" | IOSTANDARD=LVDCI_33;
175
176 #
177 # SRAMs
178 #
179
180 NET "ram0_data<35>" LOC="ab25" | IOSTANDARD=LVDCI_33 | NODELAY;
181 NET "ram0_data<34>" LOC="ah29" | IOSTANDARD=LVDCI_33 | NODELAY;
182 NET "ram0_data<33>" LOC="ag28" | IOSTANDARD=LVDCI_33 | NODELAY;
183 NET "ram0_data<32>" LOC="ag29" | IOSTANDARD=LVDCI_33 | NODELAY;
184 NET "ram0_data<31>" LOC="af27" | IOSTANDARD=LVDCI_33 | NODELAY;
185 NET "ram0_data<30>" LOC="af29" | IOSTANDARD=LVDCI_33 | NODELAY;
186 NET "ram0_data<29>" LOC="af28" | IOSTANDARD=LVDCI_33 | NODELAY;

```

```

187 NET "ram0_data<28>" LOC="ae28" | IOSTANDARD=LVDCI_33 | NODELAY;
188 NET "ram0_data<27>" LOC="ad25" | IOSTANDARD=LVDCI_33 | NODELAY;
189 NET "ram0_data<26>" LOC="aa25" | IOSTANDARD=LVDCI_33 | NODELAY;
190 NET "ram0_data<25>" LOC="ah30" | IOSTANDARD=LVDCI_33 | NODELAY;
191 NET "ram0_data<24>" LOC="ah31" | IOSTANDARD=LVDCI_33 | NODELAY;
192 NET "ram0_data<23>" LOC="ag30" | IOSTANDARD=LVDCI_33 | NODELAY;
193 NET "ram0_data<22>" LOC="ag31" | IOSTANDARD=LVDCI_33 | NODELAY;
194 NET "ram0_data<21>" LOC="af30" | IOSTANDARD=LVDCI_33 | NODELAY;
195 NET "ram0_data<20>" LOC="af31" | IOSTANDARD=LVDCI_33 | NODELAY;
196 NET "ram0_data<19>" LOC="ae30" | IOSTANDARD=LVDCI_33 | NODELAY;
197 NET "ram0_data<18>" LOC="ae31" | IOSTANDARD=LVDCI_33 | NODELAY;
198 NET "ram0_data<17>" LOC="y27" | IOSTANDARD=LVDCI_33 | NODELAY;
199 NET "ram0_data<16>" LOC="aa28" | IOSTANDARD=LVDCI_33 | NODELAY;
200 NET "ram0_data<15>" LOC="y29" | IOSTANDARD=LVDCI_33 | NODELAY;
201 NET "ram0_data<14>" LOC="y28" | IOSTANDARD=LVDCI_33 | NODELAY;
202 NET "ram0_data<13>" LOC="w29" | IOSTANDARD=LVDCI_33 | NODELAY;
203 NET "ram0_data<12>" LOC="w28" | IOSTANDARD=LVDCI_33 | NODELAY;
204 NET "ram0_data<11>" LOC="v28" | IOSTANDARD=LVDCI_33 | NODELAY;
205 NET "ram0_data<10>" LOC="u29" | IOSTANDARD=LVDCI_33 | NODELAY;
206 NET "ram0_data<9>" LOC="u28" | IOSTANDARD=LVDCI_33 | NODELAY;
207 NET "ram0_data<8>" LOC="aa27" | IOSTANDARD=LVDCI_33 | NODELAY;
208 NET "ram0_data<7>" LOC="ad31" | IOSTANDARD=LVDCI_33 | NODELAY;
209 NET "ram0_data<6>" LOC="ac30" | IOSTANDARD=LVDCI_33 | NODELAY;
210 NET "ram0_data<5>" LOC="ac31" | IOSTANDARD=LVDCI_33 | NODELAY;
211 NET "ram0_data<4>" LOC="ab30" | IOSTANDARD=LVDCI_33 | NODELAY;
212 NET "ram0_data<3>" LOC="ab31" | IOSTANDARD=LVDCI_33 | NODELAY;
213 NET "ram0_data<2>" LOC="aa30" | IOSTANDARD=LVDCI_33 | NODELAY;
214 NET "ram0_data<1>" LOC="aa31" | IOSTANDARD=LVDCI_33 | NODELAY;
215 NET "ram0_data<0>" LOC="y30" | IOSTANDARD=LVDCI_33 | NODELAY;
216
217 NET "ram0_address<18>" LOC="v31" | IOSTANDARD=LVDCI_33;
218 NET "ram0_address<17>" LOC="w31" | IOSTANDARD=LVDCI_33;
219 NET "ram0_address<16>" LOC="ad28" | IOSTANDARD=LVDCI_33;
220 NET "ram0_address<15>" LOC="ad29" | IOSTANDARD=LVDCI_33;
221 NET "ram0_address<14>" LOC="ac24" | IOSTANDARD=LVDCI_33;
222 NET "ram0_address<13>" LOC="ad26" | IOSTANDARD=LVDCI_33;
223 NET "ram0_address<12>" LOC="ad27" | IOSTANDARD=LVDCI_33;
224 NET "ram0_address<11>" LOC="ac27" | IOSTANDARD=LVDCI_33;
225 NET "ram0_address<10>" LOC="ab27" | IOSTANDARD=LVDCI_33;
226 NET "ram0_address<9>" LOC="y31" | IOSTANDARD=LVDCI_33;
227 NET "ram0_address<8>" LOC="w30" | IOSTANDARD=LVDCI_33;
228 NET "ram0_address<7>" LOC="y26" | IOSTANDARD=LVDCI_33;
229 NET "ram0_address<6>" LOC="y25" | IOSTANDARD=LVDCI_33;
230 NET "ram0_address<5>" LOC="ab24" | IOSTANDARD=LVDCI_33;
231 NET "ram0_address<4>" LOC="ac25" | IOSTANDARD=LVDCI_33;
232 NET "ram0_address<3>" LOC="aa26" | IOSTANDARD=LVDCI_33;

```

```

233 NET "ram0_address<2>" LOC="aa24" | IOSTANDARD=LVDICI_33;
234 NET "ram0_address<1>" LOC="ab29" | IOSTANDARD=LVDICI_33;
235 NET "ram0_address<0>" LOC="ac26" | IOSTANDARD=LVDICI_33;
236
237 NET "ram0_adv_ld" LOC="v26" | IOSTANDARD=LVDICI_33;
238 NET "ram0_clk" LOC="u30" | IOSTANDARD=LVDICI_33;
239 NET "ram0_cen_b" LOC="u25" | IOSTANDARD=LVDICI_33;
240 NET "ram0_ce_b" LOC="w26" | IOSTANDARD=LVDICI_33;
241 NET "ram0_oe_b" LOC="v25" | IOSTANDARD=LVDICI_33;
242 NET "ram0_we_b" LOC="u31" | IOSTANDARD=LVDICI_33;
243 NET "ram0_bwe_b<0>" LOC="v27" | IOSTANDARD=LVDICI_33;
244 NET "ram0_bwe_b<1>" LOC="u27" | IOSTANDARD=LVDICI_33;
245 NET "ram0_bwe_b<2>" LOC="w27" | IOSTANDARD=LVDICI_33;
246 NET "ram0_bwe_b<3>" LOC="u26" | IOSTANDARD=LVDICI_33;
247
248 NET "ram1_data<35>" LOC="aa9" | IOSTANDARD=LVDICI_33 | NODELAY;
249 NET "ram1_data<34>" LOC="ah2" | IOSTANDARD=LVDICI_33 | NODELAY;
250 NET "ram1_data<33>" LOC="ah1" | IOSTANDARD=LVDICI_33 | NODELAY;
251 NET "ram1_data<32>" LOC="ag2" | IOSTANDARD=LVDICI_33 | NODELAY;
252 NET "ram1_data<31>" LOC="ag1" | IOSTANDARD=LVDICI_33 | NODELAY;
253 NET "ram1_data<30>" LOC="af2" | IOSTANDARD=LVDICI_33 | NODELAY;
254 NET "ram1_data<29>" LOC="af1" | IOSTANDARD=LVDICI_33 | NODELAY;
255 NET "ram1_data<28>" LOC="ae2" | IOSTANDARD=LVDICI_33 | NODELAY;
256 NET "ram1_data<27>" LOC="ae1" | IOSTANDARD=LVDICI_33 | NODELAY;
257 NET "ram1_data<26>" LOC="ab9" | IOSTANDARD=LVDICI_33 | NODELAY;
258 NET "ram1_data<25>" LOC="ah3" | IOSTANDARD=LVDICI_33 | NODELAY;
259 NET "ram1_data<24>" LOC="ag4" | IOSTANDARD=LVDICI_33 | NODELAY;
260 NET "ram1_data<23>" LOC="ag3" | IOSTANDARD=LVDICI_33 | NODELAY;
261 NET "ram1_data<22>" LOC="af4" | IOSTANDARD=LVDICI_33 | NODELAY;
262 NET "ram1_data<21>" LOC="af3" | IOSTANDARD=LVDICI_33 | NODELAY;
263 NET "ram1_data<20>" LOC="ae4" | IOSTANDARD=LVDICI_33 | NODELAY;
264 NET "ram1_data<19>" LOC="ae5" | IOSTANDARD=LVDICI_33 | NODELAY;
265 NET "ram1_data<18>" LOC="ad5" | IOSTANDARD=LVDICI_33 | NODELAY;
266 NET "ram1_data<17>" LOC="v2" | IOSTANDARD=LVDICI_33 | NODELAY;
267 NET "ram1_data<16>" LOC="ad1" | IOSTANDARD=LVDICI_33 | NODELAY;
268 NET "ram1_data<15>" LOC="ac2" | IOSTANDARD=LVDICI_33 | NODELAY;
269 NET "ram1_data<14>" LOC="ac1" | IOSTANDARD=LVDICI_33 | NODELAY;
270 NET "ram1_data<13>" LOC="ab2" | IOSTANDARD=LVDICI_33 | NODELAY;
271 NET "ram1_data<12>" LOC="ab1" | IOSTANDARD=LVDICI_33 | NODELAY;
272 NET "ram1_data<11>" LOC="aa2" | IOSTANDARD=LVDICI_33 | NODELAY;
273 NET "ram1_data<10>" LOC="aa1" | IOSTANDARD=LVDICI_33 | NODELAY;
274 NET "ram1_data<9>" LOC="y2" | IOSTANDARD=LVDICI_33 | NODELAY;
275 NET "ram1_data<8>" LOC="v4" | IOSTANDARD=LVDICI_33 | NODELAY;
276 NET "ram1_data<7>" LOC="ac3" | IOSTANDARD=LVDICI_33 | NODELAY;
277 NET "ram1_data<6>" LOC="ac4" | IOSTANDARD=LVDICI_33 | NODELAY;
278 NET "ram1_data<5>" LOC="aa5" | IOSTANDARD=LVDICI_33 | NODELAY;

```

```

279 NET "ram1_data<4>" LOC="aa3" | IOSTANDARD=LVDCI_33 | NODELAY;
280 NET "ram1_data<3>" LOC="aa4" | IOSTANDARD=LVDCI_33 | NODELAY;
281 NET "ram1_data<2>" LOC="y3" | IOSTANDARD=LVDCI_33 | NODELAY;
282 NET "ram1_data<1>" LOC="y4" | IOSTANDARD=LVDCI_33 | NODELAY;
283 NET "ram1_data<0>" LOC="w3" | IOSTANDARD=LVDCI_33 | NODELAY;
284
285 NET "ram1_address<18>" LOC="ab3" | IOSTANDARD=LVDCI_33;
286 NET "ram1_address<17>" LOC="ac5" | IOSTANDARD=LVDCI_33;
287 NET "ram1_address<16>" LOC="u6" | IOSTANDARD=LVDCI_33;
288 NET "ram1_address<15>" LOC="v6" | IOSTANDARD=LVDCI_33;
289 NET "ram1_address<14>" LOC="w6" | IOSTANDARD=LVDCI_33;
290 NET "ram1_address<13>" LOC="y6" | IOSTANDARD=LVDCI_33;
291 NET "ram1_address<12>" LOC="aa7" | IOSTANDARD=LVDCI_33;
292 NET "ram1_address<11>" LOC="ab7" | IOSTANDARD=LVDCI_33;
293 NET "ram1_address<10>" LOC="ac6" | IOSTANDARD=LVDCI_33;
294 NET "ram1_address<9>" LOC="ad3" | IOSTANDARD=LVDCI_33;
295 NET "ram1_address<8>" LOC="ad4" | IOSTANDARD=LVDCI_33;
296 NET "ram1_address<7>" LOC="u3" | IOSTANDARD=LVDCI_33;
297 NET "ram1_address<6>" LOC="w4" | IOSTANDARD=LVDCI_33;
298 NET "ram1_address<5>" LOC="ac8" | IOSTANDARD=LVDCI_33;
299 NET "ram1_address<4>" LOC="ab8" | IOSTANDARD=LVDCI_33;
300 NET "ram1_address<3>" LOC="aa8" | IOSTANDARD=LVDCI_33;
301 NET "ram1_address<2>" LOC="y7" | IOSTANDARD=LVDCI_33;
302 NET "ram1_address<1>" LOC="y8" | IOSTANDARD=LVDCI_33;
303 NET "ram1_address<0>" LOC="ad7" | IOSTANDARD=LVDCI_33;
304
305 NET "ram1_adv_ld" LOC="y5" | IOSTANDARD=LVDCI_33;
306 NET "ram1_clk" LOC="y9" | IOSTANDARD=LVDCI_33;
307 NET "ram1_cen_b" LOC="v5" | IOSTANDARD=LVDCI_33;
308 NET "ram1_ce_b" LOC="u4" | IOSTANDARD=LVDCI_33;
309 NET "ram1_oe_b" LOC="w5" | IOSTANDARD=LVDCI_33;
310 NET "ram1_we_b" LOC="aa6" | IOSTANDARD=LVDCI_33;
311 NET "ram1_bwe_b<0>" LOC="u2" | IOSTANDARD=LVDCI_33;
312 NET "ram1_bwe_b<1>" LOC="u1" | IOSTANDARD=LVDCI_33;
313 NET "ram1_bwe_b<2>" LOC="v1" | IOSTANDARD=LVDCI_33;
314 NET "ram1_bwe_b<3>" LOC="u5" | IOSTANDARD=LVDCI_33;
315
316 NET "clock_feedback_out" LOC="al28" | IOSTANDARD=LVDCI_33;
317 NET "clock_feedback_in" LOC="aj16";
318
319 #
320 # Flash
321 #
322
323 NET "flash_data<15>" LOC="ak10" | IOSTANDARD=LVTTL;
324 NET "flash_data<14>" LOC="ak11" | IOSTANDARD=LVTTL;

```

```

325 NET "flash_data<13>" LOC="ak12" | IOSTANDARD=LVTTL;
326 NET "flash_data<12>" LOC="ak13" | IOSTANDARD=LVTTL;
327 NET "flash_data<11>" LOC="ak14" | IOSTANDARD=LVTTL;
328 NET "flash_data<10>" LOC="ak15" | IOSTANDARD=LVTTL;
329 NET "flash_data<9>" LOC="ah12" | IOSTANDARD=LVTTL;
330 NET "flash_data<8>" LOC="ah13" | IOSTANDARD=LVTTL;
331 NET "flash_data<7>" LOC="a110" | IOSTANDARD=LVTTL;
332 NET "flash_data<6>" LOC="a111" | IOSTANDARD=LVTTL;
333 NET "flash_data<5>" LOC="a112" | IOSTANDARD=LVTTL;
334 NET "flash_data<4>" LOC="a113" | IOSTANDARD=LVTTL;
335 NET "flash_data<3>" LOC="a114" | IOSTANDARD=LVTTL;
336 NET "flash_data<2>" LOC="a115" | IOSTANDARD=LVTTL;
337 NET "flash_data<1>" LOC="aj12" | IOSTANDARD=LVTTL;
338 NET "flash_data<0>" LOC="aj13" | IOSTANDARD=LVTTL;
339
340 NET "flash_address<24>" LOC="a17";
341 NET "flash_address<23>" LOC="aj15";
342 NET "flash_address<22>" LOC="a125";
343 NET "flash_address<21>" LOC="ak23";
344 NET "flash_address<20>" LOC="a123";
345 NET "flash_address<19>" LOC="ak22";
346 NET "flash_address<18>" LOC="a122";
347 NET "flash_address<17>" LOC="ak21";
348 NET "flash_address<16>" LOC="a121";
349 NET "flash_address<15>" LOC="ak20";
350 NET "flash_address<14>" LOC="a120";
351 NET "flash_address<13>" LOC="ak19";
352 NET "flash_address<12>" LOC="a119";
353 NET "flash_address<11>" LOC="a118";
354 NET "flash_address<10>" LOC="ak17";
355 NET "flash_address<9>" LOC="a117";
356 NET "flash_address<8>" LOC="ah21";
357 NET "flash_address<7>" LOC="aj20";
358 NET "flash_address<6>" LOC="ah20";
359 NET "flash_address<5>" LOC="aj19";
360 NET "flash_address<4>" LOC="ah19";
361 NET "flash_address<3>" LOC="ah18";
362 NET "flash_address<2>" LOC="aj17";
363 NET "flash_address<1>" LOC="ae14";
364 NET "flash_address<0>" LOC="ah14";
365
366 NET "flash_ce_b" LOC="aj21" | IOSTANDARD=LVDCI_33;
367
368 NET "flash_oe_b" LOC="ak9" | IOSTANDARD=LVDCI_33;
369 NET "flash_we_b" LOC="a18" | IOSTANDARD=LVDCI_33;
370 NET "flash_reset_b" LOC="ak18" | IOSTANDARD=LVDCI_33;

```



```

371 NET "flash_sts" LOC="a19" | PULLUP;
372 NET "flash_byte_b" LOC="ah15" | IOSTANDARD=LVDCI_33;
373
374 #
375 # RS-232
376 #
377
378 NET "rs232_txd" LOC="p4" | IOSTANDARD=LVDCI_33;
379 NET "rs232_rxd" LOC="p6";
380 NET "rs232_rts" LOC="r3" | IOSTANDARD=LVDCI_33;
381 NET "rs232_cts" LOC="n8";
382
383 #
384 # Mouse and Keyboard
385 #
386
387 NET "mouse_clock" LOC="ac16";
388 NET "mouse_data" LOC="ac15";
389 NET "keyboard_clock" LOC="ag16";
390 NET "keyboard_data" LOC="af16";
391
392 #
393 # Clocks
394 #
395
396 NET "clock_27mhz" LOC="c16";
397 NET "clock1" LOC="h16";
398 NET "clock2" LOC="c15";
399
400 #
401 # Alphanumeric Display
402 #
403
404 NET "disp_blank" LOC="af12" | IOSTANDARD=LVDCI_33;
405 NET "disp_data_in" LOC="ae12";
406 NET "disp_clock" LOC="af14" | IOSTANDARD=LVDCI_33;
407 NET "disp_rs" LOC="af15" | IOSTANDARD=LVDCI_33;
408 NET "disp_ce_b" LOC="af13" | IOSTANDARD=LVDCI_33;
409 NET "disp_reset_b" LOC="ag11" | IOSTANDARD=LVDCI_33;
410 NET "disp_data_out" LOC="ae15" | IOSTANDARD=LVDCI_33;
411
412 #
413 # Buttons and Switches
414 #
415
416 NET "button0" LOC="ae11";

```

```

417 NET "button1" LOC="ae10";
418 NET "button2" LOC="ad11";
419 NET "button3" LOC="ab12";
420 NET "button_enter" LOC="ak7";
421 NET "button_right" LOC="al6";
422 NET "button_left" LOC="al5";
423 NET "button_up" LOC="al4";
424 NET "button_down" LOC="ak6";
425
426 NET "switch<7>" LOC="ad22";
427 NET "switch<6>" LOC="ae23";
428 NET "switch<5>" LOC="ac20";
429 NET "switch<4>" LOC="ab20";
430 NET "switch<3>" LOC="ac21";
431 NET "switch<2>" LOC="ak25";
432 NET "switch<1>" LOC="al26";
433 NET "switch<0>" LOC="ak26";
434
435 #
436 # Discrete LEDs
437 #
438
439 NET "led<7>" LOC="ae17" | IOSTANDARD=LVTTL;
440 NET "led<6>" LOC="af17" | IOSTANDARD=LVTTL;
441 NET "led<5>" LOC="af18" | IOSTANDARD=LVTTL;
442 NET "led<4>" LOC="af19" | IOSTANDARD=LVTTL;
443 NET "led<3>" LOC="af20" | IOSTANDARD=LVTTL;
444 NET "led<2>" LOC="ag21" | IOSTANDARD=LVTTL;
445 NET "led<1>" LOC="ae21" | IOSTANDARD=LVTTL;
446 NET "led<0>" LOC="ae22" | IOSTANDARD=LVTTL;
447
448
449 #
450 # User Pins
451 #
452
453 NET "user1<31>" LOC="j15" | IOSTANDARD=LVTTL;
454 NET "user1<30>" LOC="j14" | IOSTANDARD=LVTTL;
455 NET "user1<29>" LOC="g15" | IOSTANDARD=LVTTL;
456 NET "user1<28>" LOC="f14" | IOSTANDARD=LVTTL;
457 NET "user1<27>" LOC="f12" | IOSTANDARD=LVTTL;
458 NET "user1<26>" LOC="h11" | IOSTANDARD=LVTTL;
459 NET "user1<25>" LOC="g9" | IOSTANDARD=LVTTL;
460 NET "user1<24>" LOC="h9" | IOSTANDARD=LVTTL;
461 NET "user1<23>" LOC="b15" | IOSTANDARD=LVTTL;
462 NET "user1<22>" LOC="b14" | IOSTANDARD=LVTTL;

```

```

463 NET "user1<21>" LOC="f15" | IOSTANDARD=LVTTL;
464 NET "user1<20>" LOC="e13" | IOSTANDARD=LVTTL;
465 NET "user1<19>" LOC="e11" | IOSTANDARD=LVTTL;
466 NET "user1<18>" LOC="e9" | IOSTANDARD=LVTTL;
467 NET "user1<17>" LOC="f8" | IOSTANDARD=LVTTL;
468 NET "user1<16>" LOC="f7" | IOSTANDARD=LVTTL;
469 NET "user1<15>" LOC="c13" | IOSTANDARD=LVTTL;
470 NET "user1<14>" LOC="c12" | IOSTANDARD=LVTTL;
471 NET "user1<13>" LOC="c11" | IOSTANDARD=LVTTL;
472 NET "user1<12>" LOC="c10" | IOSTANDARD=LVTTL;
473 NET "user1<11>" LOC="c9" | IOSTANDARD=LVTTL;
474 NET "user1<10>" LOC="c8" | IOSTANDARD=LVTTL;
475 NET "user1<9>" LOC="c6" | IOSTANDARD=LVTTL;
476 NET "user1<8>" LOC="e6" | IOSTANDARD=LVTTL;
477 NET "user1<7>" LOC="a11" | IOSTANDARD=LVTTL;
478 NET "user1<6>" LOC="a10" | IOSTANDARD=LVTTL;
479 NET "user1<5>" LOC="a9" | IOSTANDARD=LVTTL;
480 NET "user1<4>" LOC="a8" | IOSTANDARD=LVTTL;
481 NET "user1<3>" LOC="b6" | IOSTANDARD=LVTTL;
482 NET "user1<2>" LOC="b5" | IOSTANDARD=LVTTL;
483 NET "user1<1>" LOC="c5" | IOSTANDARD=LVTTL;
484 NET "user1<0>" LOC="b3" | IOSTANDARD=LVTTL;
485
486
487 NET "user2<31>" LOC="b27" | IOSTANDARD=LVTTL;
488 NET "user2<30>" LOC="b26" | IOSTANDARD=LVTTL;
489 NET "user2<29>" LOC="b25" | IOSTANDARD=LVTTL;
490 NET "user2<28>" LOC="a24" | IOSTANDARD=LVTTL;
491 NET "user2<27>" LOC="a23" | IOSTANDARD=LVTTL;
492 NET "user2<26>" LOC="a22" | IOSTANDARD=LVTTL;
493 NET "user2<25>" LOC="a21" | IOSTANDARD=LVTTL;
494 NET "user2<24>" LOC="a20" | IOSTANDARD=LVTTL;
495 NET "user2<23>" LOC="d26" | IOSTANDARD=LVTTL;
496 NET "user2<22>" LOC="d25" | IOSTANDARD=LVTTL;
497 NET "user2<21>" LOC="c24" | IOSTANDARD=LVTTL;
498 NET "user2<20>" LOC="d23" | IOSTANDARD=LVTTL;
499 NET "user2<19>" LOC="d21" | IOSTANDARD=LVTTL;
500 NET "user2<18>" LOC="d20" | IOSTANDARD=LVTTL;
501 NET "user2<17>" LOC="d19" | IOSTANDARD=LVTTL;
502 NET "user2<16>" LOC="d18" | IOSTANDARD=LVTTL;
503 NET "user2<15>" LOC="f24" | IOSTANDARD=LVTTL;
504 NET "user2<14>" LOC="f23" | IOSTANDARD=LVTTL;
505 NET "user2<13>" LOC="e22" | IOSTANDARD=LVTTL;
506 NET "user2<12>" LOC="e20" | IOSTANDARD=LVTTL;
507 NET "user2<11>" LOC="e18" | IOSTANDARD=LVTTL;
508 NET "user2<10>" LOC="e16" | IOSTANDARD=LVTTL;

```

```

509 NET "user2<9>" LOC="a19" | IOSTANDARD=LVTTL;
510 NET "user2<8>" LOC="a18" | IOSTANDARD=LVTTL;
511 NET "user2<7>" LOC="h22" | IOSTANDARD=LVTTL;
512 NET "user2<6>" LOC="g22" | IOSTANDARD=LVTTL;
513 NET "user2<5>" LOC="f21" | IOSTANDARD=LVTTL;
514 NET "user2<4>" LOC="f19" | IOSTANDARD=LVTTL;
515 NET "user2<3>" LOC="f17" | IOSTANDARD=LVTTL;
516 NET "user2<2>" LOC="h19" | IOSTANDARD=LVTTL;
517 NET "user2<1>" LOC="g20" | IOSTANDARD=LVTTL;
518 NET "user2<0>" LOC="h21" | IOSTANDARD=LVTTL;
519
520 NET "user3<31>" LOC="g12" | IOSTANDARD=LVTTL;
521 NET "user3<30>" LOC="h13" | IOSTANDARD=LVTTL;
522 NET "user3<29>" LOC="j13" | IOSTANDARD=LVTTL;
523 NET "user3<28>" LOC="g14" | IOSTANDARD=LVTTL;
524 NET "user3<27>" LOC="f13" | IOSTANDARD=LVTTL;
525 NET "user3<26>" LOC="f11" | IOSTANDARD=LVTTL;
526 NET "user3<25>" LOC="g10" | IOSTANDARD=LVTTL;
527 NET "user3<24>" LOC="h10" | IOSTANDARD=LVTTL;
528 NET "user3<23>" LOC="a15" | IOSTANDARD=LVTTL;
529 NET "user3<22>" LOC="a14" | IOSTANDARD=LVTTL;
530 NET "user3<21>" LOC="e15" | IOSTANDARD=LVTTL;
531 NET "user3<20>" LOC="e14" | IOSTANDARD=LVTTL;
532 NET "user3<19>" LOC="e12" | IOSTANDARD=LVTTL;
533 NET "user3<18>" LOC="e10" | IOSTANDARD=LVTTL;
534 NET "user3<17>" LOC="f9" | IOSTANDARD=LVTTL;
535 NET "user3<16>" LOC="g8" | IOSTANDARD=LVTTL;
536 NET "user3<15>" LOC="d14" | IOSTANDARD=LVTTL;
537 NET "user3<14>" LOC="d13" | IOSTANDARD=LVTTL;
538 NET "user3<13>" LOC="d12" | IOSTANDARD=LVTTL;
539 NET "user3<12>" LOC="d11" | IOSTANDARD=LVTTL;
540 NET "user3<11>" LOC="d9" | IOSTANDARD=LVTTL;
541 NET "user3<10>" LOC="d8" | IOSTANDARD=LVTTL;
542 NET "user3<9>" LOC="d7" | IOSTANDARD=LVTTL;
543 NET "user3<8>" LOC="d6" | IOSTANDARD=LVTTL;
544 NET "user3<7>" LOC="b12" | IOSTANDARD=LVTTL;
545 NET "user3<6>" LOC="b11" | IOSTANDARD=LVTTL;
546 NET "user3<5>" LOC="b10" | IOSTANDARD=LVTTL;
547 NET "user3<4>" LOC="b9" | IOSTANDARD=LVTTL;
548 NET "user3<3>" LOC="a7" | IOSTANDARD=LVTTL;
549 NET "user3<2>" LOC="a6" | IOSTANDARD=LVTTL;
550 NET "user3<1>" LOC="a5" | IOSTANDARD=LVTTL;
551 NET "user3<0>" LOC="a4" | IOSTANDARD=LVTTL;
552
553 NET "user4<31>" LOC="a28" | IOSTANDARD=LVTTL;
554 NET "user4<30>" LOC="a27" | IOSTANDARD=LVTTL;

```

```

555 NET "user4<29>" LOC="a26" | IOSTANDARD=LVTTL;
556 NET "user4<28>" LOC="a25" | IOSTANDARD=LVTTL;
557 NET "user4<27>" LOC="b23" | IOSTANDARD=LVTTL;
558 NET "user4<26>" LOC="b22" | IOSTANDARD=LVTTL;
559 NET "user4<25>" LOC="b21" | IOSTANDARD=LVTTL;
560 NET "user4<24>" LOC="b20" | IOSTANDARD=LVTTL;
561 NET "user4<23>" LOC="e25" | IOSTANDARD=LVTTL;
562 NET "user4<22>" LOC="c26" | IOSTANDARD=LVTTL;
563 NET "user4<21>" LOC="d24" | IOSTANDARD=LVTTL;
564 NET "user4<20>" LOC="c23" | IOSTANDARD=LVTTL;
565 NET "user4<19>" LOC="c22" | IOSTANDARD=LVTTL;
566 NET "user4<18>" LOC="c21" | IOSTANDARD=LVTTL;
567 NET "user4<17>" LOC="c20" | IOSTANDARD=LVTTL;
568 NET "user4<16>" LOC="c19" | IOSTANDARD=LVTTL;
569 NET "user4<15>" LOC="g24" | IOSTANDARD=LVTTL;
570 NET "user4<14>" LOC="e24" | IOSTANDARD=LVTTL;
571 NET "user4<13>" LOC="e23" | IOSTANDARD=LVTTL;
572 NET "user4<12>" LOC="e21" | IOSTANDARD=LVTTL;
573 NET "user4<11>" LOC="e19" | IOSTANDARD=LVTTL;
574 NET "user4<10>" LOC="e17" | IOSTANDARD=LVTTL;
575 NET "user4<9>" LOC="b19" | IOSTANDARD=LVTTL;
576 NET "user4<8>" LOC="b18" | IOSTANDARD=LVTTL;
577 NET "user4<7>" LOC="h23" | IOSTANDARD=LVTTL;
578 NET "user4<6>" LOC="g23" | IOSTANDARD=LVTTL;
579 NET "user4<5>" LOC="g21" | IOSTANDARD=LVTTL;
580 NET "user4<4>" LOC="f20" | IOSTANDARD=LVTTL;
581 NET "user4<3>" LOC="f18" | IOSTANDARD=LVTTL;
582 NET "user4<2>" LOC="f16" | IOSTANDARD=LVTTL;
583 NET "user4<1>" LOC="g18" | IOSTANDARD=LVTTL;
584 NET "user4<0>" LOC="g17" | IOSTANDARD=LVTTL;
585
586 #
587 # Daughter Card
588 #
589
590 NET "daughtercard<43>" LOC="L7" | IOSTANDARD=LVTTL;
591 NET "daughtercard<42>" LOC="H1" | IOSTANDARD=LVTTL;
592 NET "daughtercard<41>" LOC="J2" | IOSTANDARD=LVTTL;
593 NET "daughtercard<40>" LOC="J1" | IOSTANDARD=LVTTL;
594 NET "daughtercard<39>" LOC="K2" | IOSTANDARD=LVTTL;
595 NET "daughtercard<38>" LOC="M7" | IOSTANDARD=LVTTL;
596 NET "daughtercard<37>" LOC="M6" | IOSTANDARD=LVTTL;
597 NET "daughtercard<36>" LOC="M3" | IOSTANDARD=LVTTL;
598 NET "daughtercard<35>" LOC="M4" | IOSTANDARD=LVTTL;
599 NET "daughtercard<34>" LOC="L3" | IOSTANDARD=LVTTL;
600 NET "daughtercard<33>" LOC="K1" | IOSTANDARD=LVTTL;

```

```

601 NET "daughtercard<32>" LOC="L4" | IOSTANDARD=LVTTL;
602 NET "daughtercard<31>" LOC="K3" | IOSTANDARD=LVTTL;
603 NET "daughtercard<30>" LOC="K9" | IOSTANDARD=LVTTL;
604 NET "daughtercard<29>" LOC="L9" | IOSTANDARD=LVTTL;
605 NET "daughtercard<28>" LOC="K8" | IOSTANDARD=LVTTL;
606 NET "daughtercard<27>" LOC="K7" | IOSTANDARD=LVTTL;
607 NET "daughtercard<26>" LOC="L8" | IOSTANDARD=LVTTL;
608 NET "daughtercard<25>" LOC="L6" | IOSTANDARD=LVTTL;
609 NET "daughtercard<24>" LOC="M5" | IOSTANDARD=LVTTL;
610 NET "daughtercard<23>" LOC="N5" | IOSTANDARD=LVTTL;
611 NET "daughtercard<22>" LOC="P5" | IOSTANDARD=LVTTL;
612 NET "daughtercard<21>" LOC="D3" | IOSTANDARD=LVTTL;
613 NET "daughtercard<20>" LOC="E4" | IOSTANDARD=LVTTL;
614 NET "daughtercard<19>" LOC="E3" | IOSTANDARD=LVTTL;
615 NET "daughtercard<18>" LOC="F4" | IOSTANDARD=LVTTL;
616 NET "daughtercard<17>" LOC="F3" | IOSTANDARD=LVTTL;
617 NET "daughtercard<16>" LOC="G4" | IOSTANDARD=LVTTL;
618 NET "daughtercard<15>" LOC="H4" | IOSTANDARD=LVTTL;
619 NET "daughtercard<14>" LOC="J3" | IOSTANDARD=LVTTL;
620 NET "daughtercard<13>" LOC="J4" | IOSTANDARD=LVTTL;
621 NET "daughtercard<12>" LOC="D2" | IOSTANDARD=LVTTL;
622 NET "daughtercard<11>" LOC="D1" | IOSTANDARD=LVTTL;
623 NET "daughtercard<10>" LOC="E2" | IOSTANDARD=LVTTL;
624 NET "daughtercard<9>" LOC="E1" | IOSTANDARD=LVTTL;
625 NET "daughtercard<8>" LOC="F5" | IOSTANDARD=LVTTL;
626 NET "daughtercard<7>" LOC="G5" | IOSTANDARD=LVTTL;
627 NET "daughtercard<6>" LOC="H5" | IOSTANDARD=LVTTL;
628 NET "daughtercard<5>" LOC="J5" | IOSTANDARD=LVTTL;
629 NET "daughtercard<4>" LOC="K5" | IOSTANDARD=LVTTL;
630 NET "daughtercard<3>" LOC="H7" | IOSTANDARD=LVTTL;
631 NET "daughtercard<2>" LOC="J8" | IOSTANDARD=LVTTL;
632 NET "daughtercard<1>" LOC="J6" | IOSTANDARD=LVTTL;
633 NET "daughtercard<0>" LOC="J7" | IOSTANDARD=LVTTL;
634
635 #
636 # System Ace
637 #
638
639 NET "systemace_data<15>" LOC="F29" | IOSTANDARD=LVTTL;
640 NET "systemace_data<14>" LOC="G28" | IOSTANDARD=LVTTL;
641 NET "systemace_data<13>" LOC="H29" | IOSTANDARD=LVTTL;
642 NET "systemace_data<12>" LOC="H28" | IOSTANDARD=LVTTL;
643 NET "systemace_data<11>" LOC="J29" | IOSTANDARD=LVTTL;
644 NET "systemace_data<10>" LOC="J28" | IOSTANDARD=LVTTL;
645 NET "systemace_data<9>" LOC="K29" | IOSTANDARD=LVTTL;
646 NET "systemace_data<8>" LOC="L29" | IOSTANDARD=LVTTL;

```

```

647 NET "systemace_data<7>" LOC="L28" | IOSTANDARD=LVTTL;
648 NET "systemace_data<6>" LOC="M29" | IOSTANDARD=LVTTL;
649 NET "systemace_data<5>" LOC="M28" | IOSTANDARD=LVTTL;
650 NET "systemace_data<4>" LOC="N29" | IOSTANDARD=LVTTL;
651 NET "systemace_data<3>" LOC="N28" | IOSTANDARD=LVTTL;
652 NET "systemace_data<2>" LOC="P28" | IOSTANDARD=LVTTL;
653 NET "systemace_data<1>" LOC="R29" | IOSTANDARD=LVTTL;
654 NET "systemace_data<0>" LOC="R28" | IOSTANDARD=LVTTL;
655
656 NET "systemace_address<6>" LOC="E29" | IOSTANDARD=LVTTL;
657 NET "systemace_address<5>" LOC="F28" | IOSTANDARD=LVTTL;
658 NET "systemace_address<4>" LOC="H31" | IOSTANDARD=LVTTL;
659 NET "systemace_address<3>" LOC="J30" | IOSTANDARD=LVTTL;
660 NET "systemace_address<2>" LOC="J31" | IOSTANDARD=LVTTL;
661 NET "systemace_address<1>" LOC="K30" | IOSTANDARD=LVTTL;
662 NET "systemace_address<0>" LOC="K31" | IOSTANDARD=LVTTL;
663
664 NET "systemace_ce_b" LOC="E28" | IOSTANDARD=LVTTL;
665 NET "systemace_we_b" LOC="P31" | IOSTANDARD=LVTTL;
666 NET "systemace_oe_b" LOC="R31" | IOSTANDARD=LVTTL;
667 NET "systemace_irq" LOC="D29";
668 NET "systemace_mpbrdy" LOC="L31";
669
670 #
671 # Logic Analyzer
672 #
673
674 NET "analyzer1_data<15>" LOC="G1" | IOSTANDARD=LVTTL;
675 NET "analyzer1_data<14>" LOC="H3" | IOSTANDARD=LVTTL;
676 NET "analyzer1_data<13>" LOC="M9" | IOSTANDARD=LVTTL;
677 NET "analyzer1_data<12>" LOC="M8" | IOSTANDARD=LVTTL;
678 NET "analyzer1_data<11>" LOC="L5" | IOSTANDARD=LVTTL;
679 NET "analyzer1_data<10>" LOC="L1" | IOSTANDARD=LVTTL;
680 NET "analyzer1_data<9>" LOC="L2" | IOSTANDARD=LVTTL;
681 NET "analyzer1_data<8>" LOC="N9" | IOSTANDARD=LVTTL;
682 NET "analyzer1_data<7>" LOC="M1" | IOSTANDARD=LVTTL;
683 NET "analyzer1_data<6>" LOC="M2" | IOSTANDARD=LVTTL;
684 NET "analyzer1_data<5>" LOC="N1" | IOSTANDARD=LVTTL;
685 NET "analyzer1_data<4>" LOC="N2" | IOSTANDARD=LVTTL;
686 NET "analyzer1_data<3>" LOC="P1" | IOSTANDARD=LVTTL;
687 NET "analyzer1_data<2>" LOC="P2" | IOSTANDARD=LVTTL;
688 NET "analyzer1_data<1>" LOC="R1" | IOSTANDARD=LVTTL;
689 NET "analyzer1_data<0>" LOC="R2" | IOSTANDARD=LVTTL;
690 NET "analyzer1_clock" LOC="G2" | IOSTANDARD=LVTTL;
691
692 NET "analyzer2_data<15>" LOC="f2" | IOSTANDARD=LVTTL;

```

```

693 NET "analyzer2_data<14>" LOC="k10" | IOSTANDARD=LVTTL;
694 NET "analyzer2_data<13>" LOC="l10" | IOSTANDARD=LVTTL;
695 NET "analyzer2_data<12>" LOC="m10" | IOSTANDARD=LVTTL;
696 NET "analyzer2_data<11>" LOC="r7" | IOSTANDARD=LVTTL;
697 NET "analyzer2_data<10>" LOC="n3" | IOSTANDARD=LVTTL;
698 NET "analyzer2_data<9>" LOC="r8" | IOSTANDARD=LVTTL;
699 NET "analyzer2_data<8>" LOC="r9" | IOSTANDARD=LVTTL;
700 NET "analyzer2_data<7>" LOC="p9" | IOSTANDARD=LVTTL;
701 NET "analyzer2_data<6>" LOC="n6" | IOSTANDARD=LVTTL;
702 NET "analyzer2_data<5>" LOC="p7" | IOSTANDARD=LVTTL;
703 NET "analyzer2_data<4>" LOC="n4" | IOSTANDARD=LVTTL;
704 NET "analyzer2_data<3>" LOC="t8" | IOSTANDARD=LVTTL;
705 NET "analyzer2_data<2>" LOC="t9" | IOSTANDARD=LVTTL;
706 NET "analyzer2_data<1>" LOC="r6" | IOSTANDARD=LVTTL;
707 NET "analyzer2_data<0>" LOC="r5" | IOSTANDARD=LVTTL;
708 NET "analyzer2_clock" LOC="f1" | IOSTANDARD=LVTTL;
709
710 NET "analyzer3_data<15>" LOC="k24" | IOSTANDARD=LVTTL;
711 NET "analyzer3_data<14>" LOC="k25" | IOSTANDARD=LVTTL;
712 NET "analyzer3_data<13>" LOC="k22" | IOSTANDARD=LVTTL;
713 NET "analyzer3_data<12>" LOC="l24" | IOSTANDARD=LVTTL;
714 NET "analyzer3_data<11>" LOC="l25" | IOSTANDARD=LVTTL;
715 NET "analyzer3_data<10>" LOC="l22" | IOSTANDARD=LVTTL;
716 NET "analyzer3_data<9>" LOC="l23" | IOSTANDARD=LVTTL;
717 NET "analyzer3_data<8>" LOC="m23" | IOSTANDARD=LVTTL;
718 NET "analyzer3_data<7>" LOC="m24" | IOSTANDARD=LVTTL;
719 NET "analyzer3_data<6>" LOC="m25" | IOSTANDARD=LVTTL;
720 NET "analyzer3_data<5>" LOC="k23" | IOSTANDARD=LVTTL;
721 NET "analyzer3_data<4>" LOC="m22" | IOSTANDARD=LVTTL;
722 NET "analyzer3_data<3>" LOC="n23" | IOSTANDARD=LVTTL;
723 NET "analyzer3_data<2>" LOC="p23" | IOSTANDARD=LVTTL;
724 NET "analyzer3_data<1>" LOC="r23" | IOSTANDARD=LVTTL;
725 NET "analyzer3_data<0>" LOC="r24" | IOSTANDARD=LVTTL;
726 NET "analyzer3_clock" LOC="j24" | IOSTANDARD=LVTTL;
727
728 NET "analyzer4_data<15>" LOC="ag7" | IOSTANDARD=LVTTL;
729 NET "analyzer4_data<14>" LOC="ak3" | IOSTANDARD=LVTTL;
730 NET "analyzer4_data<13>" LOC="aj5" | IOSTANDARD=LVTTL;
731 NET "analyzer4_data<12>" LOC="ak29" | IOSTANDARD=LVTTL;
732 NET "analyzer4_data<11>" LOC="ak28" | IOSTANDARD=LVTTL;
733 NET "analyzer4_data<10>" LOC="af25" | IOSTANDARD=LVTTL;
734 NET "analyzer4_data<9>" LOC="ag24" | IOSTANDARD=LVTTL;
735 NET "analyzer4_data<8>" LOC="af24" | IOSTANDARD=LVTTL;
736 NET "analyzer4_data<7>" LOC="af23" | IOSTANDARD=LVTTL;
737 NET "analyzer4_data<6>" LOC="al27" | IOSTANDARD=LVTTL;
738 NET "analyzer4_data<5>" LOC="ak27" | IOSTANDARD=LVTTL;

```



```

739 NET "analyzer4_data<4>" LOC="ah17" | IOSTANDARD=LVTTL;
740 NET "analyzer4_data<3>" LOC="ad13" | IOSTANDARD=LVTTL;
741 NET "analyzer4_data<2>" LOC="v7" | IOSTANDARD=LVTTL;
742 NET "analyzer4_data<1>" LOC="u7" | IOSTANDARD=LVTTL;
743 NET "analyzer4_data<0>" LOC="u8" | IOSTANDARD=LVTTL;
744 NET "analyzer4_clock" LOC="ad9" | IOSTANDARD=LVTTL;

```

A.3 Our Modules

A.3.1 acc.v

```

1  'timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
3  /*
4  Copyright (C) {2014} {James Thomas} <jamesjoethomas@gmail.com>
5
6  This program is free software: you can redistribute it and/or modify
7  it under the terms of the GNU General Public License as published by
8  the Free Software Foundation, either version 3 of the License, or
9  (at your option) any later version.
10
11  This program is distributed in the hope that it will be useful,
12  but WITHOUT ANY WARRANTY; without even the implied warranty of
13  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14  GNU General Public License for more details.
15
16  You should have received a copy of the GNU General Public License
17  along with this program. If not, see <http://www.gnu.org/licenses/>.
18  */
19  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
20  /*
21  output starts on same cycle start is asserted
22  high-order bits produced first
23  input only needs to be supplied on the cycle that start is high
24  produces zeros if input exhausted
25  assume W > 1
26  */
27  module par_to_ser #(parameter W=8)
28      (input clk /* device clock */, input[W-1:0] par,
29       input start, output ser);
30      reg[W-1:0] par_reg = 0;
31
32      always @(posedge clk) begin
33          if (start) begin
34              par_reg <= {par[W-2:0], 1'b0};
35          end

```

```

36         else begin
37             par_reg <= {par_reg[W-2:0], 1'b0};
38         end
39     end
40
41     assign ser = start ? par[W-1] : par_reg[W-1];
42 endmodule
43
44 /*
45 output appears W-1 clock cycles after first serial bit sent
46 assume high-order bits are input first
47 assume W > 2
48 */
49 module ser_to_par #(parameter W=8)
50     (input clk /* device clock */, input ser,
51     output[W-1:0] par);
52     reg[W-2:0] par_reg = 0;
53
54     always @(posedge clk) begin
55         par_reg <= {par_reg[W-3:0], ser};
56     end
57
58     assign par = {par_reg, ser};
59 endmodule
60
61 /*
62 reduces the system clock by a factor of 6
63 */
64 module acc_clk(input clk /* system clock */, output dev_clk);
65     parameter TICKS = 9;
66
67     reg [3:0] count = 0;
68     reg sig_reg = 0;
69
70     always @(posedge clk) begin
71         if (count == TICKS) begin
72             // flip at half period
73             sig_reg <= ~sig_reg;
74             count <= 0;
75         end
76         else begin
77             count <= count + 1;
78         end
79     end
80     assign dev_clk = sig_reg;
81 endmodule

```

```

82
83  /*
84  assert in_ready when a new datapoint is available, avg_ready will
85  be signalled after 32 data points have been folded into the average
86  */
87  module moving_avg(
88      input clock, in_ready, reset,
89      input signed [15:0] data,
90      output signed [15:0] avg,
91      output avg_ready
92  );
93      // circular buffer
94      reg signed [15:0] samples [31:0];
95
96      reg [4:0] offset = 0;
97      reg signed [15:0] accum = 0;
98      reg [5:0] num_samples = 0;
99      reg signed [15:0] data_right_shift;
100
101      always @(*) begin
102          data_right_shift = {data[15], data[15], data[15], data[15],
103                          data[15], data[15:5]};
104      end
105
106      always @(posedge clock) begin
107          if (reset) begin
108              accum <= 0;
109              num_samples <= 0;
110              offset <= 0;
111          end
112          else if (in_ready) begin
113              num_samples <= (num_samples == 6'd32) ? num_samples : num_samples +
114              samples[offset] <= data_right_shift;
115              if (num_samples == 6'd32) begin
116                  accum <= accum + data_right_shift - samples[offset];
117              end
118              else begin
119                  accum <= accum + data_right_shift;
120              end
121              offset <= offset + 1;
122          end
123      end
124
125      assign avg = accum;
126      assign avg_ready = (num_samples == 6'd32) ? 1 : 0;
127  endmodule

```

```

128
129  /*
130  ready permanently asserted after initialization completed
131  acc operates completely with the slowed accelerometer clock
132  */
133  module acc(input clk /* system clock */, sdo, reset,
134            output ncs, sda, scl, ready, output signed [15:0] x, y);
135            // TODO use state machine -- transition through all of the initialization states
136            // register), then rotate through the value reading states
137            // one cycle gap between states to allow for CS deassertion
138            parameter MEASURE_INIT = 0;
139            parameter X_READ = 1;
140            parameter Y_READ = 2;
141            reg[1:0] state = MEASURE_INIT; // TODO: set the right number of bits for this
142            reg[4:0] count = 0; // TODO: set the right number of bits for this
143
144            reg ncs_reg;
145
146            wire dev_clk;
147            acc_clk ac(.clk(clk), .dev_clk(dev_clk));
148
149            reg[7:0] par_in;
150            reg pts_start;
151            par_to_ser pts(.clk(dev_clk), .par(par_in), .start(pts_start), .ser(sda));
152
153            wire[7:0] par_out;
154            ser_to_par stp(.clk(dev_clk), .ser(sdo), .par(par_out));
155
156            reg ma_x_in_ready;
157            reg [7:0] x_low_bits = 0;
158            reg signed [15:0] ma_x_in;
159            wire ma_x_avg_ready;
160            wire signed [15:0] ma_x_avg;
161            moving_avg ma_x(
162                .clock(dev_clk), .in_ready(ma_x_in_ready), .reset(reset),
163                .data(ma_x_in),
164                .avg(ma_x_avg),
165                .avg_ready(ma_x_avg_ready)
166            );
167
168            reg ma_y_in_ready;
169            reg [7:0] y_low_bits = 0;
170            reg signed [15:0] ma_y_in;
171            wire ma_y_avg_ready;
172            wire signed [15:0] ma_y_avg;
173            moving_avg ma_y(

```

```

174         .clock(dev_clk), .in_ready(ma_y_in_ready), .reset(reset),
175         .data(ma_y_in),
176         .avg(ma_y_avg),
177         .avg_ready(ma_y_avg_ready)
178     );
179
180     // invariants: when transitioning out of a state always set counter to 0
181     always @(posedge dev_clk) begin
182         case (state)
183             MEASURE_INIT: begin
184                 if (count == 5'd18) begin
185                     count <= 0;
186                     state <= X_READ;
187                 end
188                 else begin
189                     count <= count + 1;
190                 end
191             end
192             X_READ: begin
193                 if (count == 5'd25) begin
194                     count <= 0;
195                     state <= Y_READ;
196                 end
197                 else begin
198                     count <= count + 1;
199                 end
200                 if (count == 5'd17) begin
201                     x_low_bits <= par_out;
202                 end
203             end
204             Y_READ: begin
205                 if (count == 5'd25) begin
206                     count <= 0;
207                     state <= X_READ;
208                 end
209                 else begin
210                     count <= count + 1;
211                 end
212                 if (count == 5'd17) begin
213                     y_low_bits <= par_out;
214                 end
215             end
216         endcase
217     end
218
219     always @(*) begin

```

```

220         case (state)
221             MEASURE_INIT: begin
222                 pts_start = (count == 5'd2 || count == 5'd10) ? 1 : 0;
223                 if (count == 5'd2) begin
224                     par_in = 8'h2D; // 0 for W, 0 for MB
225                 end
226                 else if (count == 5'd10) begin
227                     par_in = 8'h08; // set measure bit
228                 end
229                 else begin
230                     par_in = 0;
231                 end
232                 ma_x_in_ready = 0; ma_x_in = 0;
233                 ma_y_in_ready = 0; ma_y_in = 0;
234                 ncs_reg = (count == 5'd18 || count == 5'd0) ? 1 : 0;
235             end
236             X_READ: begin
237                 pts_start = (count == 5'd1) ? 1 : 0;
238                 par_in = (count == 5'd1) ? 8'hF2 : 0; // 1 for R, 1 for MB
239                 ma_x_in_ready = (count == 5'd25) ? 1 : 0;
240                 ma_x_in = (count == 5'd25) ? {par_out, x_low_bits} : 0;
241                 ma_y_in_ready = 0; ma_y_in = 0;
242                 ncs_reg = (count == 5'd25) ? 1 : 0;
243             end
244             Y_READ: begin
245                 pts_start = (count == 5'd1) ? 1 : 0;
246                 par_in = (count == 5'd1) ? 8'hF4 : 0; // 1 for R, 1 for MB
247                 ma_y_in_ready = (count == 5'd25) ? 1 : 0;
248                 ma_y_in = (count == 5'd25) ? {par_out, y_low_bits} : 0;
249                 ma_x_in_ready = 0; ma_x_in = 0;
250                 ncs_reg = (count == 5'd25) ? 1 : 0;
251             end
252         endcase
253     end
254
255
256     assign scl = (ncs_reg == 1 || (state == MEASURE_INIT && count == 5'd1
257         || state != MEASURE_INIT && count == 5'd0)) ? 1 : dev_clk;
258     assign ncs = ncs_reg;
259     assign ready = ma_x_avg_ready && ma_y_avg_ready;
260     assign x = ma_x_avg;
261     assign y = ma_y_avg;
262 endmodule

```


45 12'd33: quad_corners = 76'd165313902172117936646;
46 12'd34: quad_corners = 76'd165169223208381195272;
47 12'd35: quad_corners = 76'd238811520539482660361;
48 12'd36: quad_corners = 76'd238666841575745918475;
49 12'd37: quad_corners = 76'd312165023443625184781;
50 12'd38: quad_corners = 76'd312020344479620007951;
51 12'd39: quad_corners = 76'd385662500798086774801;
52 12'd40: quad_corners = 76'd459304797854041896980;
53 12'd41: quad_corners = 76'd532802979996530634774;
54 12'd42: quad_corners = 76'd606445136314729490457;
55 12'd43: quad_corners = 76'd680087433370416177180;
56 12'd44: quad_corners = 76'd753729730151224956958;
57 12'd45: quad_corners = 76'd827227771281079520801;
58 12'd46: quad_corners = 76'd900870068061619865125;
59 12'd47: quad_corners = 76'd1048299200674119531560;
60 12'd48: quad_corners = 76'd1121941356717172044332;
61 12'd49: quad_corners = 76'd1195439397571879741487;
62 12'd50: quad_corners = 76'd1342868670646990380595;
63 12'd51: quad_corners = 76'd1490297802984343704631;
64 12'd52: quad_corners = 76'd1563795843838783490619;
65 12'd53: quad_corners = 76'd1563795843838783490619;
66 12'd54: quad_corners = 76'd1563795843838783490619;
67 12'd55: quad_corners = 76'd1563795843838783490619;
68 12'd56: quad_corners = 76'd1563795843838783490619;
69 12'd57: quad_corners = 76'd1563795843838783490619;
70 12'd58: quad_corners = 76'd1563795843838783490619;
71 12'd59: quad_corners = 76'd1563795843838783490619;
72 12'd60: quad_corners = 76'd1563795843838783490619;
73 12'd61: quad_corners = 76'd1563795843838783490619;
74 12'd62: quad_corners = 76'd1563795843838783490619;
75 12'd63: quad_corners = 76'd1563795843838783490619;
76 12'd64: quad_corners = 76'd166903815503556664320;
77 12'd65: quad_corners = 76'd166903815503556664320;
78 12'd66: quad_corners = 76'd166903815503556664320;
79 12'd67: quad_corners = 76'd166903815503556664320;
80 12'd68: quad_corners = 76'd166903815503556664320;
81 12'd69: quad_corners = 76'd166903815503556664320;
82 12'd70: quad_corners = 76'd166903815503556664320;
83 12'd71: quad_corners = 76'd166903815503556664320;
84 12'd72: quad_corners = 76'd166903815503556664320;
85 12'd73: quad_corners = 76'd166903815503556664320;
86 12'd74: quad_corners = 76'd166903815503556664320;
87 12'd75: quad_corners = 76'd166903815503556664320;
88 12'd76: quad_corners = 76'd166903815503556664320;
89 12'd77: quad_corners = 76'd166903815503556664320;
90 12'd78: quad_corners = 76'd166903815503556664320;

91 12'd79: quad_corners = 76'd166903815503556664320;
92 12'd80: quad_corners = 76'd166903815503556664320;
93 12'd81: quad_corners = 76'd166903815503556664320;
94 12'd82: quad_corners = 76'd166903815503556664320;
95 12'd83: quad_corners = 76'd166903815503556664320;
96 12'd84: quad_corners = 76'd166903815503556664320;
97 12'd85: quad_corners = 76'd166903815503556664320;
98 12'd86: quad_corners = 76'd166903815503556664320;
99 12'd87: quad_corners = 76'd166903815503556664320;
100 12'd88: quad_corners = 76'd166903815503556664320;
101 12'd89: quad_corners = 76'd166759137365526861312;
102 12'd90: quad_corners = 76'd92683508482071876096;
103 12'd91: quad_corners = 76'd92538830344042597376;
104 12'd92: quad_corners = 76'd92394292943501150208;
105 12'd93: quad_corners = 76'd92249614805471871489;
106 12'd94: quad_corners = 76'd92104936667442592770;
107 12'd95: quad_corners = 76'd91960398992023762435;
108 12'd96: quad_corners = 76'd165458581135586242565;
109 12'd97: quad_corners = 76'd165313902172117936646;
110 12'd98: quad_corners = 76'd165169223208381195272;
111 12'd99: quad_corners = 76'd238811520539482660361;
112 12'd100: quad_corners = 76'd238666841575745918475;
113 12'd101: quad_corners = 76'd312165023443625184781;
114 12'd102: quad_corners = 76'd312020344479620007951;
115 12'd103: quad_corners = 76'd385662500798086774801;
116 12'd104: quad_corners = 76'd459304797854041896980;
117 12'd105: quad_corners = 76'd532802979996530634774;
118 12'd106: quad_corners = 76'd606445136314729490457;
119 12'd107: quad_corners = 76'd680087433370416177180;
120 12'd108: quad_corners = 76'd753729730151224956958;
121 12'd109: quad_corners = 76'd827227771281079520801;
122 12'd110: quad_corners = 76'd900870068061619865125;
123 12'd111: quad_corners = 76'd1048299200674119531560;
124 12'd112: quad_corners = 76'd1121941356717172044332;
125 12'd113: quad_corners = 76'd1195439397571879741487;
126 12'd114: quad_corners = 76'd1342868670646990380595;
127 12'd115: quad_corners = 76'd1490297802984343704631;
128 12'd116: quad_corners = 76'd1563795843838783490619;
129 12'd117: quad_corners = 76'd1563795843838783490619;
130 12'd118: quad_corners = 76'd1563795843838783490619;
131 12'd119: quad_corners = 76'd1563795843838783490619;
132 12'd120: quad_corners = 76'd1563795843838783490619;
133 12'd121: quad_corners = 76'd1563795843838783490619;
134 12'd122: quad_corners = 76'd1563795843838783490619;
135 12'd123: quad_corners = 76'd1563795843838783490619;
136 12'd124: quad_corners = 76'd1563795843838783490619;

137 12'd125: quad_corners = 76'd1563795843838783490619;
138 12'd126: quad_corners = 76'd1563795843838783490619;
139 12'd127: quad_corners = 76'd1563795843838783490619;
140 12'd128: quad_corners = 76'd166903815503556664320;
141 12'd129: quad_corners = 76'd166903815503556664320;
142 12'd130: quad_corners = 76'd166903815503556664320;
143 12'd131: quad_corners = 76'd166903815503556664320;
144 12'd132: quad_corners = 76'd166903815503556664320;
145 12'd133: quad_corners = 76'd166903815503556664320;
146 12'd134: quad_corners = 76'd166903815503556664320;
147 12'd135: quad_corners = 76'd166903815503556664320;
148 12'd136: quad_corners = 76'd166903815503556664320;
149 12'd137: quad_corners = 76'd166903815503556664320;
150 12'd138: quad_corners = 76'd166903815503556664320;
151 12'd139: quad_corners = 76'd166903815503556664320;
152 12'd140: quad_corners = 76'd166903815503556664320;
153 12'd141: quad_corners = 76'd166903815503556664320;
154 12'd142: quad_corners = 76'd166903815503556664320;
155 12'd143: quad_corners = 76'd166903815503556664320;
156 12'd144: quad_corners = 76'd166903815503556664320;
157 12'd145: quad_corners = 76'd166903815503556664320;
158 12'd146: quad_corners = 76'd166903815503556664320;
159 12'd147: quad_corners = 76'd166903815503556664320;
160 12'd148: quad_corners = 76'd166903815503556664320;
161 12'd149: quad_corners = 76'd166903815503556664320;
162 12'd150: quad_corners = 76'd166903815503556664320;
163 12'd151: quad_corners = 76'd166903815503556664320;
164 12'd152: quad_corners = 76'd166903815503556664320;
165 12'd153: quad_corners = 76'd166759137365526861312;
166 12'd154: quad_corners = 76'd92683508482071876096;
167 12'd155: quad_corners = 76'd92538830344042597376;
168 12'd156: quad_corners = 76'd92394292943501150208;
169 12'd157: quad_corners = 76'd92249614805471871489;
170 12'd158: quad_corners = 76'd92104936667442592770;
171 12'd159: quad_corners = 76'd91960398992023762435;
172 12'd160: quad_corners = 76'd165458581135586242565;
173 12'd161: quad_corners = 76'd165313902172117936646;
174 12'd162: quad_corners = 76'd165169223208381195272;
175 12'd163: quad_corners = 76'd238811520539482660361;
176 12'd164: quad_corners = 76'd238666841575745918475;
177 12'd165: quad_corners = 76'd312165023443625184781;
178 12'd166: quad_corners = 76'd312020344479620007951;
179 12'd167: quad_corners = 76'd385662500798086774801;
180 12'd168: quad_corners = 76'd459304797854041896980;
181 12'd169: quad_corners = 76'd532802979996530634774;
182 12'd170: quad_corners = 76'd606445136314729490457;

183 12'd171: quad_corners = 76'd680087433370416177180;
184 12'd172: quad_corners = 76'd753729730151224956958;
185 12'd173: quad_corners = 76'd827227771281079520801;
186 12'd174: quad_corners = 76'd900870068061619865125;
187 12'd175: quad_corners = 76'd1048299200674119531560;
188 12'd176: quad_corners = 76'd1121941356717172044332;
189 12'd177: quad_corners = 76'd1195439397571879741487;
190 12'd178: quad_corners = 76'd1342868670646990380595;
191 12'd179: quad_corners = 76'd1490297802984343704631;
192 12'd180: quad_corners = 76'd1563795843838783490619;
193 12'd181: quad_corners = 76'd1563795843838783490619;
194 12'd182: quad_corners = 76'd1563795843838783490619;
195 12'd183: quad_corners = 76'd1563795843838783490619;
196 12'd184: quad_corners = 76'd1563795843838783490619;
197 12'd185: quad_corners = 76'd1563795843838783490619;
198 12'd186: quad_corners = 76'd1563795843838783490619;
199 12'd187: quad_corners = 76'd1563795843838783490619;
200 12'd188: quad_corners = 76'd1563795843838783490619;
201 12'd189: quad_corners = 76'd1563795843838783490619;
202 12'd190: quad_corners = 76'd1563795843838783490619;
203 12'd191: quad_corners = 76'd1563795843838783490619;
204 12'd192: quad_corners = 76'd166903815503556664320;
205 12'd193: quad_corners = 76'd166903815503556664320;
206 12'd194: quad_corners = 76'd166903815503556664320;
207 12'd195: quad_corners = 76'd166903815503556664320;
208 12'd196: quad_corners = 76'd166903815503556664320;
209 12'd197: quad_corners = 76'd166903815503556664320;
210 12'd198: quad_corners = 76'd166903815503556664320;
211 12'd199: quad_corners = 76'd166903815503556664320;
212 12'd200: quad_corners = 76'd166903815503556664320;
213 12'd201: quad_corners = 76'd166903815503556664320;
214 12'd202: quad_corners = 76'd166903815503556664320;
215 12'd203: quad_corners = 76'd166903815503556664320;
216 12'd204: quad_corners = 76'd166903815503556664320;
217 12'd205: quad_corners = 76'd166903815503556664320;
218 12'd206: quad_corners = 76'd166903815503556664320;
219 12'd207: quad_corners = 76'd166903815503556664320;
220 12'd208: quad_corners = 76'd166903815503556664320;
221 12'd209: quad_corners = 76'd166903815503556664320;
222 12'd210: quad_corners = 76'd166903815503556664320;
223 12'd211: quad_corners = 76'd166903815503556664320;
224 12'd212: quad_corners = 76'd166903815503556664320;
225 12'd213: quad_corners = 76'd166903815503556664320;
226 12'd214: quad_corners = 76'd166903815503556664320;
227 12'd215: quad_corners = 76'd166903815503556664320;
228 12'd216: quad_corners = 76'd166903815503556664320;

229 12'd217: quad_corners = 76'd166759137365526861312;
230 12'd218: quad_corners = 76'd92683508482071876096;
231 12'd219: quad_corners = 76'd92538830344042597376;
232 12'd220: quad_corners = 76'd92394292943501150208;
233 12'd221: quad_corners = 76'd92249614805471871489;
234 12'd222: quad_corners = 76'd92104936667442592770;
235 12'd223: quad_corners = 76'd91960398992023762435;
236 12'd224: quad_corners = 76'd165458581135586242565;
237 12'd225: quad_corners = 76'd165313902172117936646;
238 12'd226: quad_corners = 76'd165169223208381195272;
239 12'd227: quad_corners = 76'd238811520539482660361;
240 12'd228: quad_corners = 76'd238666841575745918475;
241 12'd229: quad_corners = 76'd312165023443625184781;
242 12'd230: quad_corners = 76'd312020344479620007951;
243 12'd231: quad_corners = 76'd385662500798086774801;
244 12'd232: quad_corners = 76'd459304797854041896980;
245 12'd233: quad_corners = 76'd532802979996530634774;
246 12'd234: quad_corners = 76'd606445136314729490457;
247 12'd235: quad_corners = 76'd680087433370416177180;
248 12'd236: quad_corners = 76'd753729730151224956958;
249 12'd237: quad_corners = 76'd827227771281079520801;
250 12'd238: quad_corners = 76'd900870068061619865125;
251 12'd239: quad_corners = 76'd1048299200674119531560;
252 12'd240: quad_corners = 76'd1121941356717172044332;
253 12'd241: quad_corners = 76'd1195439397571879741487;
254 12'd242: quad_corners = 76'd1342868670646990380595;
255 12'd243: quad_corners = 76'd1490297802984343704631;
256 12'd244: quad_corners = 76'd1563795843838783490619;
257 12'd245: quad_corners = 76'd1563795843838783490619;
258 12'd246: quad_corners = 76'd1563795843838783490619;
259 12'd247: quad_corners = 76'd1563795843838783490619;
260 12'd248: quad_corners = 76'd1563795843838783490619;
261 12'd249: quad_corners = 76'd1563795843838783490619;
262 12'd250: quad_corners = 76'd1563795843838783490619;
263 12'd251: quad_corners = 76'd1563795843838783490619;
264 12'd252: quad_corners = 76'd1563795843838783490619;
265 12'd253: quad_corners = 76'd1563795843838783490619;
266 12'd254: quad_corners = 76'd1563795843838783490619;
267 12'd255: quad_corners = 76'd1563795843838783490619;
268 12'd256: quad_corners = 76'd166903815503556664320;
269 12'd257: quad_corners = 76'd166903815503556664320;
270 12'd258: quad_corners = 76'd166903815503556664320;
271 12'd259: quad_corners = 76'd166903815503556664320;
272 12'd260: quad_corners = 76'd166903815503556664320;
273 12'd261: quad_corners = 76'd166903815503556664320;
274 12'd262: quad_corners = 76'd166903815503556664320;

```

275      12'd263: quad_corners = 76'd166903815503556664320;
276      12'd264: quad_corners = 76'd166903815503556664320;
277      12'd265: quad_corners = 76'd166903815503556664320;
278      12'd266: quad_corners = 76'd166903815503556664320;
279      12'd267: quad_corners = 76'd166903815503556664320;
280      12'd268: quad_corners = 76'd166903815503556664320;
281      12'd269: quad_corners = 76'd166903815503556664320;
282      12'd270: quad_corners = 76'd166903815503556664320;
283      12'd271: quad_corners = 76'd166903815503556664320;
284      12'd272: quad_corners = 76'd166903815503556664320;
285      12'd273: quad_corners = 76'd166903815503556664320;
286      12'd274: quad_corners = 76'd166903815503556664320;
287      12'd275: quad_corners = 76'd166903815503556664320;
288      12'd276: quad_corners = 76'd166903815503556664320;
289      12'd277: quad_corners = 76'd166903815503556664320;
290      12'd278: quad_corners = 76'd166903815503556664320;
291      12'd279: quad_corners = 76'd166903815503556664320;
292      12'd280: quad_corners = 76'd166903815503556664320;
293      12'd281: quad_corners = 76'd166759137365526861312;
294      12'd282: quad_corners = 76'd92683508482071876096;
295      12'd283: quad_corners = 76'd92538830344042597376;
296      12'd284: quad_corners = 76'd92394292943501150208;
297      12'd285: quad_corners = 76'd92249614805471871489;
298      12'd286: quad_corners = 76'd92104936667442592770;
299      12'd287: quad_corners = 76'd91960398992023762435;
300      12'd288: quad_corners = 76'd165458581135586242565;
301      12'd289: quad_corners = 76'd165313902172117936646;
302      12'd290: quad_corners = 76'd165169223208381195272;
303      12'd291: quad_corners = 76'd238811520539482660361;
304      12'd292: quad_corners = 76'd238666841575745918475;
305      12'd293: quad_corners = 76'd312165023443625184781;
306      12'd294: quad_corners = 76'd312020344479620007951;
307      12'd295: quad_corners = 76'd385662500798086774801;
308      12'd296: quad_corners = 76'd459304797854041896980;
309      12'd297: quad_corners = 76'd532802979996530634774;
310      12'd298: quad_corners = 76'd606445136314729490457;
311      12'd299: quad_corners = 76'd680087433370416177180;
312      12'd300: quad_corners = 76'd753729730151224956958;
313      12'd301: quad_corners = 76'd827227771281079520801;
314      12'd302: quad_corners = 76'd900870068061619865125;
315      12'd303: quad_corners = 76'd1048299200674119531560;
316      12'd304: quad_corners = 76'd1121941356717172044332;
317      12'd305: quad_corners = 76'd1195439397571879741487;
318      12'd306: quad_corners = 76'd1342868670646990380595;
319      12'd307: quad_corners = 76'd1490297802984343704631;
320      12'd308: quad_corners = 76'd1563795843838783490619;

```

321 12'd309: quad_corners = 76'd1563795843838783490619;
322 12'd310: quad_corners = 76'd1563795843838783490619;
323 12'd311: quad_corners = 76'd1563795843838783490619;
324 12'd312: quad_corners = 76'd1563795843838783490619;
325 12'd313: quad_corners = 76'd1563795843838783490619;
326 12'd314: quad_corners = 76'd1563795843838783490619;
327 12'd315: quad_corners = 76'd1563795843838783490619;
328 12'd316: quad_corners = 76'd1563795843838783490619;
329 12'd317: quad_corners = 76'd1563795843838783490619;
330 12'd318: quad_corners = 76'd1563795843838783490619;
331 12'd319: quad_corners = 76'd1563795843838783490619;
332 12'd320: quad_corners = 76'd166903815503556664320;
333 12'd321: quad_corners = 76'd166903815503556664320;
334 12'd322: quad_corners = 76'd166903815503556664320;
335 12'd323: quad_corners = 76'd166903815503556664320;
336 12'd324: quad_corners = 76'd166903815503556664320;
337 12'd325: quad_corners = 76'd166903815503556664320;
338 12'd326: quad_corners = 76'd166903815503556664320;
339 12'd327: quad_corners = 76'd166903815503556664320;
340 12'd328: quad_corners = 76'd166903815503556664320;
341 12'd329: quad_corners = 76'd166903815503556664320;
342 12'd330: quad_corners = 76'd166903815503556664320;
343 12'd331: quad_corners = 76'd166903815503556664320;
344 12'd332: quad_corners = 76'd166903815503556664320;
345 12'd333: quad_corners = 76'd166903815503556664320;
346 12'd334: quad_corners = 76'd166903815503556664320;
347 12'd335: quad_corners = 76'd166903815503556664320;
348 12'd336: quad_corners = 76'd166903815503556664320;
349 12'd337: quad_corners = 76'd166903815503556664320;
350 12'd338: quad_corners = 76'd166903815503556664320;
351 12'd339: quad_corners = 76'd166903815503556664320;
352 12'd340: quad_corners = 76'd166903815503556664320;
353 12'd341: quad_corners = 76'd166903815503556664320;
354 12'd342: quad_corners = 76'd166903815503556664320;
355 12'd343: quad_corners = 76'd166903815503556664320;
356 12'd344: quad_corners = 76'd166903815503556664320;
357 12'd345: quad_corners = 76'd166759137365526861312;
358 12'd346: quad_corners = 76'd92683508482071876096;
359 12'd347: quad_corners = 76'd92538830344042597376;
360 12'd348: quad_corners = 76'd92394292943501150208;
361 12'd349: quad_corners = 76'd92249614805471871489;
362 12'd350: quad_corners = 76'd92104936667442592770;
363 12'd351: quad_corners = 76'd91960398992023762435;
364 12'd352: quad_corners = 76'd165458581135586242565;
365 12'd353: quad_corners = 76'd165313902172117936646;
366 12'd354: quad_corners = 76'd165169223208381195272;

367 12'd355: quad_corners = 76'd238811520539482660361;
368 12'd356: quad_corners = 76'd238666841575745918475;
369 12'd357: quad_corners = 76'd312165023443625184781;
370 12'd358: quad_corners = 76'd312020344479620007951;
371 12'd359: quad_corners = 76'd385662500798086774801;
372 12'd360: quad_corners = 76'd459304797854041896980;
373 12'd361: quad_corners = 76'd532802979996530634774;
374 12'd362: quad_corners = 76'd606445136314729490457;
375 12'd363: quad_corners = 76'd680087433370416177180;
376 12'd364: quad_corners = 76'd753729730151224956958;
377 12'd365: quad_corners = 76'd827227771281079520801;
378 12'd366: quad_corners = 76'd900870068061619865125;
379 12'd367: quad_corners = 76'd1048299200674119531560;
380 12'd368: quad_corners = 76'd1121941356717172044332;
381 12'd369: quad_corners = 76'd1195439397571879741487;
382 12'd370: quad_corners = 76'd1342868670646990380595;
383 12'd371: quad_corners = 76'd1490297802984343704631;
384 12'd372: quad_corners = 76'd1563795843838783490619;
385 12'd373: quad_corners = 76'd1563795843838783490619;
386 12'd374: quad_corners = 76'd1563795843838783490619;
387 12'd375: quad_corners = 76'd1563795843838783490619;
388 12'd376: quad_corners = 76'd1563795843838783490619;
389 12'd377: quad_corners = 76'd1563795843838783490619;
390 12'd378: quad_corners = 76'd1563795843838783490619;
391 12'd379: quad_corners = 76'd1563795843838783490619;
392 12'd380: quad_corners = 76'd1563795843838783490619;
393 12'd381: quad_corners = 76'd1563795843838783490619;
394 12'd382: quad_corners = 76'd1563795843838783490619;
395 12'd383: quad_corners = 76'd1563795843838783490619;
396 12'd384: quad_corners = 76'd166903815503556664320;
397 12'd385: quad_corners = 76'd166903815503556664320;
398 12'd386: quad_corners = 76'd166903815503556664320;
399 12'd387: quad_corners = 76'd166903815503556664320;
400 12'd388: quad_corners = 76'd166903815503556664320;
401 12'd389: quad_corners = 76'd166903815503556664320;
402 12'd390: quad_corners = 76'd166903815503556664320;
403 12'd391: quad_corners = 76'd166903815503556664320;
404 12'd392: quad_corners = 76'd166903815503556664320;
405 12'd393: quad_corners = 76'd166903815503556664320;
406 12'd394: quad_corners = 76'd166903815503556664320;
407 12'd395: quad_corners = 76'd166903815503556664320;
408 12'd396: quad_corners = 76'd166903815503556664320;
409 12'd397: quad_corners = 76'd166903815503556664320;
410 12'd398: quad_corners = 76'd166903815503556664320;
411 12'd399: quad_corners = 76'd166903815503556664320;
412 12'd400: quad_corners = 76'd166903815503556664320;

413 12'd401: quad_corners = 76'd166903815503556664320;
414 12'd402: quad_corners = 76'd166903815503556664320;
415 12'd403: quad_corners = 76'd166903815503556664320;
416 12'd404: quad_corners = 76'd166903815503556664320;
417 12'd405: quad_corners = 76'd166903815503556664320;
418 12'd406: quad_corners = 76'd166903815503556664320;
419 12'd407: quad_corners = 76'd166903815503556664320;
420 12'd408: quad_corners = 76'd166903815503556664320;
421 12'd409: quad_corners = 76'd166759137365526861312;
422 12'd410: quad_corners = 76'd92683508482071876096;
423 12'd411: quad_corners = 76'd92538830344042597376;
424 12'd412: quad_corners = 76'd92394292943501150208;
425 12'd413: quad_corners = 76'd92249614805471871489;
426 12'd414: quad_corners = 76'd92104936667442592770;
427 12'd415: quad_corners = 76'd91960398992023762435;
428 12'd416: quad_corners = 76'd165458581135586242565;
429 12'd417: quad_corners = 76'd165313902172117936646;
430 12'd418: quad_corners = 76'd165169223208381195272;
431 12'd419: quad_corners = 76'd238811520539482660361;
432 12'd420: quad_corners = 76'd238666841575745918475;
433 12'd421: quad_corners = 76'd312165023443625184781;
434 12'd422: quad_corners = 76'd312020344479620007951;
435 12'd423: quad_corners = 76'd385662500798086774801;
436 12'd424: quad_corners = 76'd459304797854041896980;
437 12'd425: quad_corners = 76'd532802979996530634774;
438 12'd426: quad_corners = 76'd606445136314729490457;
439 12'd427: quad_corners = 76'd680087433370416177180;
440 12'd428: quad_corners = 76'd753729730151224956958;
441 12'd429: quad_corners = 76'd827227771281079520801;
442 12'd430: quad_corners = 76'd900870068061619865125;
443 12'd431: quad_corners = 76'd1048299200674119531560;
444 12'd432: quad_corners = 76'd1121941356717172044332;
445 12'd433: quad_corners = 76'd1195439397571879741487;
446 12'd434: quad_corners = 76'd1342868670646990380595;
447 12'd435: quad_corners = 76'd1490297802984343704631;
448 12'd436: quad_corners = 76'd1563795843838783490619;
449 12'd437: quad_corners = 76'd1563795843838783490619;
450 12'd438: quad_corners = 76'd1563795843838783490619;
451 12'd439: quad_corners = 76'd1563795843838783490619;
452 12'd440: quad_corners = 76'd1563795843838783490619;
453 12'd441: quad_corners = 76'd1563795843838783490619;
454 12'd442: quad_corners = 76'd1563795843838783490619;
455 12'd443: quad_corners = 76'd1563795843838783490619;
456 12'd444: quad_corners = 76'd1563795843838783490619;
457 12'd445: quad_corners = 76'd1563795843838783490619;
458 12'd446: quad_corners = 76'd1563795843838783490619;

459 12'd447: quad_corners = 76'd1563795843838783490619;
460 12'd448: quad_corners = 76'd166903815503556664320;
461 12'd449: quad_corners = 76'd166903815503556664320;
462 12'd450: quad_corners = 76'd166903815503556664320;
463 12'd451: quad_corners = 76'd166903815503556664320;
464 12'd452: quad_corners = 76'd166903815503556664320;
465 12'd453: quad_corners = 76'd166903815503556664320;
466 12'd454: quad_corners = 76'd166903815503556664320;
467 12'd455: quad_corners = 76'd166903815503556664320;
468 12'd456: quad_corners = 76'd166903815503556664320;
469 12'd457: quad_corners = 76'd166903815503556664320;
470 12'd458: quad_corners = 76'd166903815503556664320;
471 12'd459: quad_corners = 76'd166903815503556664320;
472 12'd460: quad_corners = 76'd166903815503556664320;
473 12'd461: quad_corners = 76'd166903815503556664320;
474 12'd462: quad_corners = 76'd166903815503556664320;
475 12'd463: quad_corners = 76'd166903815503556664320;
476 12'd464: quad_corners = 76'd166903815503556664320;
477 12'd465: quad_corners = 76'd166903815503556664320;
478 12'd466: quad_corners = 76'd166903815503556664320;
479 12'd467: quad_corners = 76'd166903815503556664320;
480 12'd468: quad_corners = 76'd166903815503556664320;
481 12'd469: quad_corners = 76'd166903815503556664320;
482 12'd470: quad_corners = 76'd166903815503556664320;
483 12'd471: quad_corners = 76'd166903815503556664320;
484 12'd472: quad_corners = 76'd166903815503556664320;
485 12'd473: quad_corners = 76'd166759137365526861312;
486 12'd474: quad_corners = 76'd92683508482071876096;
487 12'd475: quad_corners = 76'd92538830344042597376;
488 12'd476: quad_corners = 76'd92394292943501150208;
489 12'd477: quad_corners = 76'd92249614805471871489;
490 12'd478: quad_corners = 76'd92104936667442592770;
491 12'd479: quad_corners = 76'd91960398992023762435;
492 12'd480: quad_corners = 76'd165458581135586242565;
493 12'd481: quad_corners = 76'd165313902172117936646;
494 12'd482: quad_corners = 76'd165169223208381195272;
495 12'd483: quad_corners = 76'd238811520539482660361;
496 12'd484: quad_corners = 76'd238666841575745918475;
497 12'd485: quad_corners = 76'd312165023443625184781;
498 12'd486: quad_corners = 76'd312020344479620007951;
499 12'd487: quad_corners = 76'd385662500798086774801;
500 12'd488: quad_corners = 76'd459304797854041896980;
501 12'd489: quad_corners = 76'd532802979996530634774;
502 12'd490: quad_corners = 76'd606445136314729490457;
503 12'd491: quad_corners = 76'd680087433370416177180;
504 12'd492: quad_corners = 76'd753729730151224956958;

505 12'd493: quad_corners = 76'd827227771281079520801;
506 12'd494: quad_corners = 76'd900870068061619865125;
507 12'd495: quad_corners = 76'd1048299200674119531560;
508 12'd496: quad_corners = 76'd1121941356717172044332;
509 12'd497: quad_corners = 76'd1195439397571879741487;
510 12'd498: quad_corners = 76'd1342868670646990380595;
511 12'd499: quad_corners = 76'd1490297802984343704631;
512 12'd500: quad_corners = 76'd1563795843838783490619;
513 12'd501: quad_corners = 76'd1563795843838783490619;
514 12'd502: quad_corners = 76'd1563795843838783490619;
515 12'd503: quad_corners = 76'd1563795843838783490619;
516 12'd504: quad_corners = 76'd1563795843838783490619;
517 12'd505: quad_corners = 76'd1563795843838783490619;
518 12'd506: quad_corners = 76'd1563795843838783490619;
519 12'd507: quad_corners = 76'd1563795843838783490619;
520 12'd508: quad_corners = 76'd1563795843838783490619;
521 12'd509: quad_corners = 76'd1563795843838783490619;
522 12'd510: quad_corners = 76'd1563795843838783490619;
523 12'd511: quad_corners = 76'd1563795843838783490619;
524 12'd512: quad_corners = 76'd166903815503556664320;
525 12'd513: quad_corners = 76'd166903815503556664320;
526 12'd514: quad_corners = 76'd166903815503556664320;
527 12'd515: quad_corners = 76'd166903815503556664320;
528 12'd516: quad_corners = 76'd166903815503556664320;
529 12'd517: quad_corners = 76'd166903815503556664320;
530 12'd518: quad_corners = 76'd166903815503556664320;
531 12'd519: quad_corners = 76'd166903815503556664320;
532 12'd520: quad_corners = 76'd166903815503556664320;
533 12'd521: quad_corners = 76'd166903815503556664320;
534 12'd522: quad_corners = 76'd166903815503556664320;
535 12'd523: quad_corners = 76'd166903815503556664320;
536 12'd524: quad_corners = 76'd166903815503556664320;
537 12'd525: quad_corners = 76'd166903815503556664320;
538 12'd526: quad_corners = 76'd166903815503556664320;
539 12'd527: quad_corners = 76'd166903815503556664320;
540 12'd528: quad_corners = 76'd166903815503556664320;
541 12'd529: quad_corners = 76'd166903815503556664320;
542 12'd530: quad_corners = 76'd166903815503556664320;
543 12'd531: quad_corners = 76'd166903815503556664320;
544 12'd532: quad_corners = 76'd166903815503556664320;
545 12'd533: quad_corners = 76'd166903815503556664320;
546 12'd534: quad_corners = 76'd166903815503556664320;
547 12'd535: quad_corners = 76'd166903815503556664320;
548 12'd536: quad_corners = 76'd166903815503556664320;
549 12'd537: quad_corners = 76'd166759137365526861312;
550 12'd538: quad_corners = 76'd92683508482071876096;

551 12'd539: quad_corners = 76'd92538830344042597376;
552 12'd540: quad_corners = 76'd92394292943501150208;
553 12'd541: quad_corners = 76'd92249614805471871489;
554 12'd542: quad_corners = 76'd92104936667442592770;
555 12'd543: quad_corners = 76'd91960398992023762435;
556 12'd544: quad_corners = 76'd165458581135586242565;
557 12'd545: quad_corners = 76'd165313902172117936646;
558 12'd546: quad_corners = 76'd165169223208381195272;
559 12'd547: quad_corners = 76'd238811520539482660361;
560 12'd548: quad_corners = 76'd238666841575745918475;
561 12'd549: quad_corners = 76'd312165023443625184781;
562 12'd550: quad_corners = 76'd312020344479620007951;
563 12'd551: quad_corners = 76'd385662500798086774801;
564 12'd552: quad_corners = 76'd459304797854041896980;
565 12'd553: quad_corners = 76'd532802979996530634774;
566 12'd554: quad_corners = 76'd606445136314729490457;
567 12'd555: quad_corners = 76'd680087433370416177180;
568 12'd556: quad_corners = 76'd753729730151224956958;
569 12'd557: quad_corners = 76'd827227771281079520801;
570 12'd558: quad_corners = 76'd900870068061619865125;
571 12'd559: quad_corners = 76'd1048299200674119531560;
572 12'd560: quad_corners = 76'd1121941356717172044332;
573 12'd561: quad_corners = 76'd1195439397571879741487;
574 12'd562: quad_corners = 76'd1342868670646990380595;
575 12'd563: quad_corners = 76'd1490297802984343704631;
576 12'd564: quad_corners = 76'd1563795843838783490619;
577 12'd565: quad_corners = 76'd1563795843838783490619;
578 12'd566: quad_corners = 76'd1563795843838783490619;
579 12'd567: quad_corners = 76'd1563795843838783490619;
580 12'd568: quad_corners = 76'd1563795843838783490619;
581 12'd569: quad_corners = 76'd1563795843838783490619;
582 12'd570: quad_corners = 76'd1563795843838783490619;
583 12'd571: quad_corners = 76'd1563795843838783490619;
584 12'd572: quad_corners = 76'd1563795843838783490619;
585 12'd573: quad_corners = 76'd1563795843838783490619;
586 12'd574: quad_corners = 76'd1563795843838783490619;
587 12'd575: quad_corners = 76'd1563795843838783490619;
588 12'd576: quad_corners = 76'd166903815503556664320;
589 12'd577: quad_corners = 76'd166903815503556664320;
590 12'd578: quad_corners = 76'd166903815503556664320;
591 12'd579: quad_corners = 76'd166903815503556664320;
592 12'd580: quad_corners = 76'd166903815503556664320;
593 12'd581: quad_corners = 76'd166903815503556664320;
594 12'd582: quad_corners = 76'd166903815503556664320;
595 12'd583: quad_corners = 76'd166903815503556664320;
596 12'd584: quad_corners = 76'd166903815503556664320;

```

597      12'd585: quad_corners = 76'd166903815503556664320;
598      12'd586: quad_corners = 76'd166903815503556664320;
599      12'd587: quad_corners = 76'd166903815503556664320;
600      12'd588: quad_corners = 76'd166903815503556664320;
601      12'd589: quad_corners = 76'd166903815503556664320;
602      12'd590: quad_corners = 76'd166903815503556664320;
603      12'd591: quad_corners = 76'd166903815503556664320;
604      12'd592: quad_corners = 76'd166903815503556664320;
605      12'd593: quad_corners = 76'd166903815503556664320;
606      12'd594: quad_corners = 76'd166903815503556664320;
607      12'd595: quad_corners = 76'd166903815503556664320;
608      12'd596: quad_corners = 76'd166903815503556664320;
609      12'd597: quad_corners = 76'd166903815503556664320;
610      12'd598: quad_corners = 76'd166903815503556664320;
611      12'd599: quad_corners = 76'd166903815503556664320;
612      12'd600: quad_corners = 76'd166903815503556664320;
613      12'd601: quad_corners = 76'd166759137365526861312;
614      12'd602: quad_corners = 76'd92683508482071876096;
615      12'd603: quad_corners = 76'd92538830344042597376;
616      12'd604: quad_corners = 76'd92394292943501150208;
617      12'd605: quad_corners = 76'd92249614805471871489;
618      12'd606: quad_corners = 76'd92104936667442592770;
619      12'd607: quad_corners = 76'd91960398992023762435;
620      12'd608: quad_corners = 76'd165458581135586242565;
621      12'd609: quad_corners = 76'd165313902172117936646;
622      12'd610: quad_corners = 76'd165169223208381195272;
623      12'd611: quad_corners = 76'd238811520539482660361;
624      12'd612: quad_corners = 76'd238666841575745918475;
625      12'd613: quad_corners = 76'd312165023443625184781;
626      12'd614: quad_corners = 76'd312020344479620007951;
627      12'd615: quad_corners = 76'd385662500798086774801;
628      12'd616: quad_corners = 76'd459304797854041896980;
629      12'd617: quad_corners = 76'd532802979996530634774;
630      12'd618: quad_corners = 76'd606445136314729490457;
631      12'd619: quad_corners = 76'd680087433370416177180;
632      12'd620: quad_corners = 76'd753729730151224956958;
633      12'd621: quad_corners = 76'd827227771281079520801;
634      12'd622: quad_corners = 76'd900870068061619865125;
635      12'd623: quad_corners = 76'd1048299200674119531560;
636      12'd624: quad_corners = 76'd1121941356717172044332;
637      12'd625: quad_corners = 76'd1195439397571879741487;
638      12'd626: quad_corners = 76'd1342868670646990380595;
639      12'd627: quad_corners = 76'd1490297802984343704631;
640      12'd628: quad_corners = 76'd1563795843838783490619;
641      12'd629: quad_corners = 76'd1563795843838783490619;
642      12'd630: quad_corners = 76'd1563795843838783490619;

```

643 12'd631: quad_corners = 76'd1563795843838783490619;
644 12'd632: quad_corners = 76'd1563795843838783490619;
645 12'd633: quad_corners = 76'd1563795843838783490619;
646 12'd634: quad_corners = 76'd1563795843838783490619;
647 12'd635: quad_corners = 76'd1563795843838783490619;
648 12'd636: quad_corners = 76'd1563795843838783490619;
649 12'd637: quad_corners = 76'd1563795843838783490619;
650 12'd638: quad_corners = 76'd1563795843838783490619;
651 12'd639: quad_corners = 76'd1563795843838783490619;
652 12'd640: quad_corners = 76'd16690381503556664320;
653 12'd641: quad_corners = 76'd16690381503556664320;
654 12'd642: quad_corners = 76'd16690381503556664320;
655 12'd643: quad_corners = 76'd16690381503556664320;
656 12'd644: quad_corners = 76'd16690381503556664320;
657 12'd645: quad_corners = 76'd16690381503556664320;
658 12'd646: quad_corners = 76'd16690381503556664320;
659 12'd647: quad_corners = 76'd16690381503556664320;
660 12'd648: quad_corners = 76'd16690381503556664320;
661 12'd649: quad_corners = 76'd16690381503556664320;
662 12'd650: quad_corners = 76'd16690381503556664320;
663 12'd651: quad_corners = 76'd16690381503556664320;
664 12'd652: quad_corners = 76'd16690381503556664320;
665 12'd653: quad_corners = 76'd16690381503556664320;
666 12'd654: quad_corners = 76'd16690381503556664320;
667 12'd655: quad_corners = 76'd16690381503556664320;
668 12'd656: quad_corners = 76'd16690381503556664320;
669 12'd657: quad_corners = 76'd16690381503556664320;
670 12'd658: quad_corners = 76'd16690381503556664320;
671 12'd659: quad_corners = 76'd16690381503556664320;
672 12'd660: quad_corners = 76'd16690381503556664320;
673 12'd661: quad_corners = 76'd16690381503556664320;
674 12'd662: quad_corners = 76'd16690381503556664320;
675 12'd663: quad_corners = 76'd16690381503556664320;
676 12'd664: quad_corners = 76'd16690381503556664320;
677 12'd665: quad_corners = 76'd166759137365526861312;
678 12'd666: quad_corners = 76'd92683508482071876096;
679 12'd667: quad_corners = 76'd92538830344042597376;
680 12'd668: quad_corners = 76'd92394292943501150208;
681 12'd669: quad_corners = 76'd92249614805471871489;
682 12'd670: quad_corners = 76'd92104936667442592770;
683 12'd671: quad_corners = 76'd91960398992023762435;
684 12'd672: quad_corners = 76'd165458581135586242565;
685 12'd673: quad_corners = 76'd165313902172117936646;
686 12'd674: quad_corners = 76'd165169223208381195272;
687 12'd675: quad_corners = 76'd238811520539482660361;
688 12'd676: quad_corners = 76'd238666841575745918475;

689 12'd677: quad_corners = 76'd312165023443625184781;
690 12'd678: quad_corners = 76'd312020344479620007951;
691 12'd679: quad_corners = 76'd385662500798086774801;
692 12'd680: quad_corners = 76'd459304797854041896980;
693 12'd681: quad_corners = 76'd532802979996530634774;
694 12'd682: quad_corners = 76'd606445136314729490457;
695 12'd683: quad_corners = 76'd680087433370416177180;
696 12'd684: quad_corners = 76'd753729730151224956958;
697 12'd685: quad_corners = 76'd827227771281079520801;
698 12'd686: quad_corners = 76'd900870068061619865125;
699 12'd687: quad_corners = 76'd1048299200674119531560;
700 12'd688: quad_corners = 76'd1121941356717172044332;
701 12'd689: quad_corners = 76'd1195439397571879741487;
702 12'd690: quad_corners = 76'd1342868670646990380595;
703 12'd691: quad_corners = 76'd1490297802984343704631;
704 12'd692: quad_corners = 76'd1563795843838783490619;
705 12'd693: quad_corners = 76'd1563795843838783490619;
706 12'd694: quad_corners = 76'd1563795843838783490619;
707 12'd695: quad_corners = 76'd1563795843838783490619;
708 12'd696: quad_corners = 76'd1563795843838783490619;
709 12'd697: quad_corners = 76'd1563795843838783490619;
710 12'd698: quad_corners = 76'd1563795843838783490619;
711 12'd699: quad_corners = 76'd1563795843838783490619;
712 12'd700: quad_corners = 76'd1563795843838783490619;
713 12'd701: quad_corners = 76'd1563795843838783490619;
714 12'd702: quad_corners = 76'd1563795843838783490619;
715 12'd703: quad_corners = 76'd1563795843838783490619;
716 12'd704: quad_corners = 76'd166903815503556664320;
717 12'd705: quad_corners = 76'd166903815503556664320;
718 12'd706: quad_corners = 76'd166903815503556664320;
719 12'd707: quad_corners = 76'd166903815503556664320;
720 12'd708: quad_corners = 76'd166903815503556664320;
721 12'd709: quad_corners = 76'd166903815503556664320;
722 12'd710: quad_corners = 76'd166903815503556664320;
723 12'd711: quad_corners = 76'd166903815503556664320;
724 12'd712: quad_corners = 76'd166903815503556664320;
725 12'd713: quad_corners = 76'd166903815503556664320;
726 12'd714: quad_corners = 76'd166903815503556664320;
727 12'd715: quad_corners = 76'd166903815503556664320;
728 12'd716: quad_corners = 76'd166903815503556664320;
729 12'd717: quad_corners = 76'd166903815503556664320;
730 12'd718: quad_corners = 76'd166903815503556664320;
731 12'd719: quad_corners = 76'd166903815503556664320;
732 12'd720: quad_corners = 76'd166903815503556664320;
733 12'd721: quad_corners = 76'd166903815503556664320;
734 12'd722: quad_corners = 76'd166903815503556664320;

735 12'd723: quad_corners = 76'd166903815503556664320;
736 12'd724: quad_corners = 76'd166903815503556664320;
737 12'd725: quad_corners = 76'd166903815503556664320;
738 12'd726: quad_corners = 76'd166903815503556664320;
739 12'd727: quad_corners = 76'd166903815503556664320;
740 12'd728: quad_corners = 76'd166903815503556664320;
741 12'd729: quad_corners = 76'd166759137365526861312;
742 12'd730: quad_corners = 76'd92683508482071876096;
743 12'd731: quad_corners = 76'd92538830344042597376;
744 12'd732: quad_corners = 76'd92394292943501150208;
745 12'd733: quad_corners = 76'd92249614805471871489;
746 12'd734: quad_corners = 76'd92104936667442592770;
747 12'd735: quad_corners = 76'd91960398992023762435;
748 12'd736: quad_corners = 76'd165458581135586242565;
749 12'd737: quad_corners = 76'd165313902172117936646;
750 12'd738: quad_corners = 76'd165169223208381195272;
751 12'd739: quad_corners = 76'd238811520539482660361;
752 12'd740: quad_corners = 76'd238666841575745918475;
753 12'd741: quad_corners = 76'd312165023443625184781;
754 12'd742: quad_corners = 76'd312020344479620007951;
755 12'd743: quad_corners = 76'd385662500798086774801;
756 12'd744: quad_corners = 76'd459304797854041896980;
757 12'd745: quad_corners = 76'd532802979996530634774;
758 12'd746: quad_corners = 76'd606445136314729490457;
759 12'd747: quad_corners = 76'd680087433370416177180;
760 12'd748: quad_corners = 76'd753729730151224956958;
761 12'd749: quad_corners = 76'd827227771281079520801;
762 12'd750: quad_corners = 76'd900870068061619865125;
763 12'd751: quad_corners = 76'd1048299200674119531560;
764 12'd752: quad_corners = 76'd1121941356717172044332;
765 12'd753: quad_corners = 76'd1195439397571879741487;
766 12'd754: quad_corners = 76'd1342868670646990380595;
767 12'd755: quad_corners = 76'd1490297802984343704631;
768 12'd756: quad_corners = 76'd1563795843838783490619;
769 12'd757: quad_corners = 76'd1563795843838783490619;
770 12'd758: quad_corners = 76'd1563795843838783490619;
771 12'd759: quad_corners = 76'd1563795843838783490619;
772 12'd760: quad_corners = 76'd1563795843838783490619;
773 12'd761: quad_corners = 76'd1563795843838783490619;
774 12'd762: quad_corners = 76'd1563795843838783490619;
775 12'd763: quad_corners = 76'd1563795843838783490619;
776 12'd764: quad_corners = 76'd1563795843838783490619;
777 12'd765: quad_corners = 76'd1563795843838783490619;
778 12'd766: quad_corners = 76'd1563795843838783490619;
779 12'd767: quad_corners = 76'd1563795843838783490619;
780 12'd768: quad_corners = 76'd166903815503556664320;

781 12'd769: quad_corners = 76'd166903815503556664320;
782 12'd770: quad_corners = 76'd166903815503556664320;
783 12'd771: quad_corners = 76'd166903815503556664320;
784 12'd772: quad_corners = 76'd166903815503556664320;
785 12'd773: quad_corners = 76'd166903815503556664320;
786 12'd774: quad_corners = 76'd166903815503556664320;
787 12'd775: quad_corners = 76'd166903815503556664320;
788 12'd776: quad_corners = 76'd166903815503556664320;
789 12'd777: quad_corners = 76'd166903815503556664320;
790 12'd778: quad_corners = 76'd166903815503556664320;
791 12'd779: quad_corners = 76'd166903815503556664320;
792 12'd780: quad_corners = 76'd166903815503556664320;
793 12'd781: quad_corners = 76'd166903815503556664320;
794 12'd782: quad_corners = 76'd166903815503556664320;
795 12'd783: quad_corners = 76'd166903815503556664320;
796 12'd784: quad_corners = 76'd166903815503556664320;
797 12'd785: quad_corners = 76'd166903815503556664320;
798 12'd786: quad_corners = 76'd166903815503556664320;
799 12'd787: quad_corners = 76'd166903815503556664320;
800 12'd788: quad_corners = 76'd166903815503556664320;
801 12'd789: quad_corners = 76'd166903815503556664320;
802 12'd790: quad_corners = 76'd166903815503556664320;
803 12'd791: quad_corners = 76'd166903815503556664320;
804 12'd792: quad_corners = 76'd166903815503556664320;
805 12'd793: quad_corners = 76'd166759137365526861312;
806 12'd794: quad_corners = 76'd92683508482071876096;
807 12'd795: quad_corners = 76'd92538830344042597376;
808 12'd796: quad_corners = 76'd92394292943501150208;
809 12'd797: quad_corners = 76'd92249614805471871489;
810 12'd798: quad_corners = 76'd92104936667442592770;
811 12'd799: quad_corners = 76'd91960398992023762435;
812 12'd800: quad_corners = 76'd165458581135586242565;
813 12'd801: quad_corners = 76'd165313902172117936646;
814 12'd802: quad_corners = 76'd165169223208381195272;
815 12'd803: quad_corners = 76'd238811520539482660361;
816 12'd804: quad_corners = 76'd238666841575745918475;
817 12'd805: quad_corners = 76'd312165023443625184781;
818 12'd806: quad_corners = 76'd312020344479620007951;
819 12'd807: quad_corners = 76'd385662500798086774801;
820 12'd808: quad_corners = 76'd459304797854041896980;
821 12'd809: quad_corners = 76'd532802979996530634774;
822 12'd810: quad_corners = 76'd606445136314729490457;
823 12'd811: quad_corners = 76'd680087433370416177180;
824 12'd812: quad_corners = 76'd753729730151224956958;
825 12'd813: quad_corners = 76'd827227771281079520801;
826 12'd814: quad_corners = 76'd900870068061619865125;

827 12'd815: quad_corners = 76'd1048299200674119531560;
828 12'd816: quad_corners = 76'd1121941356717172044332;
829 12'd817: quad_corners = 76'd1195439397571879741487;
830 12'd818: quad_corners = 76'd1342868670646990380595;
831 12'd819: quad_corners = 76'd1490297802984343704631;
832 12'd820: quad_corners = 76'd1563795843838783490619;
833 12'd821: quad_corners = 76'd1563795843838783490619;
834 12'd822: quad_corners = 76'd1563795843838783490619;
835 12'd823: quad_corners = 76'd1563795843838783490619;
836 12'd824: quad_corners = 76'd1563795843838783490619;
837 12'd825: quad_corners = 76'd1563795843838783490619;
838 12'd826: quad_corners = 76'd1563795843838783490619;
839 12'd827: quad_corners = 76'd1563795843838783490619;
840 12'd828: quad_corners = 76'd1563795843838783490619;
841 12'd829: quad_corners = 76'd1563795843838783490619;
842 12'd830: quad_corners = 76'd1563795843838783490619;
843 12'd831: quad_corners = 76'd1563795843838783490619;
844 12'd832: quad_corners = 76'd166903815503556664320;
845 12'd833: quad_corners = 76'd166903815503556664320;
846 12'd834: quad_corners = 76'd166903815503556664320;
847 12'd835: quad_corners = 76'd166903815503556664320;
848 12'd836: quad_corners = 76'd166903815503556664320;
849 12'd837: quad_corners = 76'd166903815503556664320;
850 12'd838: quad_corners = 76'd166903815503556664320;
851 12'd839: quad_corners = 76'd166903815503556664320;
852 12'd840: quad_corners = 76'd166903815503556664320;
853 12'd841: quad_corners = 76'd166903815503556664320;
854 12'd842: quad_corners = 76'd166903815503556664320;
855 12'd843: quad_corners = 76'd166903815503556664320;
856 12'd844: quad_corners = 76'd166903815503556664320;
857 12'd845: quad_corners = 76'd166903815503556664320;
858 12'd846: quad_corners = 76'd166903815503556664320;
859 12'd847: quad_corners = 76'd166903815503556664320;
860 12'd848: quad_corners = 76'd166903815503556664320;
861 12'd849: quad_corners = 76'd166903815503556664320;
862 12'd850: quad_corners = 76'd166903815503556664320;
863 12'd851: quad_corners = 76'd166903815503556664320;
864 12'd852: quad_corners = 76'd166903815503556664320;
865 12'd853: quad_corners = 76'd166903815503556664320;
866 12'd854: quad_corners = 76'd166903815503556664320;
867 12'd855: quad_corners = 76'd166903815503556664320;
868 12'd856: quad_corners = 76'd166903815503556664320;
869 12'd857: quad_corners = 76'd166759137365526861312;
870 12'd858: quad_corners = 76'd92683508482071876096;
871 12'd859: quad_corners = 76'd92538830344042597376;
872 12'd860: quad_corners = 76'd92394292943501150208;

873 12'd861: quad_corners = 76'd92249614805471871489;
874 12'd862: quad_corners = 76'd92104936667442592770;
875 12'd863: quad_corners = 76'd91960398992023762435;
876 12'd864: quad_corners = 76'd165458581135586242565;
877 12'd865: quad_corners = 76'd165313902172117936646;
878 12'd866: quad_corners = 76'd165169223208381195272;
879 12'd867: quad_corners = 76'd238811520539482660361;
880 12'd868: quad_corners = 76'd238666841575745918475;
881 12'd869: quad_corners = 76'd312165023443625184781;
882 12'd870: quad_corners = 76'd312020344479620007951;
883 12'd871: quad_corners = 76'd385662500798086774801;
884 12'd872: quad_corners = 76'd459304797854041896980;
885 12'd873: quad_corners = 76'd532802979996530634774;
886 12'd874: quad_corners = 76'd606445136314729490457;
887 12'd875: quad_corners = 76'd680087433370416177180;
888 12'd876: quad_corners = 76'd753729730151224956958;
889 12'd877: quad_corners = 76'd827227771281079520801;
890 12'd878: quad_corners = 76'd900870068061619865125;
891 12'd879: quad_corners = 76'd1048299200674119531560;
892 12'd880: quad_corners = 76'd1121941356717172044332;
893 12'd881: quad_corners = 76'd1195439397571879741487;
894 12'd882: quad_corners = 76'd1342868670646990380595;
895 12'd883: quad_corners = 76'd1490297802984343704631;
896 12'd884: quad_corners = 76'd1563795843838783490619;
897 12'd885: quad_corners = 76'd1563795843838783490619;
898 12'd886: quad_corners = 76'd1563795843838783490619;
899 12'd887: quad_corners = 76'd1563795843838783490619;
900 12'd888: quad_corners = 76'd1563795843838783490619;
901 12'd889: quad_corners = 76'd1563795843838783490619;
902 12'd890: quad_corners = 76'd1563795843838783490619;
903 12'd891: quad_corners = 76'd1563795843838783490619;
904 12'd892: quad_corners = 76'd1563795843838783490619;
905 12'd893: quad_corners = 76'd1563795843838783490619;
906 12'd894: quad_corners = 76'd1563795843838783490619;
907 12'd895: quad_corners = 76'd1563795843838783490619;
908 12'd896: quad_corners = 76'd166903815503556664320;
909 12'd897: quad_corners = 76'd166903815503556664320;
910 12'd898: quad_corners = 76'd166903815503556664320;
911 12'd899: quad_corners = 76'd166903815503556664320;
912 12'd900: quad_corners = 76'd166903815503556664320;
913 12'd901: quad_corners = 76'd166903815503556664320;
914 12'd902: quad_corners = 76'd166903815503556664320;
915 12'd903: quad_corners = 76'd166903815503556664320;
916 12'd904: quad_corners = 76'd166903815503556664320;
917 12'd905: quad_corners = 76'd166903815503556664320;
918 12'd906: quad_corners = 76'd166903815503556664320;

919 12'd907: quad_corners = 76'd166903815503556664320;
920 12'd908: quad_corners = 76'd166903815503556664320;
921 12'd909: quad_corners = 76'd166903815503556664320;
922 12'd910: quad_corners = 76'd166903815503556664320;
923 12'd911: quad_corners = 76'd166903815503556664320;
924 12'd912: quad_corners = 76'd166903815503556664320;
925 12'd913: quad_corners = 76'd166903815503556664320;
926 12'd914: quad_corners = 76'd166903815503556664320;
927 12'd915: quad_corners = 76'd166903815503556664320;
928 12'd916: quad_corners = 76'd166903815503556664320;
929 12'd917: quad_corners = 76'd166903815503556664320;
930 12'd918: quad_corners = 76'd166903815503556664320;
931 12'd919: quad_corners = 76'd166903815503556664320;
932 12'd920: quad_corners = 76'd166903815503556664320;
933 12'd921: quad_corners = 76'd166759137365526861312;
934 12'd922: quad_corners = 76'd92683508482071876096;
935 12'd923: quad_corners = 76'd92538830344042597376;
936 12'd924: quad_corners = 76'd92394292943501150208;
937 12'd925: quad_corners = 76'd92249614805471871489;
938 12'd926: quad_corners = 76'd92104936667442592770;
939 12'd927: quad_corners = 76'd91960398992023762435;
940 12'd928: quad_corners = 76'd165458581135586242565;
941 12'd929: quad_corners = 76'd165313902172117936646;
942 12'd930: quad_corners = 76'd165169223208381195272;
943 12'd931: quad_corners = 76'd238811520539482660361;
944 12'd932: quad_corners = 76'd238666841575745918475;
945 12'd933: quad_corners = 76'd312165023443625184781;
946 12'd934: quad_corners = 76'd312020344479620007951;
947 12'd935: quad_corners = 76'd385662500798086774801;
948 12'd936: quad_corners = 76'd459304797854041896980;
949 12'd937: quad_corners = 76'd532802979996530634774;
950 12'd938: quad_corners = 76'd606445136314729490457;
951 12'd939: quad_corners = 76'd680087433370416177180;
952 12'd940: quad_corners = 76'd753729730151224956958;
953 12'd941: quad_corners = 76'd827227771281079520801;
954 12'd942: quad_corners = 76'd900870068061619865125;
955 12'd943: quad_corners = 76'd1048299200674119531560;
956 12'd944: quad_corners = 76'd1121941356717172044332;
957 12'd945: quad_corners = 76'd1195439397571879741487;
958 12'd946: quad_corners = 76'd1342868670646990380595;
959 12'd947: quad_corners = 76'd1490297802984343704631;
960 12'd948: quad_corners = 76'd1563795843838783490619;
961 12'd949: quad_corners = 76'd1563795843838783490619;
962 12'd950: quad_corners = 76'd1563795843838783490619;
963 12'd951: quad_corners = 76'd1563795843838783490619;
964 12'd952: quad_corners = 76'd1563795843838783490619;

965 12'd953: quad_corners = 76'd1563795843838783490619;
966 12'd954: quad_corners = 76'd1563795843838783490619;
967 12'd955: quad_corners = 76'd1563795843838783490619;
968 12'd956: quad_corners = 76'd1563795843838783490619;
969 12'd957: quad_corners = 76'd1563795843838783490619;
970 12'd958: quad_corners = 76'd1563795843838783490619;
971 12'd959: quad_corners = 76'd1563795843838783490619;
972 12'd960: quad_corners = 76'd166903815503556664320;
973 12'd961: quad_corners = 76'd166903815503556664320;
974 12'd962: quad_corners = 76'd166903815503556664320;
975 12'd963: quad_corners = 76'd166903815503556664320;
976 12'd964: quad_corners = 76'd166903815503556664320;
977 12'd965: quad_corners = 76'd166903815503556664320;
978 12'd966: quad_corners = 76'd166903815503556664320;
979 12'd967: quad_corners = 76'd166903815503556664320;
980 12'd968: quad_corners = 76'd166903815503556664320;
981 12'd969: quad_corners = 76'd166903815503556664320;
982 12'd970: quad_corners = 76'd166903815503556664320;
983 12'd971: quad_corners = 76'd166903815503556664320;
984 12'd972: quad_corners = 76'd166903815503556664320;
985 12'd973: quad_corners = 76'd166903815503556664320;
986 12'd974: quad_corners = 76'd166903815503556664320;
987 12'd975: quad_corners = 76'd166903815503556664320;
988 12'd976: quad_corners = 76'd166903815503556664320;
989 12'd977: quad_corners = 76'd166903815503556664320;
990 12'd978: quad_corners = 76'd166903815503556664320;
991 12'd979: quad_corners = 76'd166903815503556664320;
992 12'd980: quad_corners = 76'd166903815503556664320;
993 12'd981: quad_corners = 76'd166903815503556664320;
994 12'd982: quad_corners = 76'd166903815503556664320;
995 12'd983: quad_corners = 76'd166903815503556664320;
996 12'd984: quad_corners = 76'd166903815503556664320;
997 12'd985: quad_corners = 76'd166759137365526861312;
998 12'd986: quad_corners = 76'd92683508482071876096;
999 12'd987: quad_corners = 76'd92538830344042597376;
1000 12'd988: quad_corners = 76'd92394292943501150208;
1001 12'd989: quad_corners = 76'd92249614805471871489;
1002 12'd990: quad_corners = 76'd92104936667442592770;
1003 12'd991: quad_corners = 76'd91960398992023762435;
1004 12'd992: quad_corners = 76'd165458581135586242565;
1005 12'd993: quad_corners = 76'd165313902172117936646;
1006 12'd994: quad_corners = 76'd165169223208381195272;
1007 12'd995: quad_corners = 76'd238811520539482660361;
1008 12'd996: quad_corners = 76'd238666841575745918475;
1009 12'd997: quad_corners = 76'd312165023443625184781;
1010 12'd998: quad_corners = 76'd312020344479620007951;

```
1011 12'd999: quad_corners = 76'd385662500798086774801;
1012 12'd1000: quad_corners = 76'd459304797854041896980;
1013 12'd1001: quad_corners = 76'd532802979996530634774;
1014 12'd1002: quad_corners = 76'd606445136314729490457;
1015 12'd1003: quad_corners = 76'd680087433370416177180;
1016 12'd1004: quad_corners = 76'd753729730151224956958;
1017 12'd1005: quad_corners = 76'd827227771281079520801;
1018 12'd1006: quad_corners = 76'd900870068061619865125;
1019 12'd1007: quad_corners = 76'd1048299200674119531560;
1020 12'd1008: quad_corners = 76'd1121941356717172044332;
1021 12'd1009: quad_corners = 76'd1195439397571879741487;
1022 12'd1010: quad_corners = 76'd1342868670646990380595;
1023 12'd1011: quad_corners = 76'd1490297802984343704631;
1024 12'd1012: quad_corners = 76'd1563795843838783490619;
1025 12'd1013: quad_corners = 76'd1563795843838783490619;
1026 12'd1014: quad_corners = 76'd1563795843838783490619;
1027 12'd1015: quad_corners = 76'd1563795843838783490619;
1028 12'd1016: quad_corners = 76'd1563795843838783490619;
1029 12'd1017: quad_corners = 76'd1563795843838783490619;
1030 12'd1018: quad_corners = 76'd1563795843838783490619;
1031 12'd1019: quad_corners = 76'd1563795843838783490619;
1032 12'd1020: quad_corners = 76'd1563795843838783490619;
1033 12'd1021: quad_corners = 76'd1563795843838783490619;
1034 12'd1022: quad_corners = 76'd1563795843838783490619;
1035 12'd1023: quad_corners = 76'd1563795843838783490619;
1036 12'd1024: quad_corners = 76'd166903815503556664320;
1037 12'd1025: quad_corners = 76'd166903815503556664320;
1038 12'd1026: quad_corners = 76'd166903815503556664320;
1039 12'd1027: quad_corners = 76'd166903815503556664320;
1040 12'd1028: quad_corners = 76'd166903815503556664320;
1041 12'd1029: quad_corners = 76'd166903815503556664320;
1042 12'd1030: quad_corners = 76'd166903815503556664320;
1043 12'd1031: quad_corners = 76'd166903815503556664320;
1044 12'd1032: quad_corners = 76'd166903815503556664320;
1045 12'd1033: quad_corners = 76'd166903815503556664320;
1046 12'd1034: quad_corners = 76'd166903815503556664320;
1047 12'd1035: quad_corners = 76'd166903815503556664320;
1048 12'd1036: quad_corners = 76'd166903815503556664320;
1049 12'd1037: quad_corners = 76'd166903815503556664320;
1050 12'd1038: quad_corners = 76'd166903815503556664320;
1051 12'd1039: quad_corners = 76'd166903815503556664320;
1052 12'd1040: quad_corners = 76'd166903815503556664320;
1053 12'd1041: quad_corners = 76'd166903815503556664320;
1054 12'd1042: quad_corners = 76'd166903815503556664320;
1055 12'd1043: quad_corners = 76'd166903815503556664320;
1056 12'd1044: quad_corners = 76'd166903815503556664320;
```

1057 12'd1045: quad_corners = 76'd166903815503556664320;
1058 12'd1046: quad_corners = 76'd166903815503556664320;
1059 12'd1047: quad_corners = 76'd166903815503556664320;
1060 12'd1048: quad_corners = 76'd166903815503556664320;
1061 12'd1049: quad_corners = 76'd166759137365526861312;
1062 12'd1050: quad_corners = 76'd92683508482071876096;
1063 12'd1051: quad_corners = 76'd92538830344042597376;
1064 12'd1052: quad_corners = 76'd92394292943501150208;
1065 12'd1053: quad_corners = 76'd92249614805471871489;
1066 12'd1054: quad_corners = 76'd92104936667442592770;
1067 12'd1055: quad_corners = 76'd91960398992023762435;
1068 12'd1056: quad_corners = 76'd165458581135586242565;
1069 12'd1057: quad_corners = 76'd165313902172117936646;
1070 12'd1058: quad_corners = 76'd165169223208381195272;
1071 12'd1059: quad_corners = 76'd238811520539482660362;
1072 12'd1060: quad_corners = 76'd238666841575745918475;
1073 12'd1061: quad_corners = 76'd312165023443625184781;
1074 12'd1062: quad_corners = 76'd312020344479620007951;
1075 12'd1063: quad_corners = 76'd385662500798086774801;
1076 12'd1064: quad_corners = 76'd459304797854041896980;
1077 12'd1065: quad_corners = 76'd532802979996530634774;
1078 12'd1066: quad_corners = 76'd606445136314729490457;
1079 12'd1067: quad_corners = 76'd680087433370416177180;
1080 12'd1068: quad_corners = 76'd753729730151224956958;
1081 12'd1069: quad_corners = 76'd827227771281079520801;
1082 12'd1070: quad_corners = 76'd900870068061619865125;
1083 12'd1071: quad_corners = 76'd1048299200674119531560;
1084 12'd1072: quad_corners = 76'd1121941356717172044332;
1085 12'd1073: quad_corners = 76'd1195439397571879741487;
1086 12'd1074: quad_corners = 76'd1342868670646990380595;
1087 12'd1075: quad_corners = 76'd1490297802984343704631;
1088 12'd1076: quad_corners = 76'd1563795843838783490619;
1089 12'd1077: quad_corners = 76'd1563795843838783490619;
1090 12'd1078: quad_corners = 76'd1563795843838783490619;
1091 12'd1079: quad_corners = 76'd1563795843838783490619;
1092 12'd1080: quad_corners = 76'd1563795843838783490619;
1093 12'd1081: quad_corners = 76'd1563795843838783490619;
1094 12'd1082: quad_corners = 76'd1563795843838783490619;
1095 12'd1083: quad_corners = 76'd1563795843838783490619;
1096 12'd1084: quad_corners = 76'd1563795843838783490619;
1097 12'd1085: quad_corners = 76'd1563795843838783490619;
1098 12'd1086: quad_corners = 76'd1563795843838783490619;
1099 12'd1087: quad_corners = 76'd1563795843838783490619;
1100 12'd1088: quad_corners = 76'd17455098831505222144;
1101 12'd1089: quad_corners = 76'd17455098831505222144;
1102 12'd1090: quad_corners = 76'd17455098831505222144;

1103 12'd1091: quad_corners = 76'd17455098831505222144;
1104 12'd1092: quad_corners = 76'd17455098831505222144;
1105 12'd1093: quad_corners = 76'd17455098831505222144;
1106 12'd1094: quad_corners = 76'd17455098831505222144;
1107 12'd1095: quad_corners = 76'd17455098831505222144;
1108 12'd1096: quad_corners = 76'd17455098831505222144;
1109 12'd1097: quad_corners = 76'd17455098831505222144;
1110 12'd1098: quad_corners = 76'd17455098831505222144;
1111 12'd1099: quad_corners = 76'd17455098831505222144;
1112 12'd1100: quad_corners = 76'd17455098831505222144;
1113 12'd1101: quad_corners = 76'd17455098831505222144;
1114 12'd1102: quad_corners = 76'd17455098831505222144;
1115 12'd1103: quad_corners = 76'd17455098831505222144;
1116 12'd1104: quad_corners = 76'd17455098831505222144;
1117 12'd1105: quad_corners = 76'd17455098831505222144;
1118 12'd1106: quad_corners = 76'd17455098831505222144;
1119 12'd1107: quad_corners = 76'd17455098831505222144;
1120 12'd1108: quad_corners = 76'd17455098831505222144;
1121 12'd1109: quad_corners = 76'd17455098831505222144;
1122 12'd1110: quad_corners = 76'd17455098831505222144;
1123 12'd1111: quad_corners = 76'd17455098831505222144;
1124 12'd1112: quad_corners = 76'd17455098831505222144;
1125 12'd1113: quad_corners = 76'd17310420693475419648;
1126 12'd1114: quad_corners = 76'd17165883292934496768;
1127 12'd1115: quad_corners = 76'd17021205154905218048;
1128 12'd1116: quad_corners = 76'd16876667754364295168;
1129 12'd1117: quad_corners = 76'd16731989616334492160;
1130 12'd1118: quad_corners = 76'd16731567403869424640;
1131 12'd1119: quad_corners = 76'd16586889265840146432;
1132 12'd1120: quad_corners = 76'd16442351590421316098;
1133 12'd1121: quad_corners = 76'd16297672901830917123;
1134 12'd1122: quad_corners = 76'd89939970233200817669;
1135 12'd1123: quad_corners = 76'd89795291269464075783;
1136 12'd1124: quad_corners = 76'd163581844526129752073;
1137 12'd1125: quad_corners = 76'd163437165562393010699;
1138 12'd1126: quad_corners = 76'd237079462893494475277;
1139 12'd1127: quad_corners = 76'd310721759949718033423;
1140 12'd1128: quad_corners = 76'd384508172468626918418;
1141 12'd1129: quad_corners = 76'd458150469524582040597;
1142 12'd1130: quad_corners = 76'd531792766855146634264;
1143 12'd1131: quad_corners = 76'd605579179099178136603;
1144 12'd1132: quad_corners = 76'd679221476154864823326;
1145 12'd1133: quad_corners = 76'd752863773210283074593;
1146 12'd1134: quad_corners = 76'd900437021011395992612;
1147 12'd1135: quad_corners = 76'd974079318066814243880;
1148 12'd1136: quad_corners = 76'd1121508591417071225388;

1149 12'd1137: quad_corners = 76'd1195294862648199070256;
1150 12'd1138: quad_corners = 76'd1342724135998187616308;
1151 12'd1139: quad_corners = 76'd1490153409073566166584;
1152 12'd1140: quad_corners = 76'd1637726656874142213180;
1153 12'd1141: quad_corners = 76'd1637726656874142213180;
1154 12'd1142: quad_corners = 76'd1637726656874142213180;
1155 12'd1143: quad_corners = 76'd1637726656874142213180;
1156 12'd1144: quad_corners = 76'd1637726656874142213180;
1157 12'd1145: quad_corners = 76'd1637726656874142213180;
1158 12'd1146: quad_corners = 76'd1637726656874142213180;
1159 12'd1147: quad_corners = 76'd1637726656874142213180;
1160 12'd1148: quad_corners = 76'd1637726656874142213180;
1161 12'd1149: quad_corners = 76'd1637726656874142213180;
1162 12'd1150: quad_corners = 76'd1637726656874142213180;
1163 12'd1151: quad_corners = 76'd1637726656874142213180;
1164 12'd1152: quad_corners = 76'd15724449937206048768;
1165 12'd1153: quad_corners = 76'd15724449937206048768;
1166 12'd1154: quad_corners = 76'd15724449937206048768;
1167 12'd1155: quad_corners = 76'd15724449937206048768;
1168 12'd1156: quad_corners = 76'd15724449937206048768;
1169 12'd1157: quad_corners = 76'd15724449937206048768;
1170 12'd1158: quad_corners = 76'd15724449937206048768;
1171 12'd1159: quad_corners = 76'd15724449937206048768;
1172 12'd1160: quad_corners = 76'd15724449937206048768;
1173 12'd1161: quad_corners = 76'd15724449937206048768;
1174 12'd1162: quad_corners = 76'd15724449937206048768;
1175 12'd1163: quad_corners = 76'd15724449937206048768;
1176 12'd1164: quad_corners = 76'd15724449937206048768;
1177 12'd1165: quad_corners = 76'd15724449937206048768;
1178 12'd1166: quad_corners = 76'd15724449937206048768;
1179 12'd1167: quad_corners = 76'd15724449937206048768;
1180 12'd1168: quad_corners = 76'd15724449937206048768;
1181 12'd1169: quad_corners = 76'd15724449937206048768;
1182 12'd1170: quad_corners = 76'd15724449937206048768;
1183 12'd1171: quad_corners = 76'd15724449937206048768;
1184 12'd1172: quad_corners = 76'd15724449937206048768;
1185 12'd1173: quad_corners = 76'd15724449937206048768;
1186 12'd1174: quad_corners = 76'd15724449937206048768;
1187 12'd1175: quad_corners = 76'd15724449937206048768;
1188 12'd1176: quad_corners = 76'd15724449937206048768;
1189 12'd1177: quad_corners = 76'd15579771799176246272;
1190 12'd1178: quad_corners = 76'd15435234398635323392;
1191 12'd1179: quad_corners = 76'd15434812186169732096;
1192 12'd1180: quad_corners = 76'd15290134048140453376;
1193 12'd1181: quad_corners = 76'd15145596647599530496;
1194 12'd1182: quad_corners = 76'd15145174435134462976;

1195 12'd1183: quad_corners = 76'd15000496297105184768;
1196 12'd1184: quad_corners = 76'd15000074084639592960;
1197 12'd1185: quad_corners = 76'd14855395946341878785;
1198 12'd1186: quad_corners = 76'd14854972908437784067;
1199 12'd1187: quad_corners = 76'd14710293944969478149;
1200 12'd1188: quad_corners = 76'd14709871181943290375;
1201 12'd1189: quad_corners = 76'd88496283701120611337;
1202 12'd1190: quad_corners = 76'd162138581032222075915;
1203 12'd1191: quad_corners = 76'd235925134288887752206;
1204 12'd1192: quad_corners = 76'd309711546808065073169;
1205 12'd1193: quad_corners = 76'd383497959326974482452;
1206 12'd1194: quad_corners = 76'd457140397120417959959;
1207 12'd1195: quad_corners = 76'd530926809639326845466;
1208 12'd1196: quad_corners = 76'd678500198453073936925;
1209 12'd1197: quad_corners = 76'd752286610696837003809;
1210 12'd1198: quad_corners = 76'd826073022940867981860;
1211 12'd1199: quad_corners = 76'd973646411754078202408;
1212 12'd1200: quad_corners = 76'd1121219941030167831084;
1213 12'd1201: quad_corners = 76'd1195006353273661937712;
1214 12'd1202: quad_corners = 76'd1342579741812263211061;
1215 12'd1203: quad_corners = 76'd1490153130350327088697;
1216 12'd1204: quad_corners = 76'd1637726518613782019646;
1217 12'd1205: quad_corners = 76'd1637726518613782019646;
1218 12'd1206: quad_corners = 76'd1637726518613782019646;
1219 12'd1207: quad_corners = 76'd1637726518613782019646;
1220 12'd1208: quad_corners = 76'd1637726518613782019646;
1221 12'd1209: quad_corners = 76'd1637726518613782019646;
1222 12'd1210: quad_corners = 76'd1637726518613782019646;
1223 12'd1211: quad_corners = 76'd1637726518613782019646;
1224 12'd1212: quad_corners = 76'd1637726518613782019646;
1225 12'd1213: quad_corners = 76'd1637726518613782019646;
1226 12'd1214: quad_corners = 76'd1637726518613782019646;
1227 12'd1215: quad_corners = 76'd1637726518613782019646;
1228 12'd1216: quad_corners = 76'd13993801042906351104;
1229 12'd1217: quad_corners = 76'd13993801042906351104;
1230 12'd1218: quad_corners = 76'd13993801042906351104;
1231 12'd1219: quad_corners = 76'd13993801042906351104;
1232 12'd1220: quad_corners = 76'd13993801042906351104;
1233 12'd1221: quad_corners = 76'd13993801042906351104;
1234 12'd1222: quad_corners = 76'd13993801042906351104;
1235 12'd1223: quad_corners = 76'd13993801042906351104;
1236 12'd1224: quad_corners = 76'd13993801042906351104;
1237 12'd1225: quad_corners = 76'd13993801042906351104;
1238 12'd1226: quad_corners = 76'd13993801042906351104;
1239 12'd1227: quad_corners = 76'd13993801042906351104;
1240 12'd1228: quad_corners = 76'd13993801042906351104;

1241 12'd1229: quad_corners = 76'd13993801042906351104;
1242 12'd1230: quad_corners = 76'd13993801042906351104;
1243 12'd1231: quad_corners = 76'd13993801042906351104;
1244 12'd1232: quad_corners = 76'd13993801042906351104;
1245 12'd1233: quad_corners = 76'd13993801042906351104;
1246 12'd1234: quad_corners = 76'd13993801042906351104;
1247 12'd1235: quad_corners = 76'd13993801042906351104;
1248 12'd1236: quad_corners = 76'd13993801042906351104;
1249 12'd1237: quad_corners = 76'd13993801042906351104;
1250 12'd1238: quad_corners = 76'd13993801042906351104;
1251 12'd1239: quad_corners = 76'd13993801042906351104;
1252 12'd1240: quad_corners = 76'd13993801042906351104;
1253 12'd1241: quad_corners = 76'd13993378830440759808;
1254 12'd1242: quad_corners = 76'd13848841429899836928;
1255 12'd1243: quad_corners = 76'd13848419217434769920;
1256 12'd1244: quad_corners = 76'd13703741079404967424;
1257 12'd1245: quad_corners = 76'd13703318866939899904;
1258 12'd1246: quad_corners = 76'd13702896654474832896;
1259 12'd1247: quad_corners = 76'd13558359253933909504;
1260 12'd1248: quad_corners = 76'd13557796303979962880;
1261 12'd1249: quad_corners = 76'd13557374091514895360;
1262 12'd1250: quad_corners = 76'd13556951878512956929;
1263 12'd1251: quad_corners = 76'd13556388378266849283;
1264 12'd1252: quad_corners = 76'd13555965340362754565;
1265 12'd1253: quad_corners = 76'd13555542577336566791;
1266 12'd1254: quad_corners = 76'd87341955096513887754;
1267 12'd1255: quad_corners = 76'd161128508353447999501;
1268 12'd1256: quad_corners = 76'd234914920872625320464;
1269 12'd1257: quad_corners = 76'd308845589592245283347;
1270 12'd1258: quad_corners = 76'd456418978405992375318;
1271 12'd1259: quad_corners = 76'd530205531662658051097;
1272 12'd1260: quad_corners = 76'd603992084919055291933;
1273 12'd1261: quad_corners = 76'd751709588646000332320;
1274 12'd1262: quad_corners = 76'd825496141902398096932;
1275 12'd1263: quad_corners = 76'd973213645903952609320;
1276 12'd1264: quad_corners = 76'd1120787175180042237997;
1277 12'd1265: quad_corners = 76'd1268504679181596749873;
1278 12'd1266: quad_corners = 76'd1416222182908273354293;
1279 12'd1267: quad_corners = 76'd1563795712184094547514;
1280 12'd1268: quad_corners = 76'd1711513215910771152447;
1281 12'd1269: quad_corners = 76'd1711513215910771152447;
1282 12'd1270: quad_corners = 76'd1711513215910771152447;
1283 12'd1271: quad_corners = 76'd1711513215910771152447;
1284 12'd1272: quad_corners = 76'd1711513215910771152447;
1285 12'd1273: quad_corners = 76'd1711513215910771152447;
1286 12'd1274: quad_corners = 76'd1711513215910771152447;

1287 12'd1275: quad_corners = 76'd1711513215910771152447;
1288 12'd1276: quad_corners = 76'd1711513215910771152447;
1289 12'd1277: quad_corners = 76'd1711513215910771152447;
1290 12'd1278: quad_corners = 76'd1711513215910771152447;
1291 12'd1279: quad_corners = 76'd1711513215910771152447;
1292 12'd1280: quad_corners = 76'd12407408074170340352;
1293 12'd1281: quad_corners = 76'd12407408074170340352;
1294 12'd1282: quad_corners = 76'd12407408074170340352;
1295 12'd1283: quad_corners = 76'd12407408074170340352;
1296 12'd1284: quad_corners = 76'd12407408074170340352;
1297 12'd1285: quad_corners = 76'd12407408074170340352;
1298 12'd1286: quad_corners = 76'd12407408074170340352;
1299 12'd1287: quad_corners = 76'd12407408074170340352;
1300 12'd1288: quad_corners = 76'd12407408074170340352;
1301 12'd1289: quad_corners = 76'd12407408074170340352;
1302 12'd1290: quad_corners = 76'd12407408074170340352;
1303 12'd1291: quad_corners = 76'd12407408074170340352;
1304 12'd1292: quad_corners = 76'd12407408074170340352;
1305 12'd1293: quad_corners = 76'd12407408074170340352;
1306 12'd1294: quad_corners = 76'd12407408074170340352;
1307 12'd1295: quad_corners = 76'd12407408074170340352;
1308 12'd1296: quad_corners = 76'd12407408074170340352;
1309 12'd1297: quad_corners = 76'd12407408074170340352;
1310 12'd1298: quad_corners = 76'd12407408074170340352;
1311 12'd1299: quad_corners = 76'd12407408074170340352;
1312 12'd1300: quad_corners = 76'd12407408074170340352;
1313 12'd1301: quad_corners = 76'd12407408074170340352;
1314 12'd1302: quad_corners = 76'd12407408074170340352;
1315 12'd1303: quad_corners = 76'd12407408074170340352;
1316 12'd1304: quad_corners = 76'd12407408074170340352;
1317 12'd1305: quad_corners = 76'd12406985861705273344;
1318 12'd1306: quad_corners = 76'd12262448461164350464;
1319 12'd1307: quad_corners = 76'd12262026248698759168;
1320 12'd1308: quad_corners = 76'd12261604036233691648;
1321 12'd1309: quad_corners = 76'd12261041086280269312;
1322 12'd1310: quad_corners = 76'd12260618873814678016;
1323 12'd1311: quad_corners = 76'd12260196661349610496;
1324 12'd1312: quad_corners = 76'd12259774448884543488;
1325 12'd1313: quad_corners = 76'd12259352236419475968;
1326 12'd1314: quad_corners = 76'd12258930023954408960;
1327 12'd1315: quad_corners = 76'd12402622999028326401;
1328 12'd1316: quad_corners = 76'd12402200236002138628;
1329 12'd1317: quad_corners = 76'd12401636735756030982;
1330 12'd1318: quad_corners = 76'd86332305180765999113;
1331 12'd1319: quad_corners = 76'd160118858712309582348;
1332 12'd1320: quad_corners = 76'd234049527157319549455;

```

1333      12'd1321: quad_corners = 76'd307980195876938988562;
1334      12'd1322: quad_corners = 76'd381766749133873100309;
1335      12'd1323: quad_corners = 76'd529484394148062310425;
1336      12'd1324: quad_corners = 76'd603414921855316011037;
1337      12'd1325: quad_corners = 76'd751132566594895749665;
1338      12'd1326: quad_corners = 76'd825063235039368845861;
1339      12'd1327: quad_corners = 76'd972780879778680149033;
1340      12'd1328: quad_corners = 76'd1120498524517991975981;
1341      12'd1329: quad_corners = 76'd1268216028519546487858;
1342      12'd1330: quad_corners = 76'd1415933673258858315319;
1343      12'd1331: quad_corners = 76'd1563795432911099131451;
1344      12'd1332: quad_corners = 76'd1711513077650142522945;
1345      12'd1333: quad_corners = 76'd1711513077650142522945;
1346      12'd1334: quad_corners = 76'd1711513077650142522945;
1347      12'd1335: quad_corners = 76'd1711513077650142522945;
1348      12'd1336: quad_corners = 76'd1711513077650142522945;
1349      12'd1337: quad_corners = 76'd1711513077650142522945;
1350      12'd1338: quad_corners = 76'd1711513077650142522945;
1351      12'd1339: quad_corners = 76'd1711513077650142522945;
1352      12'd1340: quad_corners = 76'd1711513077650142522945;
1353      12'd1341: quad_corners = 76'd1711513077650142522945;
1354      12'd1342: quad_corners = 76'd1711513077650142522945;
1355      12'd1343: quad_corners = 76'd1711513077650142522945;
1356      12'd1344: quad_corners = 76'd10821015105434329088;
1357      12'd1345: quad_corners = 76'd10821015105434329088;
1358      12'd1346: quad_corners = 76'd10821015105434329088;
1359      12'd1347: quad_corners = 76'd10821015105434329088;
1360      12'd1348: quad_corners = 76'd10821015105434329088;
1361      12'd1349: quad_corners = 76'd10821015105434329088;
1362      12'd1350: quad_corners = 76'd10821015105434329088;
1363      12'd1351: quad_corners = 76'd10821015105434329088;
1364      12'd1352: quad_corners = 76'd10821015105434329088;
1365      12'd1353: quad_corners = 76'd10821015105434329088;
1366      12'd1354: quad_corners = 76'd10821015105434329088;
1367      12'd1355: quad_corners = 76'd10821015105434329088;
1368      12'd1356: quad_corners = 76'd10821015105434329088;
1369      12'd1357: quad_corners = 76'd10821015105434329088;
1370      12'd1358: quad_corners = 76'd10821015105434329088;
1371      12'd1359: quad_corners = 76'd10821015105434329088;
1372      12'd1360: quad_corners = 76'd10821015105434329088;
1373      12'd1361: quad_corners = 76'd10821015105434329088;
1374      12'd1362: quad_corners = 76'd10821015105434329088;
1375      12'd1363: quad_corners = 76'd10821015105434329088;
1376      12'd1364: quad_corners = 76'd10821015105434329088;
1377      12'd1365: quad_corners = 76'd10821015105434329088;
1378      12'd1366: quad_corners = 76'd10821015105434329088;

```

1379 12'd1367: quad_corners = 76'd10821015105434329088;
1380 12'd1368: quad_corners = 76'd10821015105434329088;
1381 12'd1369: quad_corners = 76'd10820592892969262080;
1382 12'd1370: quad_corners = 76'd10820170680504195072;
1383 12'd1371: quad_corners = 76'd10819748468038603776;
1384 12'd1372: quad_corners = 76'd10819326255573536768;
1385 12'd1373: quad_corners = 76'd10818904043108469760;
1386 12'd1374: quad_corners = 76'd10962597018718734336;
1387 12'd1375: quad_corners = 76'd10962174806253666816;
1388 12'd1376: quad_corners = 76'd10961752593788599808;
1389 12'd1377: quad_corners = 76'd11105445569399388672;
1390 12'd1378: quad_corners = 76'd11105023356934321152;
1391 12'd1379: quad_corners = 76'd11248716332545110016;
1392 12'd1380: quad_corners = 76'd11248294119274736131;
1393 12'd1381: quad_corners = 76'd11392127282081195013;
1394 12'd1382: quad_corners = 76'd11535819707399299080;
1395 12'd1383: quad_corners = 76'd85466488427287173643;
1396 12'd1384: quad_corners = 76'd233184133442013254670;
1397 12'd1385: quad_corners = 76'd307114802161901129234;
1398 12'd1386: quad_corners = 76'd381045470606642661397;
1399 12'd1387: quad_corners = 76'd528763115621368742425;
1400 12'd1388: quad_corners = 76'd602837899529064037405;
1401 12'd1389: quad_corners = 76'd750555544268644299809;
1402 12'd1390: quad_corners = 76'd898273329745712393765;
1403 12'd1391: quad_corners = 76'd1046135089947977459753;
1404 12'd1392: quad_corners = 76'd1193852734687557722158;
1405 12'd1393: quad_corners = 76'd1341714494614944881203;
1406 12'd1394: quad_corners = 76'd1489576254817210470968;
1407 12'd1395: quad_corners = 76'd1637438014744329194045;
1408 12'd1396: quad_corners = 76'd1785299915409204708418;
1409 12'd1397: quad_corners = 76'd1785299915409204708418;
1410 12'd1398: quad_corners = 76'd1785299915409204708418;
1411 12'd1399: quad_corners = 76'd1785299915409204708418;
1412 12'd1400: quad_corners = 76'd1785299915409204708418;
1413 12'd1401: quad_corners = 76'd1785299915409204708418;
1414 12'd1402: quad_corners = 76'd1785299915409204708418;
1415 12'd1403: quad_corners = 76'd1785299915409204708418;
1416 12'd1404: quad_corners = 76'd1785299915409204708418;
1417 12'd1405: quad_corners = 76'd1785299915409204708418;
1418 12'd1406: quad_corners = 76'd1785299915409204708418;
1419 12'd1407: quad_corners = 76'd1785299915409204708418;
1420 12'd1408: quad_corners = 76'd9522993250338384896;
1421 12'd1409: quad_corners = 76'd9522993250338384896;
1422 12'd1410: quad_corners = 76'd9522993250338384896;
1423 12'd1411: quad_corners = 76'd9522993250338384896;
1424 12'd1412: quad_corners = 76'd9522993250338384896;

1425 12'd1413: quad_corners = 76'd9522993250338384896;
1426 12'd1414: quad_corners = 76'd9522993250338384896;
1427 12'd1415: quad_corners = 76'd9522993250338384896;
1428 12'd1416: quad_corners = 76'd9522993250338384896;
1429 12'd1417: quad_corners = 76'd9522993250338384896;
1430 12'd1418: quad_corners = 76'd9522993250338384896;
1431 12'd1419: quad_corners = 76'd9522993250338384896;
1432 12'd1420: quad_corners = 76'd9522993250338384896;
1433 12'd1421: quad_corners = 76'd9522993250338384896;
1434 12'd1422: quad_corners = 76'd9522993250338384896;
1435 12'd1423: quad_corners = 76'd9522993250338384896;
1436 12'd1424: quad_corners = 76'd9522993250338384896;
1437 12'd1425: quad_corners = 76'd9522993250338384896;
1438 12'd1426: quad_corners = 76'd9522993250338384896;
1439 12'd1427: quad_corners = 76'd9522993250338384896;
1440 12'd1428: quad_corners = 76'd9522993250338384896;
1441 12'd1429: quad_corners = 76'd9522993250338384896;
1442 12'd1430: quad_corners = 76'd9522993250338384896;
1443 12'd1431: quad_corners = 76'd9522993250338384896;
1444 12'd1432: quad_corners = 76'd9522993250338384896;
1445 12'd1433: quad_corners = 76'd9522571037873317888;
1446 12'd1434: quad_corners = 76'd9522148825407726592;
1447 12'd1435: quad_corners = 76'd9521726612942659584;
1448 12'd1436: quad_corners = 76'd9521304400477068288;
1449 12'd1437: quad_corners = 76'd9520882188012001280;
1450 12'd1438: quad_corners = 76'd9664575163622790144;
1451 12'd1439: quad_corners = 76'd9664293688646078464;
1452 12'd1440: quad_corners = 76'd9807986664256343040;
1453 12'd1441: quad_corners = 76'd9951679639867131904;
1454 12'd1442: quad_corners = 76'd9951257427402064384;
1455 12'd1443: quad_corners = 76'd10094950403012853248;
1456 12'd1444: quad_corners = 76'd10238784115843561474;
1457 12'd1445: quad_corners = 76'd10382477090917479429;
1458 12'd1446: quad_corners = 76'd10526169791113489928;
1459 12'd1447: quad_corners = 76'd84601094436834011659;
1460 12'd1448: quad_corners = 76'd232318739451560092686;
1461 12'd1449: quad_corners = 76'd306249408171716402706;
1462 12'd1450: quad_corners = 76'd380324332817168488469;
1463 12'd1451: quad_corners = 76'd528186093019970949657;
1464 12'd1452: quad_corners = 76'd675903878771916950557;
1465 12'd1453: quad_corners = 76'd749978662679880680993;
1466 12'd1454: quad_corners = 76'd897840422607536275494;
1467 12'd1455: quad_corners = 76'd1045702323547558656554;
1468 12'd1456: quad_corners = 76'd1193564083750092157999;
1469 12'd1457: quad_corners = 76'd1341425984415236107828;
1470 12'd1458: quad_corners = 76'd1489287744617501697593;

1471 12'd1459: quad_corners = 76'd1711080736765291274302;
1472 12'd1460: quad_corners = 76'd1858942637705045219396;
1473 12'd1461: quad_corners = 76'd1858942637705045219396;
1474 12'd1462: quad_corners = 76'd1858942637705045219396;
1475 12'd1463: quad_corners = 76'd1858942637705045219396;
1476 12'd1464: quad_corners = 76'd1858942637705045219396;
1477 12'd1465: quad_corners = 76'd1858942637705045219396;
1478 12'd1466: quad_corners = 76'd1858942637705045219396;
1479 12'd1467: quad_corners = 76'd1858942637705045219396;
1480 12'd1468: quad_corners = 76'd1858942637705045219396;
1481 12'd1469: quad_corners = 76'd1858942637705045219396;
1482 12'd1470: quad_corners = 76'd1858942637705045219396;
1483 12'd1471: quad_corners = 76'd1858942637705045219396;
1484 12'd1472: quad_corners = 76'd8080715469677704704;
1485 12'd1473: quad_corners = 76'd8080715469677704704;
1486 12'd1474: quad_corners = 76'd8080715469677704704;
1487 12'd1475: quad_corners = 76'd8080715469677704704;
1488 12'd1476: quad_corners = 76'd8080715469677704704;
1489 12'd1477: quad_corners = 76'd8080715469677704704;
1490 12'd1478: quad_corners = 76'd8080715469677704704;
1491 12'd1479: quad_corners = 76'd8080715469677704704;
1492 12'd1480: quad_corners = 76'd8080715469677704704;
1493 12'd1481: quad_corners = 76'd8080715469677704704;
1494 12'd1482: quad_corners = 76'd8080715469677704704;
1495 12'd1483: quad_corners = 76'd8080715469677704704;
1496 12'd1484: quad_corners = 76'd8080715469677704704;
1497 12'd1485: quad_corners = 76'd8080715469677704704;
1498 12'd1486: quad_corners = 76'd8080715469677704704;
1499 12'd1487: quad_corners = 76'd8080715469677704704;
1500 12'd1488: quad_corners = 76'd8080715469677704704;
1501 12'd1489: quad_corners = 76'd8080715469677704704;
1502 12'd1490: quad_corners = 76'd8080715469677704704;
1503 12'd1491: quad_corners = 76'd8080715469677704704;
1504 12'd1492: quad_corners = 76'd8080715469677704704;
1505 12'd1493: quad_corners = 76'd8080715469677704704;
1506 12'd1494: quad_corners = 76'd8080715469677704704;
1507 12'd1495: quad_corners = 76'd8080715469677704704;
1508 12'd1496: quad_corners = 76'd8080715469677704704;
1509 12'd1497: quad_corners = 76'd8080433994700993536;
1510 12'd1498: quad_corners = 76'd8080011782235402240;
1511 12'd1499: quad_corners = 76'd8223704757846191104;
1512 12'd1500: quad_corners = 76'd8223282545380599808;
1513 12'd1501: quad_corners = 76'd8367116258479744000;
1514 12'd1502: quad_corners = 76'd8366694046014676992;
1515 12'd1503: quad_corners = 76'd8510387021625465856;
1516 12'd1504: quad_corners = 76'd8654220734724085760;

1517 12'd1505: quad_corners = 76'd8797913710334874624;
1518 12'd1506: quad_corners = 76'd8941606685945663488;
1519 12'd1507: quad_corners = 76'd9085440399044807680;
1520 12'd1508: quad_corners = 76'd9229133374655596034;
1521 12'd1509: quad_corners = 76'd9517082275562160645;
1522 12'd1510: quad_corners = 76'd9660775250904514056;
1523 12'd1511: quad_corners = 76'd157522676466341148683;
1524 12'd1512: quad_corners = 76'd231453345461106930190;
1525 12'd1513: quad_corners = 76'd305528270106827451922;
1526 12'd1514: quad_corners = 76'd453390171047386179606;
1527 12'd1515: quad_corners = 76'd527464954955350434841;
1528 12'd1516: quad_corners = 76'd675326856170518634014;
1529 12'd1517: quad_corners = 76'd823188757110808926242;
1530 12'd1518: quad_corners = 76'd971050517313610863142;
1531 12'd1519: quad_corners = 76'd1119056533441709100075;
1532 12'd1520: quad_corners = 76'd1266918434381999392304;
1533 12'd1521: quad_corners = 76'd1414924450510097104949;
1534 12'd1522: quad_corners = 76'd1562930466363317434938;
1535 12'd1523: quad_corners = 76'd1784579343598177498176;
1536 12'd1524: quad_corners = 76'd1932585359726007299653;
1537 12'd1525: quad_corners = 76'd1932585359726007299653;
1538 12'd1526: quad_corners = 76'd1932585359726007299653;
1539 12'd1527: quad_corners = 76'd1932585359726007299653;
1540 12'd1528: quad_corners = 76'd1932585359726007299653;
1541 12'd1529: quad_corners = 76'd1932585359726007299653;
1542 12'd1530: quad_corners = 76'd1932585359726007299653;
1543 12'd1531: quad_corners = 76'd1932585359726007299653;
1544 12'd1532: quad_corners = 76'd1932585359726007299653;
1545 12'd1533: quad_corners = 76'd1932585359726007299653;
1546 12'd1534: quad_corners = 76'd1932585359726007299653;
1547 12'd1535: quad_corners = 76'd1932585359726007299653;
1548 12'd1536: quad_corners = 76'd6782834352069067264;
1549 12'd1537: quad_corners = 76'd6782834352069067264;
1550 12'd1538: quad_corners = 76'd6782834352069067264;
1551 12'd1539: quad_corners = 76'd6782834352069067264;
1552 12'd1540: quad_corners = 76'd6782834352069067264;
1553 12'd1541: quad_corners = 76'd6782834352069067264;
1554 12'd1542: quad_corners = 76'd6782834352069067264;
1555 12'd1543: quad_corners = 76'd6782834352069067264;
1556 12'd1544: quad_corners = 76'd6782834352069067264;
1557 12'd1545: quad_corners = 76'd6782834352069067264;
1558 12'd1546: quad_corners = 76'd6782834352069067264;
1559 12'd1547: quad_corners = 76'd6782834352069067264;
1560 12'd1548: quad_corners = 76'd6782834352069067264;
1561 12'd1549: quad_corners = 76'd6782834352069067264;
1562 12'd1550: quad_corners = 76'd6782834352069067264;

1563 12'd1551: quad_corners = 76'd6782834352069067264;
1564 12'd1552: quad_corners = 76'd6782834352069067264;
1565 12'd1553: quad_corners = 76'd6782834352069067264;
1566 12'd1554: quad_corners = 76'd6782834352069067264;
1567 12'd1555: quad_corners = 76'd6782834352069067264;
1568 12'd1556: quad_corners = 76'd6782834352069067264;
1569 12'd1557: quad_corners = 76'd6782834352069067264;
1570 12'd1558: quad_corners = 76'd6782834352069067264;
1571 12'd1559: quad_corners = 76'd6782834352069067264;
1572 12'd1560: quad_corners = 76'd6782834352069067264;
1573 12'd1561: quad_corners = 76'd6926527327679856128;
1574 12'd1562: quad_corners = 76'd6926105115214789120;
1575 12'd1563: quad_corners = 76'd6925682902749198336;
1576 12'd1564: quad_corners = 76'd7069516615848342528;
1577 12'd1565: quad_corners = 76'd7213209591459131392;
1578 12'd1566: quad_corners = 76'd7212928116481895424;
1579 12'd1567: quad_corners = 76'd7356621092092684288;
1580 12'd1568: quad_corners = 76'd7500454805191828480;
1581 12'd1569: quad_corners = 76'd7788262968878473216;
1582 12'd1570: quad_corners = 76'd7932096681977617408;
1583 12'd1571: quad_corners = 76'd8075789657588406272;
1584 12'd1572: quad_corners = 76'd8363738558763406338;
1585 12'd1573: quad_corners = 76'd8507572271862550533;
1586 12'd1574: quad_corners = 76'd82582497467607321096;
1587 12'd1575: quad_corners = 76'd156657281925863736843;
1588 12'd1576: quad_corners = 76'd230732207121340072463;
1589 12'd1577: quad_corners = 76'd378594108061898800146;
1590 12'd1578: quad_corners = 76'd452669032982497228822;
1591 12'd1579: quad_corners = 76'd600675049111131812378;
1592 12'd1580: quad_corners = 76'd748536950326300012062;
1593 12'd1581: quad_corners = 76'd896398851266859264035;
1594 12'd1582: quad_corners = 76'd1044404867395225412135;
1595 12'd1583: quad_corners = 76'd1192410883798469467692;
1596 12'd1584: quad_corners = 76'd1340416899926835615793;
1597 12'd1585: quad_corners = 76'd1488422916054933852726;
1598 12'd1586: quad_corners = 76'd1636428932183300000827;
1599 12'd1587: quad_corners = 76'd1858221924881114351169;
1600 12'd1588: quad_corners = 76'd2006228081746700419143;
1601 12'd1589: quad_corners = 76'd2006228081746700419143;
1602 12'd1590: quad_corners = 76'd2006228081746700419143;
1603 12'd1591: quad_corners = 76'd2006228081746700419143;
1604 12'd1592: quad_corners = 76'd2006228081746700419143;
1605 12'd1593: quad_corners = 76'd2006228081746700419143;
1606 12'd1594: quad_corners = 76'd2006228081746700419143;
1607 12'd1595: quad_corners = 76'd2006228081746700419143;
1608 12'd1596: quad_corners = 76'd2006228081746700419143;

1609 12'd1597: quad_corners = 76'd2006228081746700419143;
1610 12'd1598: quad_corners = 76'd2006228081746700419143;
1611 12'd1599: quad_corners = 76'd2006228081746700419143;
1612 12'd1600: quad_corners = 76'd5628927685047929344;
1613 12'd1601: quad_corners = 76'd5628927685047929344;
1614 12'd1602: quad_corners = 76'd5628927685047929344;
1615 12'd1603: quad_corners = 76'd5628927685047929344;
1616 12'd1604: quad_corners = 76'd5628927685047929344;
1617 12'd1605: quad_corners = 76'd5628927685047929344;
1618 12'd1606: quad_corners = 76'd5628927685047929344;
1619 12'd1607: quad_corners = 76'd5628927685047929344;
1620 12'd1608: quad_corners = 76'd5628927685047929344;
1621 12'd1609: quad_corners = 76'd5628927685047929344;
1622 12'd1610: quad_corners = 76'd5628927685047929344;
1623 12'd1611: quad_corners = 76'd5628927685047929344;
1624 12'd1612: quad_corners = 76'd5628927685047929344;
1625 12'd1613: quad_corners = 76'd5628927685047929344;
1626 12'd1614: quad_corners = 76'd5628927685047929344;
1627 12'd1615: quad_corners = 76'd5628927685047929344;
1628 12'd1616: quad_corners = 76'd5628927685047929344;
1629 12'd1617: quad_corners = 76'd5628927685047929344;
1630 12'd1618: quad_corners = 76'd5628927685047929344;
1631 12'd1619: quad_corners = 76'd5628927685047929344;
1632 12'd1620: quad_corners = 76'd5628927685047929344;
1633 12'd1621: quad_corners = 76'd5628927685047929344;
1634 12'd1622: quad_corners = 76'd5628927685047929344;
1635 12'd1623: quad_corners = 76'd5628927685047929344;
1636 12'd1624: quad_corners = 76'd5628927685047929344;
1637 12'd1625: quad_corners = 76'd5628505472582862848;
1638 12'd1626: quad_corners = 76'd5772339185681482752;
1639 12'd1627: quad_corners = 76'd5771916973216416256;
1640 12'd1628: quad_corners = 76'd5915750686315560448;
1641 12'd1629: quad_corners = 76'd6059443661926349312;
1642 12'd1630: quad_corners = 76'd6203277375024969728;
1643 12'd1631: quad_corners = 76'd6346970350635758592;
1644 12'd1632: quad_corners = 76'd6490804063734902784;
1645 12'd1633: quad_corners = 76'd6778752964909902848;
1646 12'd1634: quad_corners = 76'd6922586678009047040;
1647 12'd1635: quad_corners = 76'd7210394841695691776;
1648 12'd1636: quad_corners = 76'd7498343742870691842;
1649 12'd1637: quad_corners = 76'd7642177455969836037;
1650 12'd1638: quad_corners = 76'd155504078946821249033;
1651 12'd1639: quad_corners = 76'd229723119330910311436;
1652 12'd1640: quad_corners = 76'd303798044526386647055;
1653 12'd1641: quad_corners = 76'd451659946016969624083;
1654 12'd1642: quad_corners = 76'd525878986125643908631;

1655 12'd1643: quad_corners = 76'd673740887341080543771;
1656 12'd1644: quad_corners = 76'd821747044207203482655;
1657 12'd1645: quad_corners = 76'd969753060610716497956;
1658 12'd1646: quad_corners = 76'd1117759077013960553000;
1659 12'd1647: quad_corners = 76'd1265765233880083492397;
1660 12'd1648: quad_corners = 76'd1413771250283328071730;
1661 12'd1649: quad_corners = 76'd1561921381599770076216;
1662 12'd1650: quad_corners = 76'd1783714515035340693053;
1663 12'd1651: quad_corners = 76'd1931864646351783221315;
1664 12'd1652: quad_corners = 76'd2153657779787085402696;
1665 12'd1653: quad_corners = 76'd2153657779787085402696;
1666 12'd1654: quad_corners = 76'd2153657779787085402696;
1667 12'd1655: quad_corners = 76'd2153657779787085402696;
1668 12'd1656: quad_corners = 76'd2153657779787085402696;
1669 12'd1657: quad_corners = 76'd2153657779787085402696;
1670 12'd1658: quad_corners = 76'd2153657779787085402696;
1671 12'd1659: quad_corners = 76'd2153657779787085402696;
1672 12'd1660: quad_corners = 76'd2153657779787085402696;
1673 12'd1661: quad_corners = 76'd2153657779787085402696;
1674 12'd1662: quad_corners = 76'd2153657779787085402696;
1675 12'd1663: quad_corners = 76'd2153657779787085402696;
1676 12'd1664: quad_corners = 76'd4475161754978276352;
1677 12'd1665: quad_corners = 76'd4475161754978276352;
1678 12'd1666: quad_corners = 76'd4475161754978276352;
1679 12'd1667: quad_corners = 76'd4475161754978276352;
1680 12'd1668: quad_corners = 76'd4475161754978276352;
1681 12'd1669: quad_corners = 76'd4475161754978276352;
1682 12'd1670: quad_corners = 76'd4475161754978276352;
1683 12'd1671: quad_corners = 76'd4475161754978276352;
1684 12'd1672: quad_corners = 76'd4475161754978276352;
1685 12'd1673: quad_corners = 76'd4475161754978276352;
1686 12'd1674: quad_corners = 76'd4475161754978276352;
1687 12'd1675: quad_corners = 76'd4475161754978276352;
1688 12'd1676: quad_corners = 76'd4475161754978276352;
1689 12'd1677: quad_corners = 76'd4475161754978276352;
1690 12'd1678: quad_corners = 76'd4475161754978276352;
1691 12'd1679: quad_corners = 76'd4475161754978276352;
1692 12'd1680: quad_corners = 76'd4475161754978276352;
1693 12'd1681: quad_corners = 76'd4475161754978276352;
1694 12'd1682: quad_corners = 76'd4475161754978276352;
1695 12'd1683: quad_corners = 76'd4475161754978276352;
1696 12'd1684: quad_corners = 76'd4475161754978276352;
1697 12'd1685: quad_corners = 76'd4475161754978276352;
1698 12'd1686: quad_corners = 76'd4475161754978276352;
1699 12'd1687: quad_corners = 76'd4475161754978276352;
1700 12'd1688: quad_corners = 76'd4475161754978276352;

1701 12'd1689: quad_corners = 76'd4618854731125412352;
1702 12'd1690: quad_corners = 76'd4618573256148700672;
1703 12'd1691: quad_corners = 76'd4762266231759490048;
1704 12'd1692: quad_corners = 76'd4906099944858109952;
1705 12'd1693: quad_corners = 76'd5049933657957254144;
1706 12'd1694: quad_corners = 76'd5193626633568043520;
1707 12'd1695: quad_corners = 76'd5337460346667187712;
1708 12'd1696: quad_corners = 76'd5625409247841663488;
1709 12'd1697: quad_corners = 76'd5769242960940808192;
1710 12'd1698: quad_corners = 76'd6057191862115808256;
1711 12'd1699: quad_corners = 76'd6345140763290808320;
1712 12'd1700: quad_corners = 76'd6633089664465808387;
1713 12'd1701: quad_corners = 76'd80708014860479014918;
1714 12'd1702: quad_corners = 76'd154782940056492221449;
1715 12'd1703: quad_corners = 76'd302644841547343634445;
1716 12'd1704: quad_corners = 76'd376863881931432696848;
1717 12'd1705: quad_corners = 76'd524725924159504029716;
1718 12'd1706: quad_corners = 76'd598944964543056221208;
1719 12'd1707: quad_corners = 76'd746950980946837671964;
1720 12'd1708: quad_corners = 76'd894957138087838517792;
1721 12'd1709: quad_corners = 76'd1042963154491351008805;
1722 12'd1710: quad_corners = 76'd1191113426820427711018;
1723 12'd1711: quad_corners = 76'd1339119443223940725807;
1724 12'd1712: quad_corners = 76'd1487269715553017428020;
1725 12'd1713: quad_corners = 76'd1709062848988588045369;
1726 12'd1714: quad_corners = 76'd1857213121317665271358;
1727 12'd1715: quad_corners = 76'd2005363252634107275844;
1728 12'd1716: quad_corners = 76'd2227300501257754272842;
1729 12'd1717: quad_corners = 76'd2227300501257754272842;
1730 12'd1718: quad_corners = 76'd2227300501257754272842;
1731 12'd1719: quad_corners = 76'd2227300501257754272842;
1732 12'd1720: quad_corners = 76'd2227300501257754272842;
1733 12'd1721: quad_corners = 76'd2227300501257754272842;
1734 12'd1722: quad_corners = 76'd2227300501257754272842;
1735 12'd1723: quad_corners = 76'd2227300501257754272842;
1736 12'd1724: quad_corners = 76'd2227300501257754272842;
1737 12'd1725: quad_corners = 76'd2227300501257754272842;
1738 12'd1726: quad_corners = 76'd2227300501257754272842;
1739 12'd1727: quad_corners = 76'd2227300501257754272842;
1740 12'd1728: quad_corners = 76'd3465511012983430144;
1741 12'd1729: quad_corners = 76'd3465511012983430144;
1742 12'd1730: quad_corners = 76'd3465511012983430144;
1743 12'd1731: quad_corners = 76'd3465511012983430144;
1744 12'd1732: quad_corners = 76'd3465511012983430144;
1745 12'd1733: quad_corners = 76'd3465511012983430144;
1746 12'd1734: quad_corners = 76'd3465511012983430144;

1747 12'd1735: quad_corners = 76'd3465511012983430144;
1748 12'd1736: quad_corners = 76'd3465511012983430144;
1749 12'd1737: quad_corners = 76'd3465511012983430144;
1750 12'd1738: quad_corners = 76'd3465511012983430144;
1751 12'd1739: quad_corners = 76'd3465511012983430144;
1752 12'd1740: quad_corners = 76'd3465511012983430144;
1753 12'd1741: quad_corners = 76'd3465511012983430144;
1754 12'd1742: quad_corners = 76'd3465511012983430144;
1755 12'd1743: quad_corners = 76'd3465511012983430144;
1756 12'd1744: quad_corners = 76'd3465511012983430144;
1757 12'd1745: quad_corners = 76'd3465511012983430144;
1758 12'd1746: quad_corners = 76'd3465511012983430144;
1759 12'd1747: quad_corners = 76'd3465511012983430144;
1760 12'd1748: quad_corners = 76'd3465511012983430144;
1761 12'd1749: quad_corners = 76'd3465511012983430144;
1762 12'd1750: quad_corners = 76'd3465511012983430144;
1763 12'd1751: quad_corners = 76'd3465511012983430144;
1764 12'd1752: quad_corners = 76'd3465511012983430144;
1765 12'd1753: quad_corners = 76'd3609344726351010304;
1766 12'd1754: quad_corners = 76'd3608922514154379264;
1767 12'd1755: quad_corners = 76'd3752756227789870080;
1768 12'd1756: quad_corners = 76'd3896589940889014784;
1769 12'd1757: quad_corners = 76'd4040282916499803648;
1770 12'd1758: quad_corners = 76'd4328231817674804224;
1771 12'd1759: quad_corners = 76'd4472065530773424128;
1772 12'd1760: quad_corners = 76'd4760014431948424192;
1773 12'd1761: quad_corners = 76'd4903848145047568896;
1774 12'd1762: quad_corners = 76'd5191797046222568960;
1775 12'd1763: quad_corners = 76'd5479745947397569026;
1776 12'd1764: quad_corners = 76'd79554671143410776068;
1777 12'd1765: quad_corners = 76'd227416713371750544391;
1778 12'd1766: quad_corners = 76'd301635753755839606795;
1779 12'd1767: quad_corners = 76'd375710678951852813326;
1780 12'd1768: quad_corners = 76'd449929860073430231058;
1781 12'd1769: quad_corners = 76'd597935876752357499925;
1782 12'd1770: quad_corners = 76'd745942034168505212953;
1783 12'd1771: quad_corners = 76'd820161074552325839901;
1784 12'd1772: quad_corners = 76'd968167231968473028642;
1785 12'd1773: quad_corners = 76'd1116317504297549730854;
1786 12'd1774: quad_corners = 76'd1264323661438819536939;
1787 12'd1775: quad_corners = 76'd1412473933767896239152;
1788 12'd1776: quad_corners = 76'd1634411182391811147317;
1789 12'd1777: quad_corners = 76'd1782561454995766280763;
1790 12'd1778: quad_corners = 76'd1930711727324842982976;
1791 12'd1779: quad_corners = 76'd2152648975948489455686;
1792 12'd1780: quad_corners = 76'd2374730339760480744524;

1793 12'd1781: quad_corners = 76'd2374730339760480744524;
1794 12'd1782: quad_corners = 76'd2374730339760480744524;
1795 12'd1783: quad_corners = 76'd2374730339760480744524;
1796 12'd1784: quad_corners = 76'd2374730339760480744524;
1797 12'd1785: quad_corners = 76'd2374730339760480744524;
1798 12'd1786: quad_corners = 76'd2374730339760480744524;
1799 12'd1787: quad_corners = 76'd2374730339760480744524;
1800 12'd1788: quad_corners = 76'd2374730339760480744524;
1801 12'd1789: quad_corners = 76'd2374730339760480744524;
1802 12'd1790: quad_corners = 76'd2374730339760480744524;
1803 12'd1791: quad_corners = 76'd2374730339760480744524;
1804 12'd1792: quad_corners = 76'd2600116196284883968;
1805 12'd1793: quad_corners = 76'd2600116196284883968;
1806 12'd1794: quad_corners = 76'd2600116196284883968;
1807 12'd1795: quad_corners = 76'd2600116196284883968;
1808 12'd1796: quad_corners = 76'd2600116196284883968;
1809 12'd1797: quad_corners = 76'd2600116196284883968;
1810 12'd1798: quad_corners = 76'd2600116196284883968;
1811 12'd1799: quad_corners = 76'd2600116196284883968;
1812 12'd1800: quad_corners = 76'd2600116196284883968;
1813 12'd1801: quad_corners = 76'd2600116196284883968;
1814 12'd1802: quad_corners = 76'd2600116196284883968;
1815 12'd1803: quad_corners = 76'd2600116196284883968;
1816 12'd1804: quad_corners = 76'd2600116196284883968;
1817 12'd1805: quad_corners = 76'd2600116196284883968;
1818 12'd1806: quad_corners = 76'd2600116196284883968;
1819 12'd1807: quad_corners = 76'd2600116196284883968;
1820 12'd1808: quad_corners = 76'd2600116196284883968;
1821 12'd1809: quad_corners = 76'd2600116196284883968;
1822 12'd1810: quad_corners = 76'd2600116196284883968;
1823 12'd1811: quad_corners = 76'd2600116196284883968;
1824 12'd1812: quad_corners = 76'd2600116196284883968;
1825 12'd1813: quad_corners = 76'd2600116196284883968;
1826 12'd1814: quad_corners = 76'd2600116196284883968;
1827 12'd1815: quad_corners = 76'd2600116196284883968;
1828 12'd1816: quad_corners = 76'd2600116196284883968;
1829 12'd1817: quad_corners = 76'd2599693984356164096;
1830 12'd1818: quad_corners = 76'd2743527697723744256;
1831 12'd1819: quad_corners = 76'd2887361411091323904;
1832 12'd1820: quad_corners = 76'd3031195124726815232;
1833 12'd1821: quad_corners = 76'd3174888100606040064;
1834 12'd1822: quad_corners = 76'd3462837001781040128;
1835 12'd1823: quad_corners = 76'd3606670714880184832;
1836 12'd1824: quad_corners = 76'd3894619616055184896;
1837 12'd1825: quad_corners = 76'd4182709254718540800;
1838 12'd1826: quad_corners = 76'd78257634450731747328;

1839 12'd1827: quad_corners = 76'd152332559646744953859;
1840 12'd1828: quad_corners = 76'd226407484842758160902;
1841 12'd1829: quad_corners = 76'd300626665964335578633;
1842 12'd1830: quad_corners = 76'd374701591160348785164;
1843 12'd1831: quad_corners = 76'd522707748576764409872;
1844 12'd1832: quad_corners = 76'd596926788960853472275;
1845 12'd1833: quad_corners = 76'd744932946377269096471;
1846 12'd1834: quad_corners = 76'd819152127498846514203;
1847 12'd1835: quad_corners = 76'd967158284915262138399;
1848 12'd1836: quad_corners = 76'd1115308557519485707299;
1849 12'd1837: quad_corners = 76'd1263314714935632896040;
1850 12'd1838: quad_corners = 76'd1411464987539587505197;
1851 12'd1839: quad_corners = 76'd1559615259868932642866;
1852 12'd1840: quad_corners = 76'd1707765532198009869367;
1853 12'd1841: quad_corners = 76'd1929702781097071120444;
1854 12'd1842: quad_corners = 76'd2077997309351712558146;
1855 12'd1843: quad_corners = 76'd2299934557975627466824;
1856 12'd1844: quad_corners = 76'd2448229086505146811469;
1857 12'd1845: quad_corners = 76'd2448229086505146811469;
1858 12'd1846: quad_corners = 76'd2448229086505146811469;
1859 12'd1847: quad_corners = 76'd2448229086505146811469;
1860 12'd1848: quad_corners = 76'd2448229086505146811469;
1861 12'd1849: quad_corners = 76'd2448229086505146811469;
1862 12'd1850: quad_corners = 76'd2448229086505146811469;
1863 12'd1851: quad_corners = 76'd2448229086505146811469;
1864 12'd1852: quad_corners = 76'd2448229086505146811469;
1865 12'd1853: quad_corners = 76'd2448229086505146811469;
1866 12'd1854: quad_corners = 76'd2448229086505146811469;
1867 12'd1855: quad_corners = 76'd2448229086505146811469;
1868 12'd1856: quad_corners = 76'd1734721379585289216;
1869 12'd1857: quad_corners = 76'd1734721379585289216;
1870 12'd1858: quad_corners = 76'd1734721379585289216;
1871 12'd1859: quad_corners = 76'd1734721379585289216;
1872 12'd1860: quad_corners = 76'd1734721379585289216;
1873 12'd1861: quad_corners = 76'd1734721379585289216;
1874 12'd1862: quad_corners = 76'd1734721379585289216;
1875 12'd1863: quad_corners = 76'd1734721379585289216;
1876 12'd1864: quad_corners = 76'd1734721379585289216;
1877 12'd1865: quad_corners = 76'd1734721379585289216;
1878 12'd1866: quad_corners = 76'd1734721379585289216;
1879 12'd1867: quad_corners = 76'd1734721379585289216;
1880 12'd1868: quad_corners = 76'd1734721379585289216;
1881 12'd1869: quad_corners = 76'd1734721379585289216;
1882 12'd1870: quad_corners = 76'd1734721379585289216;
1883 12'd1871: quad_corners = 76'd1734721379585289216;
1884 12'd1872: quad_corners = 76'd1734721379585289216;

1885 12'd1873: quad_corners = 76'd1734721379585289216;
1886 12'd1874: quad_corners = 76'd1734721379585289216;
1887 12'd1875: quad_corners = 76'd1734721379585289216;
1888 12'd1876: quad_corners = 76'd1734721379585289216;
1889 12'd1877: quad_corners = 76'd1734721379585289216;
1890 12'd1878: quad_corners = 76'd1734721379585289216;
1891 12'd1879: quad_corners = 76'd1734721379585289216;
1892 12'd1880: quad_corners = 76'd1734721379585289216;
1893 12'd1881: quad_corners = 76'd1734299167657093632;
1894 12'd1882: quad_corners = 76'd1878132881024673280;
1895 12'd1883: quad_corners = 76'd2021966594660164608;
1896 12'd1884: quad_corners = 76'd2165800308027744768;
1897 12'd1885: quad_corners = 76'd2453749209471180800;
1898 12'd1886: quad_corners = 76'd2597582923107195904;
1899 12'd1887: quad_corners = 76'd2885531824550631936;
1900 12'd1888: quad_corners = 76'd76960457020563838976;
1901 12'd1889: quad_corners = 76'd151035382216577045504;
1902 12'd1890: quad_corners = 76'd225110448150078607874;
1903 12'd1891: quad_corners = 76'd299185373346091290117;
1904 12'd1892: quad_corners = 76'd373260439279593376776;
1905 12'd1893: quad_corners = 76'd447479479663682439179;
1906 12'd1894: quad_corners = 76'd521698660785259857422;
1907 12'd1895: quad_corners = 76'd595773726718761419281;
1908 12'd1896: quad_corners = 76'd743779884135177043477;
1909 12'd1897: quad_corners = 76'd818143039707341962265;
1910 12'd1898: quad_corners = 76'd966149197123757586461;
1911 12'd1899: quad_corners = 76'd1114299469728249066529;
1912 12'd1900: quad_corners = 76'd1188518791587315363877;
1913 12'd1901: quad_corners = 76'd1336669064191806843946;
1914 12'd1902: quad_corners = 76'd1484819336796029889071;
1915 12'd1903: quad_corners = 76'd1706756585694823228980;
1916 12'd1904: quad_corners = 76'd1854906999036534628921;
1917 12'd1905: quad_corners = 76'd2003201386553955622462;
1918 12'd1906: quad_corners = 76'd2225138776190237317700;
1919 12'd1907: quad_corners = 76'd2373433304445146666569;
1920 12'd1908: quad_corners = 76'd2595514668532015862351;
1921 12'd1909: quad_corners = 76'd2595514668532015862351;
1922 12'd1910: quad_corners = 76'd2595514668532015862351;
1923 12'd1911: quad_corners = 76'd2595514668532015862351;
1924 12'd1912: quad_corners = 76'd2595514668532015862351;
1925 12'd1913: quad_corners = 76'd2595514668532015862351;
1926 12'd1914: quad_corners = 76'd2595514668532015862351;
1927 12'd1915: quad_corners = 76'd2595514668532015862351;
1928 12'd1916: quad_corners = 76'd2595514668532015862351;
1929 12'd1917: quad_corners = 76'd2595514668532015862351;
1930 12'd1918: quad_corners = 76'd2595514668532015862351;

1931 12'd1919: quad_corners = 76'd2595514668532015862351;
1932 12'd1920: quad_corners = 76'd74656302857723900416;
1933 12'd1921: quad_corners = 76'd74656302857723900416;
1934 12'd1922: quad_corners = 76'd74656302857723900416;
1935 12'd1923: quad_corners = 76'd74656302857723900416;
1936 12'd1924: quad_corners = 76'd74656302857723900416;
1937 12'd1925: quad_corners = 76'd74656302857723900416;
1938 12'd1926: quad_corners = 76'd74656302857723900416;
1939 12'd1927: quad_corners = 76'd74656302857723900416;
1940 12'd1928: quad_corners = 76'd74656302857723900416;
1941 12'd1929: quad_corners = 76'd74656302857723900416;
1942 12'd1930: quad_corners = 76'd74656302857723900416;
1943 12'd1931: quad_corners = 76'd74656302857723900416;
1944 12'd1932: quad_corners = 76'd74656302857723900416;
1945 12'd1933: quad_corners = 76'd74656302857723900416;
1946 12'd1934: quad_corners = 76'd74656302857723900416;
1947 12'd1935: quad_corners = 76'd74656302857723900416;
1948 12'd1936: quad_corners = 76'd74656302857723900416;
1949 12'd1937: quad_corners = 76'd74656302857723900416;
1950 12'd1938: quad_corners = 76'd74656302857723900416;
1951 12'd1939: quad_corners = 76'd74656302857723900416;
1952 12'd1940: quad_corners = 76'd74656302857723900416;
1953 12'd1941: quad_corners = 76'd74656302857723900416;
1954 12'd1942: quad_corners = 76'd74656302857723900416;
1955 12'd1943: quad_corners = 76'd74656302857723900416;
1956 12'd1944: quad_corners = 76'd74656302857723900416;
1957 12'd1945: quad_corners = 76'd74800136571359916032;
1958 12'd1946: quad_corners = 76'd74943829547507576320;
1959 12'd1947: quad_corners = 76'd75087663260875156480;
1960 12'd1948: quad_corners = 76'd75231496974510647296;
1961 12'd1949: quad_corners = 76'd149306422170792289792;
1962 12'd1950: quad_corners = 76'd149450396621648225280;
1963 12'd1951: quad_corners = 76'd223525321818198303232;
1964 12'd1952: quad_corners = 76'd223813270719641739264;
1965 12'd1953: quad_corners = 76'd297888336653143301122;
1966 12'd1954: quad_corners = 76'd371963261849424943620;
1967 12'd1955: quad_corners = 76'd446038327782926505991;
1968 12'd1956: quad_corners = 76'd520257368167015568394;
1969 12'd1957: quad_corners = 76'd594332434100517130765;
1970 12'd1958: quad_corners = 76'd668551615222094548496;
1971 12'd1959: quad_corners = 76'd742770796343671966739;
1972 12'd1960: quad_corners = 76'd890776953760087590935;
1973 12'd1961: quad_corners = 76'd965140250069741389339;
1974 12'd1962: quad_corners = 76'd1113146407486157013535;
1975 12'd1963: quad_corners = 76'd1261296680090648493603;
1976 12'd1964: quad_corners = 76'd1335660117137790123047;

```

1977      12'd1965: quad_corners = 76'd1483810389742282127404;
1978      12'd1966: quad_corners = 76'd1631960662346773607473;
1979      12'd1967: quad_corners = 76'd1853898051983323214389;
1980      12'd1968: quad_corners = 76'd2002192580513110994491;
1981      12'd1969: quad_corners = 76'd2150342853117334039104;
1982      12'd1970: quad_corners = 76'd2372424357941960025669;
1983      12'd1971: quad_corners = 76'd2520718886471747281483;
1984      12'd1972: quad_corners = 76'd2742800391296373268561;
1985      12'd1973: quad_corners = 76'd2742800391296373268561;
1986      12'd1974: quad_corners = 76'd2742800391296373268561;
1987      12'd1975: quad_corners = 76'd2742800391296373268561;
1988      12'd1976: quad_corners = 76'd2742800391296373268561;
1989      12'd1977: quad_corners = 76'd2742800391296373268561;
1990      12'd1978: quad_corners = 76'd2742800391296373268561;
1991      12'd1979: quad_corners = 76'd2742800391296373268561;
1992      12'd1980: quad_corners = 76'd2742800391296373268561;
1993      12'd1981: quad_corners = 76'd2742800391296373268561;
1994      12'd1982: quad_corners = 76'd2742800391296373268561;
1995      12'd1983: quad_corners = 76'd2742800391296373268561;
1996      12'd1984: quad_corners = 76'd221508975818776573952;
1997      12'd1985: quad_corners = 76'd221508975818776573952;
1998      12'd1986: quad_corners = 76'd221508975818776573952;
1999      12'd1987: quad_corners = 76'd221508975818776573952;
2000      12'd1988: quad_corners = 76'd221508975818776573952;
2001      12'd1989: quad_corners = 76'd221508975818776573952;
2002      12'd1990: quad_corners = 76'd221508975818776573952;
2003      12'd1991: quad_corners = 76'd221508975818776573952;
2004      12'd1992: quad_corners = 76'd221508975818776573952;
2005      12'd1993: quad_corners = 76'd221508975818776573952;
2006      12'd1994: quad_corners = 76'd221508975818776573952;
2007      12'd1995: quad_corners = 76'd221508975818776573952;
2008      12'd1996: quad_corners = 76'd221508975818776573952;
2009      12'd1997: quad_corners = 76'd221508975818776573952;
2010      12'd1998: quad_corners = 76'd221508975818776573952;
2011      12'd1999: quad_corners = 76'd221508975818776573952;
2012      12'd2000: quad_corners = 76'd221508975818776573952;
2013      12'd2001: quad_corners = 76'd221508975818776573952;
2014      12'd2002: quad_corners = 76'd221508975818776573952;
2015      12'd2003: quad_corners = 76'd221508975818776573952;
2016      12'd2004: quad_corners = 76'd221508975818776573952;
2017      12'd2005: quad_corners = 76'd221508975818776573952;
2018      12'd2006: quad_corners = 76'd221508975818776573952;
2019      12'd2007: quad_corners = 76'd221508975818776573952;
2020      12'd2008: quad_corners = 76'd221508975818776573952;
2021      12'd2009: quad_corners = 76'd221652809532412589568;
2022      12'd2010: quad_corners = 76'd221796643246048605184;

```

2023 12'd2011: quad_corners = 76'd221940476959415661056;
 2024 12'd2012: quad_corners = 76'd295871286967889883136;
 2025 12'd2013: quad_corners = 76'd296159235869601754624;
 2026 12'd2014: quad_corners = 76'd296447325508533545984;
 2027 12'd2015: quad_corners = 76'd370378135516739332096;
 2028 12'd2016: quad_corners = 76'd370666225155939558914;
 2029 12'd2017: quad_corners = 76'd444741150352221201413;
 2030 12'd2018: quad_corners = 76'd518960331474067055111;
 2031 12'd2019: quad_corners = 76'd593035256670080262154;
 2032 12'd2020: quad_corners = 76'd667254437791926115340;
 2033 12'd2021: quad_corners = 76'd741329503725427677711;
 2034 12'd2022: quad_corners = 76'd815548684847005095955;
 2035 12'd2023: quad_corners = 76'd963554842263420720150;
 2036 12'd2024: quad_corners = 76'd1037918138573074518553;
 2037 12'd2025: quad_corners = 76'd1112137460432140291613;
 2038 12'd2026: quad_corners = 76'd1260287733036631772193;
 2039 12'd2027: quad_corners = 76'd1408438005641123252261;
 2040 12'd2028: quad_corners = 76'd1556444303795027756586;
 2041 12'd2029: quad_corners = 76'd1704738691587595092526;
 2042 12'd2030: quad_corners = 76'd1852889104929574928435;
 2043 12'd2031: quad_corners = 76'd2001039518271555288120;
 2044 12'd2032: quad_corners = 76'd2149334046801610979901;
 2045 12'd2033: quad_corners = 76'd2297628575331398235714;
 2046 12'd2034: quad_corners = 76'd2519565964967948366407;
 2047 12'd2035: quad_corners = 76'd2668004608685811478093;
 2048 12'd2036: quad_corners = 76'd2890086113510437465171;
 2049 12'd2037: quad_corners = 76'd2890086113510437465171;
 2050 12'd2038: quad_corners = 76'd2890086113510437465171;
 2051 12'd2039: quad_corners = 76'd2890086113510437465171;
 2052 12'd2040: quad_corners = 76'd2890086113510437465171;
 2053 12'd2041: quad_corners = 76'd2890086113510437465171;
 2054 12'd2042: quad_corners = 76'd2890086113510437465171;
 2055 12'd2043: quad_corners = 76'd2890086113510437465171;
 2056 12'd2044: quad_corners = 76'd2890086113510437465171;
 2057 12'd2045: quad_corners = 76'd2890086113510437465171;
 2058 12'd2046: quad_corners = 76'd2890086113510437465171;
 2059 12'd2047: quad_corners = 76'd2890086113510437465171;
 2060 12'd2048: quad_corners = 76'd442725226289312365568;
 2061 12'd2049: quad_corners = 76'd442725226289312365568;
 2062 12'd2050: quad_corners = 76'd442725226289312365568;
 2063 12'd2051: quad_corners = 76'd442725226289312365568;
 2064 12'd2052: quad_corners = 76'd442725226289312365568;
 2065 12'd2053: quad_corners = 76'd442725226289312365568;
 2066 12'd2054: quad_corners = 76'd442725226289312365568;
 2067 12'd2055: quad_corners = 76'd442725226289312365568;
 2068 12'd2056: quad_corners = 76'd442725226289312365568;

2069 12'd2057: quad_corners = 76'd442725226289312365568;
2070 12'd2058: quad_corners = 76'd442725226289312365568;
2071 12'd2059: quad_corners = 76'd442725226289312365568;
2072 12'd2060: quad_corners = 76'd442725226289312365568;
2073 12'd2061: quad_corners = 76'd442725226289312365568;
2074 12'd2062: quad_corners = 76'd442725226289312365568;
2075 12'd2063: quad_corners = 76'd442725226289312365568;
2076 12'd2064: quad_corners = 76'd442725226289312365568;
2077 12'd2065: quad_corners = 76'd442725226289312365568;
2078 12'd2066: quad_corners = 76'd442725226289312365568;
2079 12'd2067: quad_corners = 76'd442725226289312365568;
2080 12'd2068: quad_corners = 76'd442725226289312365568;
2081 12'd2069: quad_corners = 76'd442725226289312365568;
2082 12'd2070: quad_corners = 76'd442725226289312365568;
2083 12'd2071: quad_corners = 76'd442725226289312365568;
2084 12'd2072: quad_corners = 76'd442725226289312365568;
2085 12'd2073: quad_corners = 76'd442724944539994618880;
2086 12'd2074: quad_corners = 76'd442724663065554778624;
2087 12'd2075: quad_corners = 76'd442724381591114938368;
2088 12'd2076: quad_corners = 76'd442868356042238785024;
2089 12'd2077: quad_corners = 76'd516943281238788862976;
2090 12'd2078: quad_corners = 76'd517087114952156443138;
2091 12'd2079: quad_corners = 76'd517375204591356669955;
2092 12'd2080: quad_corners = 76'd591450129787638312453;
2093 12'd2081: quad_corners = 76'd665525195721408309768;
2094 12'd2082: quad_corners = 76'd739744376843522598922;
2095 12'd2083: quad_corners = 76'd740032466482454390285;
2096 12'd2084: quad_corners = 76'd814251506866543977487;
2097 12'd2085: quad_corners = 76'd962257805020716392978;
2098 12'd2086: quad_corners = 76'd1036476986142562246677;
2099 12'd2087: quad_corners = 76'd1110696167539017571353;
2100 12'd2088: quad_corners = 76'd1184915348660594989596;
2101 12'd2089: quad_corners = 76'd1333065762002574825504;
2102 12'd2090: quad_corners = 76'd1407284943124152767524;
2103 12'd2091: quad_corners = 76'd1555435356466132603432;
2104 12'd2092: quad_corners = 76'd1703585769808112439340;
2105 12'd2093: quad_corners = 76'd1851736183150092799024;
2106 12'd2094: quad_corners = 76'd2000030711955026397749;
2107 12'd2095: quad_corners = 76'd2148181125297006233146;
2108 12'd2096: quad_corners = 76'd2296475653827062449215;
2109 12'd2097: quad_corners = 76'd2444770182357118140484;
2110 12'd2098: quad_corners = 76'd2666851687182012038729;
2111 12'd2099: quad_corners = 76'd2815146215711799294543;
2112 12'd2100: quad_corners = 76'd3037227720536424756821;
2113 12'd2101: quad_corners = 76'd3037227720536424756821;
2114 12'd2102: quad_corners = 76'd3037227720536424756821;

```

2115      12'd2103: quad_corners = 76'd3037227720536424756821;
2116      12'd2104: quad_corners = 76'd3037227720536424756821;
2117      12'd2105: quad_corners = 76'd3037227720536424756821;
2118      12'd2106: quad_corners = 76'd3037227720536424756821;
2119      12'd2107: quad_corners = 76'd3037227720536424756821;
2120      12'd2108: quad_corners = 76'd3037227720536424756821;
2121      12'd2109: quad_corners = 76'd3037227720536424756821;
2122      12'd2110: quad_corners = 76'd3037227720536424756821;
2123      12'd2111: quad_corners = 76'd3037227720536424756821;
2124      12'd2112: quad_corners = 76'd664085591123558202880;
2125      12'd2113: quad_corners = 76'd664085591123558202880;
2126      12'd2114: quad_corners = 76'd664085591123558202880;
2127      12'd2115: quad_corners = 76'd664085591123558202880;
2128      12'd2116: quad_corners = 76'd664085591123558202880;
2129      12'd2117: quad_corners = 76'd664085591123558202880;
2130      12'd2118: quad_corners = 76'd664085591123558202880;
2131      12'd2119: quad_corners = 76'd664085591123558202880;
2132      12'd2120: quad_corners = 76'd664085591123558202880;
2133      12'd2121: quad_corners = 76'd664085591123558202880;
2134      12'd2122: quad_corners = 76'd664085591123558202880;
2135      12'd2123: quad_corners = 76'd664085591123558202880;
2136      12'd2124: quad_corners = 76'd664085591123558202880;
2137      12'd2125: quad_corners = 76'd664085591123558202880;
2138      12'd2126: quad_corners = 76'd664085591123558202880;
2139      12'd2127: quad_corners = 76'd664085591123558202880;
2140      12'd2128: quad_corners = 76'd664085591123558202880;
2141      12'd2129: quad_corners = 76'd664085591123558202880;
2142      12'd2130: quad_corners = 76'd664085591123558202880;
2143      12'd2131: quad_corners = 76'd664085591123558202880;
2144      12'd2132: quad_corners = 76'd664085591123558202880;
2145      12'd2133: quad_corners = 76'd664085591123558202880;
2146      12'd2134: quad_corners = 76'd664085591123558202880;
2147      12'd2135: quad_corners = 76'd664085591123558202880;
2148      12'd2136: quad_corners = 76'd664085591123558202880;
2149      12'd2137: quad_corners = 76'd664085450386606718464;
2150      12'd2138: quad_corners = 76'd664085168912166878720;
2151      12'd2139: quad_corners = 76'd664084887437727038465;
2152      12'd2140: quad_corners = 76'd664084605963287198210;
2153      12'd2141: quad_corners = 76'd664084465226335713284;
2154      12'd2142: quad_corners = 76'd738015275234541500421;
2155      12'd2143: quad_corners = 76'd738303364873741727239;
2156      12'd2144: quad_corners = 76'd812378290070023369737;
2157      12'd2145: quad_corners = 76'd886453356004061803019;
2158      12'd2146: quad_corners = 76'd886741445642993594381;
2159      12'd2147: quad_corners = 76'd960960626764839447568;
2160      12'd2148: quad_corners = 76'd1035035692698609445395;

```

2161 12'd2149: quad_corners = 76'd1109254874095333206037;
2162 12'd2150: quad_corners = 76'd1183474055217179059736;
2163 12'd2151: quad_corners = 76'd1331624468559427331100;
2164 12'd2152: quad_corners = 76'd1405843790418761540127;
2165 12'd2153: quad_corners = 76'd1553994204035619282979;
2166 12'd2154: quad_corners = 76'd1628213525894685056039;
2167 12'd2155: quad_corners = 76'd1776363939236664891947;
2168 12'd2156: quad_corners = 76'd1924514352578644727855;
2169 12'd2157: quad_corners = 76'd1998877789900664263731;
2170 12'd2158: quad_corners = 76'd2147172318430719955512;
2171 12'd2159: quad_corners = 76'd2295322731772699791420;
2172 12'd2160: quad_corners = 76'd2517404236872471596097;
2173 12'd2161: quad_corners = 76'd2665698765402527287878;
2174 12'd2162: quad_corners = 76'd2813993294207460886092;
2175 12'd2163: quad_corners = 76'd3036074799032354784337;
2176 12'd2164: quad_corners = 76'd3184513442750486331479;
2177 12'd2165: quad_corners = 76'd3184513442750486331479;
2178 12'd2166: quad_corners = 76'd3184513442750486331479;
2179 12'd2167: quad_corners = 76'd3184513442750486331479;
2180 12'd2168: quad_corners = 76'd3184513442750486331479;
2181 12'd2169: quad_corners = 76'd3184513442750486331479;
2182 12'd2170: quad_corners = 76'd3184513442750486331479;
2183 12'd2171: quad_corners = 76'd3184513442750486331479;
2184 12'd2172: quad_corners = 76'd3184513442750486331479;
2185 12'd2173: quad_corners = 76'd3184513442750486331479;
2186 12'd2174: quad_corners = 76'd3184513442750486331479;
2187 12'd2175: quad_corners = 76'd3184513442750486331479;
2188 12'd2176: quad_corners = 76'd885446096420144479747;
2189 12'd2177: quad_corners = 76'd885446096420144479747;
2190 12'd2178: quad_corners = 76'd885446096420144479747;
2191 12'd2179: quad_corners = 76'd885446096420144479747;
2192 12'd2180: quad_corners = 76'd885446096420144479747;
2193 12'd2181: quad_corners = 76'd885446096420144479747;
2194 12'd2182: quad_corners = 76'd885446096420144479747;
2195 12'd2183: quad_corners = 76'd885446096420144479747;
2196 12'd2184: quad_corners = 76'd885446096420144479747;
2197 12'd2185: quad_corners = 76'd885446096420144479747;
2198 12'd2186: quad_corners = 76'd885446096420144479747;
2199 12'd2187: quad_corners = 76'd885446096420144479747;
2200 12'd2188: quad_corners = 76'd885446096420144479747;
2201 12'd2189: quad_corners = 76'd885446096420144479747;
2202 12'd2190: quad_corners = 76'd885446096420144479747;
2203 12'd2191: quad_corners = 76'd885446096420144479747;
2204 12'd2192: quad_corners = 76'd885446096420144479747;
2205 12'd2193: quad_corners = 76'd885446096420144479747;
2206 12'd2194: quad_corners = 76'd885446096420144479747;

2207 12'd2195: quad_corners = 76'd885446096420144479747;
2208 12'd2196: quad_corners = 76'd885446096420144479747;
2209 12'd2197: quad_corners = 76'd885446096420144479747;
2210 12'd2198: quad_corners = 76'd885446096420144479747;
2211 12'd2199: quad_corners = 76'd885446096420144479747;
2212 12'd2200: quad_corners = 76'd885446096420144479747;
2213 12'd2201: quad_corners = 76'd885445814945704640003;
2214 12'd2202: quad_corners = 76'd885445533471264800260;
2215 12'd2203: quad_corners = 76'd885445392734313315845;
2216 12'd2204: quad_corners = 76'd885445111534751383046;
2217 12'd2205: quad_corners = 76'd959231947092638105096;
2218 12'd2206: quad_corners = 76'd959231665618198264841;
2219 12'd2207: quad_corners = 76'd1033018501176084986379;
2220 12'd2208: quad_corners = 76'd1033306590815285213197;
2221 12'd2209: quad_corners = 76'd1107381656749055211023;
2222 12'd2210: quad_corners = 76'd1181456722957703115793;
2223 12'd2211: quad_corners = 76'd1181888927784979198483;
2224 12'd2212: quad_corners = 76'd1255963993718749196310;
2225 12'd2213: quad_corners = 76'd1330183315578083405337;
2226 12'd2214: quad_corners = 76'd1478189614007133727772;
2227 12'd2215: quad_corners = 76'd1552553051054543792671;
2228 12'd2216: quad_corners = 76'd1626772372913878001698;
2229 12'd2217: quad_corners = 76'd1700991695048090117670;
2230 12'd2218: quad_corners = 76'd1849142108390069953578;
2231 12'd2219: quad_corners = 76'd1997292522007196131885;
2232 12'd2220: quad_corners = 76'd2071655959054337761330;
2233 12'd2221: quad_corners = 76'd2219806372671195503670;
2234 12'd2222: quad_corners = 76'd2368100901201251195450;
2235 12'd2223: quad_corners = 76'd2516251314818108938303;
2236 12'd2224: quad_corners = 76'd2664545843348164630084;
2237 12'd2225: quad_corners = 76'd2812840372153098228297;
2238 12'd2226: quad_corners = 76'd3034921876977992126542;
2239 12'd2227: quad_corners = 76'd3183360520971001581139;
2240 12'd2228: quad_corners = 76'd3331655190238545627737;
2241 12'd2229: quad_corners = 76'd3331655190238545627737;
2242 12'd2230: quad_corners = 76'd3331655190238545627737;
2243 12'd2231: quad_corners = 76'd3331655190238545627737;
2244 12'd2232: quad_corners = 76'd3331655190238545627737;
2245 12'd2233: quad_corners = 76'd3331655190238545627737;
2246 12'd2234: quad_corners = 76'd3331655190238545627737;
2247 12'd2235: quad_corners = 76'd3331655190238545627737;
2248 12'd2236: quad_corners = 76'd3331655190238545627737;
2249 12'd2237: quad_corners = 76'd3331655190238545627737;
2250 12'd2238: quad_corners = 76'd3331655190238545627737;
2251 12'd2239: quad_corners = 76'd3331655190238545627737;
2252 12'd2240: quad_corners = 76'd1180593578011568963079;

2253 12'd2241: quad_corners = 76'd1180593578011568963079;
2254 12'd2242: quad_corners = 76'd1180593578011568963079;
2255 12'd2243: quad_corners = 76'd1180593578011568963079;
2256 12'd2244: quad_corners = 76'd1180593578011568963079;
2257 12'd2245: quad_corners = 76'd1180593578011568963079;
2258 12'd2246: quad_corners = 76'd1180593578011568963079;
2259 12'd2247: quad_corners = 76'd1180593578011568963079;
2260 12'd2248: quad_corners = 76'd1180593578011568963079;
2261 12'd2249: quad_corners = 76'd1180593578011568963079;
2262 12'd2250: quad_corners = 76'd1180593578011568963079;
2263 12'd2251: quad_corners = 76'd1180593578011568963079;
2264 12'd2252: quad_corners = 76'd1180593578011568963079;
2265 12'd2253: quad_corners = 76'd1180593578011568963079;
2266 12'd2254: quad_corners = 76'd1180593578011568963079;
2267 12'd2255: quad_corners = 76'd1180593578011568963079;
2268 12'd2256: quad_corners = 76'd1180593578011568963079;
2269 12'd2257: quad_corners = 76'd1180593578011568963079;
2270 12'd2258: quad_corners = 76'd1180593578011568963079;
2271 12'd2259: quad_corners = 76'd1180593578011568963079;
2272 12'd2260: quad_corners = 76'd1180593578011568963079;
2273 12'd2261: quad_corners = 76'd1180593578011568963079;
2274 12'd2262: quad_corners = 76'd1180593578011568963079;
2275 12'd2263: quad_corners = 76'd1180593578011568963079;
2276 12'd2264: quad_corners = 76'd1180593578011568963079;
2277 12'd2265: quad_corners = 76'd1180593296812007030280;
2278 12'd2266: quad_corners = 76'd1180593015337567190537;
2279 12'd2267: quad_corners = 76'd1180592874600884141577;
2280 12'd2268: quad_corners = 76'd1180592593126444301835;
2281 12'd2269: quad_corners = 76'd1180592452389492817420;
2282 12'd2270: quad_corners = 76'd125437928822257446413;
2283 12'd2271: quad_corners = 76'd1254379006747817606671;
2284 12'd2272: quad_corners = 76'd1328165842305435893265;
2285 12'd2273: quad_corners = 76'd1328309816756560264211;
2286 12'd2274: quad_corners = 76'd1402385023702964959765;
2287 12'd2275: quad_corners = 76'd1476604345562299168791;
2288 12'd2276: quad_corners = 76'd1550823667421633377818;
2289 12'd2277: quad_corners = 76'd1624898874368038073373;
2290 12'd2278: quad_corners = 76'd1699262311415448138272;
2291 12'd2279: quad_corners = 76'd1773481633549660254243;
2292 12'd2280: quad_corners = 76'd1847700955408994463270;
2293 12'd2281: quad_corners = 76'd1995851369026120641577;
2294 12'd2282: quad_corners = 76'd2070070690885186415149;
2295 12'd2283: quad_corners = 76'd2218221104502312593457;
2296 12'd2284: quad_corners = 76'd2292584541549722658357;
2297 12'd2285: quad_corners = 76'd2440879070354656257081;
2298 12'd2286: quad_corners = 76'd2589029483696904528445;

2299 12'd2287: quad_corners = 76'd2737324012501838127170;
2300 12'd2288: quad_corners = 76'd2885474426118695869510;
2301 12'd2289: quad_corners = 76'd3033768954648751561291;
2302 12'd2290: quad_corners = 76'd3182063483453685160016;
2303 12'd2291: quad_corners = 76'd3330502127446694614613;
2304 12'd2292: quad_corners = 76'd3552583773283954774619;
2305 12'd2293: quad_corners = 76'd3552583773283954774619;
2306 12'd2294: quad_corners = 76'd3552583773283954774619;
2307 12'd2295: quad_corners = 76'd3552583773283954774619;
2308 12'd2296: quad_corners = 76'd3552583773283954774619;
2309 12'd2297: quad_corners = 76'd3552583773283954774619;
2310 12'd2298: quad_corners = 76'd3552583773283954774619;
2311 12'd2299: quad_corners = 76'd3552583773283954774619;
2312 12'd2300: quad_corners = 76'd3552583773283954774619;
2313 12'd2301: quad_corners = 76'd3552583773283954774619;
2314 12'd2302: quad_corners = 76'd3552583773283954774619;
2315 12'd2303: quad_corners = 76'd3552583773283954774619;
2316 12'd2304: quad_corners = 76'd1475741059602993446412;
2317 12'd2305: quad_corners = 76'd1475741059602993446412;
2318 12'd2306: quad_corners = 76'd1475741059602993446412;
2319 12'd2307: quad_corners = 76'd1475741059602993446412;
2320 12'd2308: quad_corners = 76'd1475741059602993446412;
2321 12'd2309: quad_corners = 76'd1475741059602993446412;
2322 12'd2310: quad_corners = 76'd1475741059602993446412;
2323 12'd2311: quad_corners = 76'd1475741059602993446412;
2324 12'd2312: quad_corners = 76'd1475741059602993446412;
2325 12'd2313: quad_corners = 76'd1475741059602993446412;
2326 12'd2314: quad_corners = 76'd1475741059602993446412;
2327 12'd2315: quad_corners = 76'd1475741059602993446412;
2328 12'd2316: quad_corners = 76'd1475741059602993446412;
2329 12'd2317: quad_corners = 76'd1475741059602993446412;
2330 12'd2318: quad_corners = 76'd1475741059602993446412;
2331 12'd2319: quad_corners = 76'd1475741059602993446412;
2332 12'd2320: quad_corners = 76'd1475741059602993446412;
2333 12'd2321: quad_corners = 76'd1475741059602993446412;
2334 12'd2322: quad_corners = 76'd1475741059602993446412;
2335 12'd2323: quad_corners = 76'd1475741059602993446412;
2336 12'd2324: quad_corners = 76'd1475741059602993446412;
2337 12'd2325: quad_corners = 76'd1475741059602993446412;
2338 12'd2326: quad_corners = 76'd1475741059602993446412;
2339 12'd2327: quad_corners = 76'd1475741059602993446412;
2340 12'd2328: quad_corners = 76'd1475741059602993446412;
2341 12'd2329: quad_corners = 76'd1475740778403431513613;
2342 12'd2330: quad_corners = 76'd1475740637666480029197;
2343 12'd2331: quad_corners = 76'd1475740356192308624910;
2344 12'd2332: quad_corners = 76'd1475740215455357140495;

2345 12'd2333: quad_corners = 76'd1475740074993283563025;
2346 12'd2334: quad_corners = 76'd1475739793518843723282;
2347 12'd2335: quad_corners = 76'd1549526629076730445332;
2348 12'd2336: quad_corners = 76'd1549526629352145223190;
2349 12'd2337: quad_corners = 76'd1623313605647520300567;
2350 12'd2338: quad_corners = 76'd1697244697130702798362;
2351 12'd2339: quad_corners = 76'd1697677042970345143324;
2352 12'd2340: quad_corners = 76'd1771752249641603496478;
2353 12'd2341: quad_corners = 76'd1845971571776084048417;
2354 12'd2342: quad_corners = 76'd1920190893635418257444;
2355 12'd2343: quad_corners = 76'd1994410215769630373415;
2356 12'd2344: quad_corners = 76'd2142560629111878644778;
2357 12'd2345: quad_corners = 76'd2216779951246090760749;
2358 12'd2346: quad_corners = 76'd2291143388568378732593;
2359 12'd2347: quad_corners = 76'd2439293801910627003956;
2360 12'd2348: quad_corners = 76'd2513657239232914975800;
2361 12'd2349: quad_corners = 76'd2661807652849772718652;
2362 12'd2350: quad_corners = 76'd2809958066192020990016;
2363 12'd2351: quad_corners = 76'd2958252594996954588740;
2364 12'd2352: quad_corners = 76'd3106403008614080767049;
2365 12'd2353: quad_corners = 76'd3254697537419014365774;
2366 12'd2354: quad_corners = 76'd3402992065949070057042;
2367 12'd2355: quad_corners = 76'd3551286735491492011095;
2368 12'd2356: quad_corners = 76'd3699725520221989821021;
2369 12'd2357: quad_corners = 76'd3699725520221989821021;
2370 12'd2358: quad_corners = 76'd3699725520221989821021;
2371 12'd2359: quad_corners = 76'd3699725520221989821021;
2372 12'd2360: quad_corners = 76'd3699725520221989821021;
2373 12'd2361: quad_corners = 76'd3699725520221989821021;
2374 12'd2362: quad_corners = 76'd3699725520221989821021;
2375 12'd2363: quad_corners = 76'd3699725520221989821021;
2376 12'd2364: quad_corners = 76'd3699725520221989821021;
2377 12'd2365: quad_corners = 76'd3699725520221989821021;
2378 12'd2366: quad_corners = 76'd3699725520221989821021;
2379 12'd2367: quad_corners = 76'd3699725520221989821021;
2380 12'd2368: quad_corners = 76'd1770888681931637849618;
2381 12'd2369: quad_corners = 76'd1770888681931637849618;
2382 12'd2370: quad_corners = 76'd1770888681931637849618;
2383 12'd2371: quad_corners = 76'd1770888681931637849618;
2384 12'd2372: quad_corners = 76'd1770888681931637849618;
2385 12'd2373: quad_corners = 76'd1770888681931637849618;
2386 12'd2374: quad_corners = 76'd1770888681931637849618;
2387 12'd2375: quad_corners = 76'd1770888681931637849618;
2388 12'd2376: quad_corners = 76'd1770888681931637849618;
2389 12'd2377: quad_corners = 76'd1770888681931637849618;
2390 12'd2378: quad_corners = 76'd1770888681931637849618;

2391 12'd2379: quad_corners = 76'd1770888681931637849618;
2392 12'd2380: quad_corners = 76'd1770888681931637849618;
2393 12'd2381: quad_corners = 76'd1770888681931637849618;
2394 12'd2382: quad_corners = 76'd1770888681931637849618;
2395 12'd2383: quad_corners = 76'd1770888681931637849618;
2396 12'd2384: quad_corners = 76'd1770888681931637849618;
2397 12'd2385: quad_corners = 76'd1770888681931637849618;
2398 12'd2386: quad_corners = 76'd1770888681931637849618;
2399 12'd2387: quad_corners = 76'd1770888681931637849618;
2400 12'd2388: quad_corners = 76'd1770888681931637849618;
2401 12'd2389: quad_corners = 76'd1770888681931637849618;
2402 12'd2390: quad_corners = 76'd1770888681931637849618;
2403 12'd2391: quad_corners = 76'd1770888681931637849618;
2404 12'd2392: quad_corners = 76'd1770888681931637849618;
2405 12'd2393: quad_corners = 76'd1770888400457466445330;
2406 12'd2394: quad_corners = 76'd1770888259720514960915;
2407 12'd2395: quad_corners = 76'd1770887978521221463572;
2408 12'd2396: quad_corners = 76'd1770887837784269979157;
2409 12'd2397: quad_corners = 76'd1770887697047318494742;
2410 12'd2398: quad_corners = 76'd1844674532880083123735;
2411 12'd2399: quad_corners = 76'd1844674532880619994649;
2412 12'd2400: quad_corners = 76'd1844674533156034772507;
2413 12'd2401: quad_corners = 76'd1918461509451409849884;
2414 12'd2402: quad_corners = 76'd1992248486021662834206;
2415 12'd2403: quad_corners = 76'd1992536716398082981409;
2416 12'd2404: quad_corners = 76'd2066611923344487677475;
2417 12'd2405: quad_corners = 76'd2140831245203821887013;
2418 12'd2406: quad_corners = 76'd2215050567338302438440;
2419 12'd2407: quad_corners = 76'd2289269889197636647467;
2420 12'd2408: quad_corners = 76'd2363489211331848763438;
2421 12'd2409: quad_corners = 76'd2511639624948974941745;
2422 12'd2410: quad_corners = 76'd2585858946808309151284;
2423 12'd2411: quad_corners = 76'd2660222384130597123128;
2424 12'd2412: quad_corners = 76'd2808372797747723301436;
2425 12'd2413: quad_corners = 76'd2882736235070011273279;
2426 12'd2414: quad_corners = 76'd3030886648686869016131;
2427 12'd2415: quad_corners = 76'd3179181177217193143367;
2428 12'd2416: quad_corners = 76'd3327331590834050886220;
2429 12'd2417: quad_corners = 76'd3475626119639252920400;
2430 12'd2418: quad_corners = 76'd3623920648444186518613;
2431 12'd2419: quad_corners = 76'd3772215317986608472666;
2432 12'd2420: quad_corners = 76'd3920510128266518782047;
2433 12'd2421: quad_corners = 76'd3920510128266518782047;
2434 12'd2422: quad_corners = 76'd3920510128266518782047;
2435 12'd2423: quad_corners = 76'd3920510128266518782047;
2436 12'd2424: quad_corners = 76'd3920510128266518782047;

2437 12'd2425: quad_corners = 76'd3920510128266518782047;
2438 12'd2426: quad_corners = 76'd3920510128266518782047;
2439 12'd2427: quad_corners = 76'd3920510128266518782047;
2440 12'd2428: quad_corners = 76'd3920510128266518782047;
2441 12'd2429: quad_corners = 76'd3920510128266518782047;
2442 12'd2430: quad_corners = 76'd3920510128266518782047;
2443 12'd2431: quad_corners = 76'd3920510128266518782047;
2444 12'd2432: quad_corners = 76'd2139823280280510987800;
2445 12'd2433: quad_corners = 76'd2139823280280510987800;
2446 12'd2434: quad_corners = 76'd2139823280280510987800;
2447 12'd2435: quad_corners = 76'd2139823280280510987800;
2448 12'd2436: quad_corners = 76'd2139823280280510987800;
2449 12'd2437: quad_corners = 76'd2139823280280510987800;
2450 12'd2438: quad_corners = 76'd2139823280280510987800;
2451 12'd2439: quad_corners = 76'd2139823280280510987800;
2452 12'd2440: quad_corners = 76'd2139823280280510987800;
2453 12'd2441: quad_corners = 76'd2139823280280510987800;
2454 12'd2442: quad_corners = 76'd2139823280280510987800;
2455 12'd2443: quad_corners = 76'd2139823280280510987800;
2456 12'd2444: quad_corners = 76'd2139823280280510987800;
2457 12'd2445: quad_corners = 76'd2139823280280510987800;
2458 12'd2446: quad_corners = 76'd2139823280280510987800;
2459 12'd2447: quad_corners = 76'd2139823280280510987800;
2460 12'd2448: quad_corners = 76'd2139823280280510987800;
2461 12'd2449: quad_corners = 76'd2139823280280510987800;
2462 12'd2450: quad_corners = 76'd2139823280280510987800;
2463 12'd2451: quad_corners = 76'd2139823280280510987800;
2464 12'd2452: quad_corners = 76'd2139823280280510987800;
2465 12'd2453: quad_corners = 76'd2139823280280510987800;
2466 12'd2454: quad_corners = 76'd2139823280280510987800;
2467 12'd2455: quad_corners = 76'd2139823280280510987800;
2468 12'd2456: quad_corners = 76'd2139823280280510987800;
2469 12'd2457: quad_corners = 76'd2066036022511232941592;
2470 12'd2458: quad_corners = 76'd2066035882049427799577;
2471 12'd2459: quad_corners = 76'd2066035600574987959834;
2472 12'd2460: quad_corners = 76'd2139822436133143117339;
2473 12'd2461: quad_corners = 76'd2139822436408557895196;
2474 12'd2462: quad_corners = 76'd2139822436409094766109;
2475 12'd2463: quad_corners = 76'd2139822436684509543966;
2476 12'd2464: quad_corners = 76'd2213609412979884621344;
2477 12'd2465: quad_corners = 76'd2213609413255299399202;
2478 12'd2466: quad_corners = 76'd2287396389550674476580;
2479 12'd2467: quad_corners = 76'd2361183366120659025446;
2480 12'd2468: quad_corners = 76'd2361471596772225515048;
2481 12'd2469: quad_corners = 76'd2435690918631559724586;
2482 12'd2470: quad_corners = 76'd2509910240766040276525;

2483 12'd2471: quad_corners = 76'd2584129562900252393007;
2484 12'd2472: quad_corners = 76'd2658348884759855037490;
2485 12'd2473: quad_corners = 76'd2732568206894067153461;
2486 12'd2474: quad_corners = 76'd2880718620511193332280;
2487 12'd2475: quad_corners = 76'd2955082057833481304124;
2488 12'd2476: quad_corners = 76'd3029301379692815513151;
2489 12'd2477: quad_corners = 76'd3177451793309941691459;
2490 12'd2478: quad_corners = 76'd3325746322114875290183;
2491 12'd2479: quad_corners = 76'd3400109759437163262027;
2492 12'd2480: quad_corners = 76'd3548260173054289440335;
2493 12'd2481: quad_corners = 76'd3696554701859223039059;
2494 12'd2482: quad_corners = 76'd3844849371401913428567;
2495 12'd2483: quad_corners = 76'd3993144040944335382620;
2496 12'd2484: quad_corners = 76'd4141438710486757336673;
2497 12'd2485: quad_corners = 76'd4141438710486757336673;
2498 12'd2486: quad_corners = 76'd4141438710486757336673;
2499 12'd2487: quad_corners = 76'd4141438710486757336673;
2500 12'd2488: quad_corners = 76'd4141438710486757336673;
2501 12'd2489: quad_corners = 76'd4141438710486757336673;
2502 12'd2490: quad_corners = 76'd4141438710486757336673;
2503 12'd2491: quad_corners = 76'd4141438710486757336673;
2504 12'd2492: quad_corners = 76'd4141438710486757336673;
2505 12'd2493: quad_corners = 76'd4141438710486757336673;
2506 12'd2494: quad_corners = 76'd4141438710486757336673;
2507 12'd2495: quad_corners = 76'd4141438710486757336673;
2508 12'd2496: quad_corners = 76'd2434970902334277484062;
2509 12'd2497: quad_corners = 76'd2434970902334277484062;
2510 12'd2498: quad_corners = 76'd2434970902334277484062;
2511 12'd2499: quad_corners = 76'd2434970902334277484062;
2512 12'd2500: quad_corners = 76'd2434970902334277484062;
2513 12'd2501: quad_corners = 76'd2434970902334277484062;
2514 12'd2502: quad_corners = 76'd2434970902334277484062;
2515 12'd2503: quad_corners = 76'd2434970902334277484062;
2516 12'd2504: quad_corners = 76'd2434970902334277484062;
2517 12'd2505: quad_corners = 76'd2434970902334277484062;
2518 12'd2506: quad_corners = 76'd2434970902334277484062;
2519 12'd2507: quad_corners = 76'd2434970902334277484062;
2520 12'd2508: quad_corners = 76'd2434970902334277484062;
2521 12'd2509: quad_corners = 76'd2434970902334277484062;
2522 12'd2510: quad_corners = 76'd2434970902334277484062;
2523 12'd2511: quad_corners = 76'd2434970902334277484062;
2524 12'd2512: quad_corners = 76'd2434970902334277484062;
2525 12'd2513: quad_corners = 76'd2434970902334277484062;
2526 12'd2514: quad_corners = 76'd2434970902334277484062;
2527 12'd2515: quad_corners = 76'd2434970902334277484062;
2528 12'd2516: quad_corners = 76'd2434970902334277484062;

2529 12'd2517: quad_corners = 76'd2434970902334277484062;
 2530 12'd2518: quad_corners = 76'd2434970902334277484062;
 2531 12'd2519: quad_corners = 76'd2434970902334277484062;
 2532 12'd2520: quad_corners = 76'd2434970902334277484062;
 2533 12'd2521: quad_corners = 76'd2434970761597594435102;
 2534 12'd2522: quad_corners = 76'd2434970480398032502303;
 2535 12'd2523: quad_corners = 76'd2434970339661349453344;
 2536 12'd2524: quad_corners = 76'd2434970339936764231201;
 2537 12'd2525: quad_corners = 76'd2434970339937301102114;
 2538 12'd2526: quad_corners = 76'd2508757316507822521891;
 2539 12'd2527: quad_corners = 76'd2508757316508359392804;
 2540 12'd2528: quad_corners = 76'd2508757316783774170662;
 2541 12'd2529: quad_corners = 76'd2582544293079149248039;
 2542 12'd2530: quad_corners = 76'd2656331269649133796905;
 2543 12'd2531: quad_corners = 76'd2656331269649670667819;
 2544 12'd2532: quad_corners = 76'd2730262361407999508013;
 2545 12'd2533: quad_corners = 76'd2804481683542211624495;
 2546 12'd2534: quad_corners = 76'd2878556890213738413618;
 2547 12'd2535: quad_corners = 76'd2878989236053112323636;
 2548 12'd2536: quad_corners = 76'd2953208558187592875575;
 2549 12'd2537: quad_corners = 76'd3101214856616643198522;
 2550 12'd2538: quad_corners = 76'd3175578293938931170365;
 2551 12'd2539: quad_corners = 76'd3249797616073143286336;
 2552 12'd2540: quad_corners = 76'd3324161053120553351747;
 2553 12'd2541: quad_corners = 76'd3472311466737679530055;
 2554 12'd2542: quad_corners = 76'd3546674904059967501898;
 2555 12'd2543: quad_corners = 76'd3694825317677093680206;
 2556 12'd2544: quad_corners = 76'd3769188754999113216594;
 2557 12'd2545: quad_corners = 76'd3917483283804315250774;
 2558 12'd2546: quad_corners = 76'd4065633838158661348954;
 2559 12'd2547: quad_corners = 76'd4213928648713718000734;
 2560 12'd2548: quad_corners = 76'd4362223318256139954787;
 2561 12'd2549: quad_corners = 76'd4362223318256139954787;
 2562 12'd2550: quad_corners = 76'd4362223318256139954787;
 2563 12'd2551: quad_corners = 76'd4362223318256139954787;
 2564 12'd2552: quad_corners = 76'd4362223318256139954787;
 2565 12'd2553: quad_corners = 76'd4362223318256139954787;
 2566 12'd2554: quad_corners = 76'd4362223318256139954787;
 2567 12'd2555: quad_corners = 76'd4362223318256139954787;
 2568 12'd2556: quad_corners = 76'd4362223318256139954787;
 2569 12'd2557: quad_corners = 76'd4362223318256139954787;
 2570 12'd2558: quad_corners = 76'd4362223318256139954787;
 2571 12'd2559: quad_corners = 76'd4362223318256139954787;
 2572 12'd2560: quad_corners = 76'd2877692617715208748581;
 2573 12'd2561: quad_corners = 76'd2877692617715208748581;
 2574 12'd2562: quad_corners = 76'd2877692617715208748581;

2575 12'd2563: quad_corners = 76'd2877692617715208748581;
2576 12'd2564: quad_corners = 76'd2877692617715208748581;
2577 12'd2565: quad_corners = 76'd2877692617715208748581;
2578 12'd2566: quad_corners = 76'd2877692617715208748581;
2579 12'd2567: quad_corners = 76'd2877692617715208748581;
2580 12'd2568: quad_corners = 76'd2877692617715208748581;
2581 12'd2569: quad_corners = 76'd2877692617715208748581;
2582 12'd2570: quad_corners = 76'd2877692617715208748581;
2583 12'd2571: quad_corners = 76'd2877692617715208748581;
2584 12'd2572: quad_corners = 76'd2877692617715208748581;
2585 12'd2573: quad_corners = 76'd2877692617715208748581;
2586 12'd2574: quad_corners = 76'd2877692617715208748581;
2587 12'd2575: quad_corners = 76'd2877692617715208748581;
2588 12'd2576: quad_corners = 76'd2877692617715208748581;
2589 12'd2577: quad_corners = 76'd2877692617715208748581;
2590 12'd2578: quad_corners = 76'd2877692617715208748581;
2591 12'd2579: quad_corners = 76'd2877692617715208748581;
2592 12'd2580: quad_corners = 76'd2877692617715208748581;
2593 12'd2581: quad_corners = 76'd2877692617715208748581;
2594 12'd2582: quad_corners = 76'd2877692617715208748581;
2595 12'd2583: quad_corners = 76'd2877692617715208748581;
2596 12'd2584: quad_corners = 76'd2877692617715208748581;
2597 12'd2585: quad_corners = 76'd2803905359946199137829;
2598 12'd2586: quad_corners = 76'd2803905219209516088870;
2599 12'd2587: quad_corners = 76'd2803905219484930866726;
2600 12'd2588: quad_corners = 76'd2803905219485467737639;
2601 12'd2589: quad_corners = 76'd2877692196055989157416;
2602 12'd2590: quad_corners = 76'd2877692196056526028329;
2603 12'd2591: quad_corners = 76'd2877692196331940806187;
2604 12'd2592: quad_corners = 76'd2877692196332477677100;
2605 12'd2593: quad_corners = 76'd2951479172902730661421;
2606 12'd2594: quad_corners = 76'd2951479173178145439279;
2607 12'd2595: quad_corners = 76'd3025266149473520516657;
2608 12'd2596: quad_corners = 76'd3099053126043505065523;
2609 12'd2597: quad_corners = 76'd3099341356695071555125;
2610 12'd2598: quad_corners = 76'd3173560678829283671607;
2611 12'd2599: quad_corners = 76'd3247780000688886316602;
2612 12'd2600: quad_corners = 76'd3321999322823098433084;
2613 12'd2601: quad_corners = 76'd3396218644957310549567;
2614 12'd2602: quad_corners = 76'd3470437967091791101505;
2615 12'd2603: quad_corners = 76'd3544657289226003217988;
2616 12'd2604: quad_corners = 76'd3619020726548291189831;
2617 12'd2605: quad_corners = 76'd3767027024977341512267;
2618 12'd2606: quad_corners = 76'd3841390462299629484110;
2619 12'd2607: quad_corners = 76'd3915753899621649021009;
2620 12'd2608: quad_corners = 76'd4063904313238775199317;

2621 12'd2609: quad_corners = 76'd4138412006486627382361;
2622 12'd2610: quad_corners = 76'd4286562560840973480541;
2623 12'd2611: quad_corners = 76'd4434713255933076369505;
2624 12'd2612: quad_corners = 76'd4583007925475498323557;
2625 12'd2613: quad_corners = 76'd4583007925475498323557;
2626 12'd2614: quad_corners = 76'd4583007925475498323557;
2627 12'd2615: quad_corners = 76'd4583007925475498323557;
2628 12'd2616: quad_corners = 76'd4583007925475498323557;
2629 12'd2617: quad_corners = 76'd4583007925475498323557;
2630 12'd2618: quad_corners = 76'd4583007925475498323557;
2631 12'd2619: quad_corners = 76'd4583007925475498323557;
2632 12'd2620: quad_corners = 76'd4583007925475498323557;
2633 12'd2621: quad_corners = 76'd4583007925475498323557;
2634 12'd2622: quad_corners = 76'd4583007925475498323557;
2635 12'd2623: quad_corners = 76'd4583007925475498323557;
2636 12'd2624: quad_corners = 76'd3246627356526423899692;
2637 12'd2625: quad_corners = 76'd3246627356526423899692;
2638 12'd2626: quad_corners = 76'd3246627356526423899692;
2639 12'd2627: quad_corners = 76'd3246627356526423899692;
2640 12'd2628: quad_corners = 76'd3246627356526423899692;
2641 12'd2629: quad_corners = 76'd3246627356526423899692;
2642 12'd2630: quad_corners = 76'd3246627356526423899692;
2643 12'd2631: quad_corners = 76'd3246627356526423899692;
2644 12'd2632: quad_corners = 76'd3246627356526423899692;
2645 12'd2633: quad_corners = 76'd3246627356526423899692;
2646 12'd2634: quad_corners = 76'd3246627356526423899692;
2647 12'd2635: quad_corners = 76'd3246627356526423899692;
2648 12'd2636: quad_corners = 76'd3246627356526423899692;
2649 12'd2637: quad_corners = 76'd3246627356526423899692;
2650 12'd2638: quad_corners = 76'd3246627356526423899692;
2651 12'd2639: quad_corners = 76'd3246627356526423899692;
2652 12'd2640: quad_corners = 76'd3246627356526423899692;
2653 12'd2641: quad_corners = 76'd3246627356526423899692;
2654 12'd2642: quad_corners = 76'd3246627356526423899692;
2655 12'd2643: quad_corners = 76'd3246627356526423899692;
2656 12'd2644: quad_corners = 76'd3246627356526423899692;
2657 12'd2645: quad_corners = 76'd3246627356526423899692;
2658 12'd2646: quad_corners = 76'd3246627356526423899692;
2659 12'd2647: quad_corners = 76'd3246627356526423899692;
2660 12'd2648: quad_corners = 76'd3246627356526423899692;
2661 12'd2649: quad_corners = 76'd3246627075052252495405;
2662 12'd2650: quad_corners = 76'd3246627075327935708717;
2663 12'd2651: quad_corners = 76'd3246627075328472579630;
2664 12'd2652: quad_corners = 76'd3246627075603887357486;
2665 12'd2653: quad_corners = 76'd3246627075604692663855;
2666 12'd2654: quad_corners = 76'd3246627075880107441712;

2667 12'd2655: quad_corners = 76'd3246627075880644312625;
2668 12'd2656: quad_corners = 76'd3320414052450897296947;
2669 12'd2657: quad_corners = 76'd3320414052726312074804;
2670 12'd2658: quad_corners = 76'd3320414052726848945717;
2671 12'd2659: quad_corners = 76'd3394201029297101930039;
2672 12'd2660: quad_corners = 76'd3467988005867086478905;
2673 12'd2661: quad_corners = 76'd3468276236243775061563;
2674 12'd2662: quad_corners = 76'd3542351443190179757629;
2675 12'd2663: quad_corners = 76'd3616570765324391874111;
2676 12'd2664: quad_corners = 76'd3690645972270528134721;
2677 12'd2665: quad_corners = 76'd3691078318110170480196;
2678 12'd2666: quad_corners = 76'd3765297640244382596678;
2679 12'd2667: quad_corners = 76'd3839516962378594713161;
2680 12'd2668: quad_corners = 76'd3987667375995720891980;
2681 12'd2669: quad_corners = 76'd4061886698129933007951;
2682 12'd2670: quad_corners = 76'd4136250135452220979794;
2683 12'd2671: quad_corners = 76'd4210469457586433096277;
2684 12'd2672: quad_corners = 76'd4358620011940779194456;
2685 12'd2673: quad_corners = 76'd443298359000055521628;
2686 12'd2674: quad_corners = 76'd4507347168060331848800;
2687 12'd2675: quad_corners = 76'd4655641978340242158179;
2688 12'd2676: quad_corners = 76'd4803792673706954518631;
2689 12'd2677: quad_corners = 76'd4803792673706954518631;
2690 12'd2678: quad_corners = 76'd4803792673706954518631;
2691 12'd2679: quad_corners = 76'd4803792673706954518631;
2692 12'd2680: quad_corners = 76'd4803792673706954518631;
2693 12'd2681: quad_corners = 76'd4803792673706954518631;
2694 12'd2682: quad_corners = 76'd4803792673706954518631;
2695 12'd2683: quad_corners = 76'd4803792673706954518631;
2696 12'd2684: quad_corners = 76'd4803792673706954518631;
2697 12'd2685: quad_corners = 76'd4803792673706954518631;
2698 12'd2686: quad_corners = 76'd4803792673706954518631;
2699 12'd2687: quad_corners = 76'd4803792673706954518631;
2700 12'd2688: quad_corners = 76'd3689349071357599350324;
2701 12'd2689: quad_corners = 76'd3689349071357599350324;
2702 12'd2690: quad_corners = 76'd3689349071357599350324;
2703 12'd2691: quad_corners = 76'd3689349071357599350324;
2704 12'd2692: quad_corners = 76'd3689349071357599350324;
2705 12'd2693: quad_corners = 76'd3689349071357599350324;
2706 12'd2694: quad_corners = 76'd3689349071357599350324;
2707 12'd2695: quad_corners = 76'd3689349071357599350324;
2708 12'd2696: quad_corners = 76'd3689349071357599350324;
2709 12'd2697: quad_corners = 76'd3689349071357599350324;
2710 12'd2698: quad_corners = 76'd3689349071357599350324;
2711 12'd2699: quad_corners = 76'd3689349071357599350324;
2712 12'd2700: quad_corners = 76'd3689349071357599350324;

2713 12'd2701: quad_corners = 76'd3689349071357599350324;
2714 12'd2702: quad_corners = 76'd3689349071357599350324;
2715 12'd2703: quad_corners = 76'd3689349071357599350324;
2716 12'd2704: quad_corners = 76'd3689349071357599350324;
2717 12'd2705: quad_corners = 76'd3689349071357599350324;
2718 12'd2706: quad_corners = 76'd3689349071357599350324;
2719 12'd2707: quad_corners = 76'd3689349071357599350324;
2720 12'd2708: quad_corners = 76'd3689349071357599350324;
2721 12'd2709: quad_corners = 76'd3689349071357599350324;
2722 12'd2710: quad_corners = 76'd3689349071357599350324;
2723 12'd2711: quad_corners = 76'd3689349071357599350324;
2724 12'd2712: quad_corners = 76'd3689349071357599350324;
2725 12'd2713: quad_corners = 76'd3689348930895794208308;
2726 12'd2714: quad_corners = 76'd3689348930896331079221;
2727 12'd2715: quad_corners = 76'd3689348931172014292533;
2728 12'd2716: quad_corners = 76'd3689348931447429070390;
2729 12'd2717: quad_corners = 76'd3689348931447965941303;
2730 12'd2718: quad_corners = 76'd3689348931723649154615;
2731 12'd2719: quad_corners = 76'd3689348931724186025528;
2732 12'd2720: quad_corners = 76'd3689348931999600803386;
2733 12'd2721: quad_corners = 76'd3763135908569853787707;
2734 12'd2722: quad_corners = 76'd3763135908570390658620;
2735 12'd2723: quad_corners = 76'd3763135908845537001022;
2736 12'd2724: quad_corners = 76'd3836922885415789985343;
2737 12'd2725: quad_corners = 76'd3837067000879280619073;
2738 12'd2726: quad_corners = 76'd3911142207825416879683;
2739 12'd2727: quad_corners = 76'd3985361529685019524677;
2740 12'd2728: quad_corners = 76'd3985649760336317578823;
2741 12'd2729: quad_corners = 76'd4059869082470529695305;
2742 12'd2730: quad_corners = 76'd4134088404605010247243;
2743 12'd2731: quad_corners = 76'd4208163611551146507854;
2744 12'd2732: quad_corners = 76'd4282382933685358624336;
2745 12'd2733: quad_corners = 76'd4356746371007646596691;
2746 12'd2734: quad_corners = 76'd4430965693141858712662;
2747 12'd2735: quad_corners = 76'd4505185015275802393177;
2748 12'd2736: quad_corners = 76'd4579548734073067075676;
2749 12'd2737: quad_corners = 76'd4727699288427681609311;
2750 12'd2738: quad_corners = 76'd4802063007499555763810;
2751 12'd2739: quad_corners = 76'd4876426585559332090982;
2752 12'd2740: quad_corners = 76'd5024577280651166544489;
2753 12'd2741: quad_corners = 76'd5024577280651166544489;
2754 12'd2742: quad_corners = 76'd5024577280651166544489;
2755 12'd2743: quad_corners = 76'd5024577280651166544489;
2756 12'd2744: quad_corners = 76'd5024577280651166544489;
2757 12'd2745: quad_corners = 76'd5024577280651166544489;
2758 12'd2746: quad_corners = 76'd5024577280651166544489;

2759 12'd2747: quad_corners = 76'd5024577280651166544489;
2760 12'd2748: quad_corners = 76'd5024577280651166544489;
2761 12'd2749: quad_corners = 76'd5024577280651166544489;
2762 12'd2750: quad_corners = 76'd5024577280651166544489;
2763 12'd2751: quad_corners = 76'd5024577280651166544489;
2764 12'd2752: quad_corners = 76'd4132070927201141063228;
2765 12'd2753: quad_corners = 76'd4132070927201141063228;
2766 12'd2754: quad_corners = 76'd4132070927201141063228;
2767 12'd2755: quad_corners = 76'd4132070927201141063228;
2768 12'd2756: quad_corners = 76'd4132070927201141063228;
2769 12'd2757: quad_corners = 76'd4132070927201141063228;
2770 12'd2758: quad_corners = 76'd4132070927201141063228;
2771 12'd2759: quad_corners = 76'd4132070927201141063228;
2772 12'd2760: quad_corners = 76'd4132070927201141063228;
2773 12'd2761: quad_corners = 76'd4132070927201141063228;
2774 12'd2762: quad_corners = 76'd4132070927201141063228;
2775 12'd2763: quad_corners = 76'd4132070927201141063228;
2776 12'd2764: quad_corners = 76'd4132070927201141063228;
2777 12'd2765: quad_corners = 76'd4132070927201141063228;
2778 12'd2766: quad_corners = 76'd4132070927201141063228;
2779 12'd2767: quad_corners = 76'd4132070927201141063228;
2780 12'd2768: quad_corners = 76'd4132070927201141063228;
2781 12'd2769: quad_corners = 76'd4132070927201141063228;
2782 12'd2770: quad_corners = 76'd4132070927201141063228;
2783 12'd2771: quad_corners = 76'd4132070927201141063228;
2784 12'd2772: quad_corners = 76'd4132070927201141063228;
2785 12'd2773: quad_corners = 76'd4132070927201141063228;
2786 12'd2774: quad_corners = 76'd4132070927201141063228;
2787 12'd2775: quad_corners = 76'd4132070927201141063228;
2788 12'd2776: quad_corners = 76'd4132070927201141063228;
2789 12'd2777: quad_corners = 76'd4132070786464458014268;
2790 12'd2778: quad_corners = 76'd4132070786739872792125;
2791 12'd2779: quad_corners = 76'd4132070786740409663037;
2792 12'd2780: quad_corners = 76'd4132070787016092876350;
2793 12'd2781: quad_corners = 76'd4132070787291507654206;
2794 12'd2782: quad_corners = 76'd4132070787292044525119;
2795 12'd2783: quad_corners = 76'd4132070787567459302976;
2796 12'd2784: quad_corners = 76'd4132070787842874080833;
2797 12'd2785: quad_corners = 76'd4132070787843410951746;
2798 12'd2786: quad_corners = 76'd4205857764413663936067;
2799 12'd2787: quad_corners = 76'd4205857764689078713925;
2800 12'd2788: quad_corners = 76'd4205857764964493491782;
2801 12'd2789: quad_corners = 76'd4279788856447675989575;
2802 12'd2790: quad_corners = 76'd4280077087099242479177;
2803 12'd2791: quad_corners = 76'd4354152294045378739787;
2804 12'd2792: quad_corners = 76'd4428227500991515000397;

2805 12'd2793: quad_corners = 76'd4428659846831157345871;
2806 12'd2794: quad_corners = 76'd4502735053777293606481;
2807 12'd2795: quad_corners = 76'd4576954375911505722963;
2808 12'd2796: quad_corners = 76'd4651029582857641983573;
2809 12'd2797: quad_corners = 76'd4651461928697015893592;
2810 12'd2798: quad_corners = 76'd4725681250831228009562;
2811 12'd2799: quad_corners = 76'd4800044828890735901277;
2812 12'd2800: quad_corners = 76'd4874264432499924728416;
2813 12'd2801: quad_corners = 76'd5022270871666463406178;
2814 12'd2802: quad_corners = 76'd5096634590463459653221;
2815 12'd2803: quad_corners = 76'd5170854194347526386792;
2816 12'd2804: quad_corners = 76'd5245217772407034278508;
2817 12'd2805: quad_corners = 76'd5245217772407034278508;
2818 12'd2806: quad_corners = 76'd5245217772407034278508;
2819 12'd2807: quad_corners = 76'd5245217772407034278508;
2820 12'd2808: quad_corners = 76'd5245217772407034278508;
2821 12'd2809: quad_corners = 76'd5245217772407034278508;
2822 12'd2810: quad_corners = 76'd5245217772407034278508;
2823 12'd2811: quad_corners = 76'd5245217772407034278508;
2824 12'd2812: quad_corners = 76'd5245217772407034278508;
2825 12'd2813: quad_corners = 76'd5245217772407034278508;
2826 12'd2814: quad_corners = 76'd5245217772407034278508;
2827 12'd2815: quad_corners = 76'd5245217772407034278508;
2828 12'd2816: quad_corners = 76'd4574792782769536433733;
2829 12'd2817: quad_corners = 76'd4574792782769536433733;
2830 12'd2818: quad_corners = 76'd4574792782769536433733;
2831 12'd2819: quad_corners = 76'd4574792782769536433733;
2832 12'd2820: quad_corners = 76'd4574792782769536433733;
2833 12'd2821: quad_corners = 76'd4574792782769536433733;
2834 12'd2822: quad_corners = 76'd4574792782769536433733;
2835 12'd2823: quad_corners = 76'd4574792782769536433733;
2836 12'd2824: quad_corners = 76'd4574792782769536433733;
2837 12'd2825: quad_corners = 76'd4574792782769536433733;
2838 12'd2826: quad_corners = 76'd4574792782769536433733;
2839 12'd2827: quad_corners = 76'd4574792782769536433733;
2840 12'd2828: quad_corners = 76'd4574792782769536433733;
2841 12'd2829: quad_corners = 76'd4574792782769536433733;
2842 12'd2830: quad_corners = 76'd4574792782769536433733;
2843 12'd2831: quad_corners = 76'd4574792782769536433733;
2844 12'd2832: quad_corners = 76'd4574792782769536433733;
2845 12'd2833: quad_corners = 76'd4574792782769536433733;
2846 12'd2834: quad_corners = 76'd4574792782769536433733;
2847 12'd2835: quad_corners = 76'd4574792782769536433733;
2848 12'd2836: quad_corners = 76'd4574792782769536433733;
2849 12'd2837: quad_corners = 76'd4574792782769536433733;
2850 12'd2838: quad_corners = 76'd4574792782769536433733;

2851 12'd2839: quad_corners = 76'd4574792782769536433733;
2852 12'd2840: quad_corners = 76'd4574792782769536433733;
2853 12'd2841: quad_corners = 76'd4574792642307731291717;
2854 12'd2842: quad_corners = 76'd4574792642308268162629;
2855 12'd2843: quad_corners = 76'd4574792642583951375942;
2856 12'd2844: quad_corners = 76'd4574792642584488246854;
2857 12'd2845: quad_corners = 76'd4574792642859903024711;
2858 12'd2846: quad_corners = 76'd4574792643135586238023;
2859 12'd2847: quad_corners = 76'd4574792643136123108936;
2860 12'd2848: quad_corners = 76'd4574792643411537886793;
2861 12'd2849: quad_corners = 76'd4574792643686684229194;
2862 12'd2850: quad_corners = 76'd4648579619982059306571;
2863 12'd2851: quad_corners = 76'd4648579620257474084428;
2864 12'd2852: quad_corners = 76'd4648579620532888862285;
2865 12'd2853: quad_corners = 76'd4722510712290949267022;
2866 12'd2854: quad_corners = 76'd4722798942942515756624;
2867 12'd2855: quad_corners = 76'd4723087173318935903825;
2868 12'd2856: quad_corners = 76'd4797162380265072164435;
2869 12'd2857: quad_corners = 76'd4797450610916370218581;
2870 12'd2858: quad_corners = 76'd4871525817862774914646;
2871 12'd2859: quad_corners = 76'd4945601024808911175256;
2872 12'd2860: quad_corners = 76'd4946033370648285085274;
2873 12'd2861: quad_corners = 76'd5020108577594152910428;
2874 12'd2862: quad_corners = 76'd5094328181203341737055;
2875 12'd2863: quad_corners = 76'd5168547644075042208353;
2876 12'd2864: quad_corners = 76'd5242767106946474244195;
2877 12'd2865: quad_corners = 76'd5316986710555663070822;
2878 12'd2866: quad_corners = 76'd5391206173701973013608;
2879 12'd2867: quad_corners = 76'd5465425777311161840235;
2880 12'd2868: quad_corners = 76'd5539789496108158087278;
2881 12'd2869: quad_corners = 76'd5539789496108158087278;
2882 12'd2870: quad_corners = 76'd5539789496108158087278;
2883 12'd2871: quad_corners = 76'd5539789496108158087278;
2884 12'd2872: quad_corners = 76'd5539789496108158087278;
2885 12'd2873: quad_corners = 76'd5539789496108158087278;
2886 12'd2874: quad_corners = 76'd5539789496108158087278;
2887 12'd2875: quad_corners = 76'd5539789496108158087278;
2888 12'd2876: quad_corners = 76'd5539789496108158087278;
2889 12'd2877: quad_corners = 76'd5539789496108158087278;
2890 12'd2878: quad_corners = 76'd5539789496108158087278;
2891 12'd2879: quad_corners = 76'd5539789496108158087278;
2892 12'd2880: quad_corners = 76'd5091301614633037921870;
2893 12'd2881: quad_corners = 76'd5091301614633037921870;
2894 12'd2882: quad_corners = 76'd5091301614633037921870;
2895 12'd2883: quad_corners = 76'd5091301614633037921870;
2896 12'd2884: quad_corners = 76'd5091301614633037921870;

2897 12'd2885: quad_corners = 76'd5091301614633037921870;
2898 12'd2886: quad_corners = 76'd5091301614633037921870;
2899 12'd2887: quad_corners = 76'd5091301614633037921870;
2900 12'd2888: quad_corners = 76'd5091301614633037921870;
2901 12'd2889: quad_corners = 76'd5091301614633037921870;
2902 12'd2890: quad_corners = 76'd5091301614633037921870;
2903 12'd2891: quad_corners = 76'd5091301614633037921870;
2904 12'd2892: quad_corners = 76'd5091301614633037921870;
2905 12'd2893: quad_corners = 76'd5091301614633037921870;
2906 12'd2894: quad_corners = 76'd5091301614633037921870;
2907 12'd2895: quad_corners = 76'd5091301614633037921870;
2908 12'd2896: quad_corners = 76'd5091301614633037921870;
2909 12'd2897: quad_corners = 76'd5091301614633037921870;
2910 12'd2898: quad_corners = 76'd5091301614633037921870;
2911 12'd2899: quad_corners = 76'd5091301614633037921870;
2912 12'd2900: quad_corners = 76'd5091301614633037921870;
2913 12'd2901: quad_corners = 76'd5091301614633037921870;
2914 12'd2902: quad_corners = 76'd5091301614633037921870;
2915 12'd2903: quad_corners = 76'd5091301614633037921870;
2916 12'd2904: quad_corners = 76'd5091301614633037921870;
2917 12'd2905: quad_corners = 76'd5091301473896086961742;
2918 12'd2906: quad_corners = 76'd5091301474171770175054;
2919 12'd2907: quad_corners = 76'd5091301474172307045966;
2920 12'd2908: quad_corners = 76'd50913014744472721823823;
2921 12'd2909: quad_corners = 76'd5091301474723405037135;
2922 12'd2910: quad_corners = 76'd5091301474723941908048;
2923 12'd2911: quad_corners = 76'd5091301474999356685904;
2924 12'd2912: quad_corners = 76'd5091301475274771463761;
2925 12'd2913: quad_corners = 76'd5091301475275039899218;
2926 12'd2914: quad_corners = 76'd5091301475550454677074;
2927 12'd2915: quad_corners = 76'd5091301475825869454931;
2928 12'd2916: quad_corners = 76'd5091301476101015797332;
2929 12'd2917: quad_corners = 76'd5165232567859344637525;
2930 12'd2918: quad_corners = 76'd5165520798235764784727;
2931 12'd2919: quad_corners = 76'd5165664913698986983000;
2932 12'd2920: quad_corners = 76'd5239740120645391679065;
2933 12'd2921: quad_corners = 76'd5240028351296689733211;
2934 12'd2922: quad_corners = 76'd5240316581947987787356;
2935 12'd2923: quad_corners = 76'd5314391788894124047966;
2936 12'd2924: quad_corners = 76'd5314680160282910457440;
2937 12'd2925: quad_corners = 76'd5388755507966266637409;
2938 12'd2926: quad_corners = 76'd5462830855649891252835;
2939 12'd2927: quad_corners = 76'd5463263342226753517669;
2940 12'd2928: quad_corners = 76'd5537338830647598053479;
2941 12'd2929: quad_corners = 76'd5611558293519298524777;
2942 12'd2930: quad_corners = 76'd5611990920833380709483;

2943 12'd2931: quad_corners = 76'd5686210524442301100654;
2944 12'd2932: quad_corners = 76'd5760430128051221491824;
2945 12'd2933: quad_corners = 76'd5760430128051221491824;
2946 12'd2934: quad_corners = 76'd5760430128051221491824;
2947 12'd2935: quad_corners = 76'd5760430128051221491824;
2948 12'd2936: quad_corners = 76'd5760430128051221491824;
2949 12'd2937: quad_corners = 76'd5760430128051221491824;
2950 12'd2938: quad_corners = 76'd5760430128051221491824;
2951 12'd2939: quad_corners = 76'd5760430128051221491824;
2952 12'd2940: quad_corners = 76'd5760430128051221491824;
2953 12'd2941: quad_corners = 76'd5760430128051221491824;
2954 12'd2942: quad_corners = 76'd5760430128051221491824;
2955 12'd2943: quad_corners = 76'd5760430128051221491824;
2956 12'd2944: quad_corners = 76'd5607810446221392019032;
2957 12'd2945: quad_corners = 76'd5607810446221392019032;
2958 12'd2946: quad_corners = 76'd5607810446221392019032;
2959 12'd2947: quad_corners = 76'd5607810446221392019032;
2960 12'd2948: quad_corners = 76'd5607810446221392019032;
2961 12'd2949: quad_corners = 76'd5607810446221392019032;
2962 12'd2950: quad_corners = 76'd5607810446221392019032;
2963 12'd2951: quad_corners = 76'd5607810446221392019032;
2964 12'd2952: quad_corners = 76'd5607810446221392019032;
2965 12'd2953: quad_corners = 76'd5607810446221392019032;
2966 12'd2954: quad_corners = 76'd5607810446221392019032;
2967 12'd2955: quad_corners = 76'd5607810446221392019032;
2968 12'd2956: quad_corners = 76'd5607810446221392019032;
2969 12'd2957: quad_corners = 76'd5607810446221392019032;
2970 12'd2958: quad_corners = 76'd5607810446221392019032;
2971 12'd2959: quad_corners = 76'd5607810446221392019032;
2972 12'd2960: quad_corners = 76'd5607810446221392019032;
2973 12'd2961: quad_corners = 76'd5607810446221392019032;
2974 12'd2962: quad_corners = 76'd5607810446221392019032;
2975 12'd2963: quad_corners = 76'd5607810446221392019032;
2976 12'd2964: quad_corners = 76'd5607810446221392019032;
2977 12'd2965: quad_corners = 76'd5607810446221392019032;
2978 12'd2966: quad_corners = 76'd5607810446221392019032;
2979 12'd2967: quad_corners = 76'd5607810446221392019032;
2980 12'd2968: quad_corners = 76'd5607810446221392019032;
2981 12'd2969: quad_corners = 76'd5607810446496807321176;
2982 12'd2970: quad_corners = 76'd5607810305760124796504;
2983 12'd2971: quad_corners = 76'd5607810306035539574360;
2984 12'd2972: quad_corners = 76'd5607810306310954876504;
2985 12'd2973: quad_corners = 76'd5534023330016654065240;
2986 12'd2974: quad_corners = 76'd5534023330292068843097;
2987 12'd2975: quad_corners = 76'd5534023330292605714009;
2988 12'd2976: quad_corners = 76'd5534023330568020491865;

2989 12'd2977: quad_corners = 76'd5534023330843435269722;
2990 12'd2978: quad_corners = 76'd5534023331118850047579;
2991 12'd2979: quad_corners = 76'd5534023331119118483035;
2992 12'd2980: quad_corners = 76'd5607954422877447323228;
2993 12'd2981: quad_corners = 76'd5608098538340669521501;
2994 12'd2982: quad_corners = 76'd5608242653803891719774;
2995 12'd2983: quad_corners = 76'd5608530884455458209375;
2996 12'd2984: quad_corners = 76'd5608674999643802500704;
2997 12'd2985: quad_corners = 76'd5682606091401862905441;
2998 12'd2986: quad_corners = 76'd5682894322053160959586;
2999 12'd2987: quad_corners = 76'd5683182693441678933092;
3000 12'd2988: quad_corners = 76'd5757113925937227693157;
3001 12'd2989: quad_corners = 76'd5757402438063502457446;
3002 12'd2990: quad_corners = 76'd5831477785747127072872;
3003 12'd2991: quad_corners = 76'd5831766157135645046378;
3004 12'd2992: quad_corners = 76'd5832054669261651375211;
3005 12'd2993: quad_corners = 76'd5906130016945275990637;
3006 12'd2994: quad_corners = 76'd5980349620554196381807;
3007 12'd2995: quad_corners = 76'd5980638132680202710641;
3008 12'd2996: quad_corners = 76'd6054857595551634746483;
3009 12'd2997: quad_corners = 76'd6054857595551634746483;
3010 12'd2998: quad_corners = 76'd6054857595551634746483;
3011 12'd2999: quad_corners = 76'd6054857595551634746483;
3012 12'd3000: quad_corners = 76'd6054857595551634746483;
3013 12'd3001: quad_corners = 76'd6054857595551634746483;
3014 12'd3002: quad_corners = 76'd6054857595551634746483;
3015 12'd3003: quad_corners = 76'd6054857595551634746483;
3016 12'd3004: quad_corners = 76'd6054857595551634746483;
3017 12'd3005: quad_corners = 76'd6054857595551634746483;
3018 12'd3006: quad_corners = 76'd6054857595551634746483;
3019 12'd3007: quad_corners = 76'd6054857595551634746483;
3020 12'd3008: quad_corners = 76'd6198106394842071628898;
3021 12'd3009: quad_corners = 76'd6198106394842071628898;
3022 12'd3010: quad_corners = 76'd6198106394842071628898;
3023 12'd3011: quad_corners = 76'd6198106394842071628898;
3024 12'd3012: quad_corners = 76'd6198106394842071628898;
3025 12'd3013: quad_corners = 76'd6198106394842071628898;
3026 12'd3014: quad_corners = 76'd6198106394842071628898;
3027 12'd3015: quad_corners = 76'd6198106394842071628898;
3028 12'd3016: quad_corners = 76'd6198106394842071628898;
3029 12'd3017: quad_corners = 76'd6198106394842071628898;
3030 12'd3018: quad_corners = 76'd6198106394842071628898;
3031 12'd3019: quad_corners = 76'd6198106394842071628898;
3032 12'd3020: quad_corners = 76'd6198106394842071628898;
3033 12'd3021: quad_corners = 76'd6198106394842071628898;
3034 12'd3022: quad_corners = 76'd6198106394842071628898;

3035 12'd3023: quad_corners = 76'd6198106394842071628898;
3036 12'd3024: quad_corners = 76'd6198106394842071628898;
3037 12'd3025: quad_corners = 76'd6198106394842071628898;
3038 12'd3026: quad_corners = 76'd6198106394842071628898;
3039 12'd3027: quad_corners = 76'd6198106394842071628898;
3040 12'd3028: quad_corners = 76'd6198106394842071628898;
3041 12'd3029: quad_corners = 76'd6198106394842071628898;
3042 12'd3030: quad_corners = 76'd6198106394842071628898;
3043 12'd3031: quad_corners = 76'd6198106394842071628898;
3044 12'd3032: quad_corners = 76'd6198106394842071628898;
3045 12'd3033: quad_corners = 76'd6124319418822648724578;
3046 12'd3034: quad_corners = 76'd6124319278085966724706;
3047 12'd3035: quad_corners = 76'd6124319137623893671522;
3048 12'd3036: quad_corners = 76'd6124319137899308973666;
3049 12'd3037: quad_corners = 76'd6124319137899846368866;
3050 12'd3038: quad_corners = 76'd6124319138175261671010;
3051 12'd3039: quad_corners = 76'd6050532162155838766690;
3052 12'd3040: quad_corners = 76'd6050532162156376161890;
3053 12'd3041: quad_corners = 76'd6050676277619598360163;
3054 12'd3042: quad_corners = 76'd6050676277895013138019;
3055 12'd3043: quad_corners = 76'd6050820393083357429348;
3056 12'd3044: quad_corners = 76'd6050820393358772207204;
3057 12'd3045: quad_corners = 76'd6050964508821994405477;
3058 12'd3046: quad_corners = 76'd6051108624285216603749;
3059 12'd3047: quad_corners = 76'd6051252739473560895078;
3060 12'd3048: quad_corners = 76'd6051396995674271448167;
3061 12'd3049: quad_corners = 76'd6125328228169820208232;
3062 12'd3050: quad_corners = 76'd6125472343633042405993;
3063 12'd3051: quad_corners = 76'd6125616599833752959594;
3064 12'd3052: quad_corners = 76'd6125904971222270933099;
3065 12'd3053: quad_corners = 76'd6126049368160469841516;
3066 12'd3054: quad_corners = 76'd6200124715568948115053;
3067 12'd3055: quad_corners = 76'd6200268971769658668142;
3068 12'd3056: quad_corners = 76'd6200557483895664996975;
3069 12'd3057: quad_corners = 76'd6200845855284182970481;
3070 12'd3058: quad_corners = 76'd6274777228516951649906;
3071 12'd3059: quad_corners = 76'd6275065740642957978739;
3072 12'd3060: quad_corners = 76'd6349141088326314158709;
3073 12'd3061: quad_corners = 76'd6349141088326314158709;
3074 12'd3062: quad_corners = 76'd6349141088326314158709;
3075 12'd3063: quad_corners = 76'd6349141088326314158709;
3076 12'd3064: quad_corners = 76'd6349141088326314158709;
3077 12'd3065: quad_corners = 76'd6349141088326314158709;
3078 12'd3066: quad_corners = 76'd6349141088326314158709;
3079 12'd3067: quad_corners = 76'd6349141088326314158709;
3080 12'd3068: quad_corners = 76'd6349141088326314158709;

3081 12'd3069: quad_corners = 76'd6349141088326314158709;
3082 12'd3070: quad_corners = 76'd6349141088326314158709;
3083 12'd3071: quad_corners = 76'd6349141088326314158709;
3084 12'd3072: quad_corners = 76'd6715336083845781190252;
3085 12'd3073: quad_corners = 76'd6715336083845781190252;
3086 12'd3074: quad_corners = 76'd6715336083845781190252;
3087 12'd3075: quad_corners = 76'd6715336083845781190252;
3088 12'd3076: quad_corners = 76'd6715336083845781190252;
3089 12'd3077: quad_corners = 76'd6715336083845781190252;
3090 12'd3078: quad_corners = 76'd6715336083845781190252;
3091 12'd3079: quad_corners = 76'd6715336083845781190252;
3092 12'd3080: quad_corners = 76'd6715336083845781190252;
3093 12'd3081: quad_corners = 76'd6715336083845781190252;
3094 12'd3082: quad_corners = 76'd6715336083845781190252;
3095 12'd3083: quad_corners = 76'd6715336083845781190252;
3096 12'd3084: quad_corners = 76'd6715336083845781190252;
3097 12'd3085: quad_corners = 76'd6715336083845781190252;
3098 12'd3086: quad_corners = 76'd6715336083845781190252;
3099 12'd3087: quad_corners = 76'd6715336083845781190252;
3100 12'd3088: quad_corners = 76'd6715336083845781190252;
3101 12'd3089: quad_corners = 76'd6715336083845781190252;
3102 12'd3090: quad_corners = 76'd6715336083845781190252;
3103 12'd3091: quad_corners = 76'd6715336083845781190252;
3104 12'd3092: quad_corners = 76'd6715336083845781190252;
3105 12'd3093: quad_corners = 76'd6715336083845781190252;
3106 12'd3094: quad_corners = 76'd6715336083845781190252;
3107 12'd3095: quad_corners = 76'd6715336083845781190252;
3108 12'd3096: quad_corners = 76'd6715336083845781190252;
3109 12'd3097: quad_corners = 76'd6715191828195632281708;
3110 12'd3098: quad_corners = 76'd6715191687458681321580;
3111 12'd3099: quad_corners = 76'd6715191687734096624236;
3112 12'd3100: quad_corners = 76'd6641260455514231601772;
3113 12'd3101: quad_corners = 76'd6641260455789646903916;
3114 12'd3102: quad_corners = 76'd6641260456065062730860;
3115 12'd3103: quad_corners = 76'd6641260315328111770732;
3116 12'd3104: quad_corners = 76'd6641260315603527072876;
3117 12'd3105: quad_corners = 76'd6567473339583835733100;
3118 12'd3106: quad_corners = 76'd6567617454772448984172;
3119 12'd3107: quad_corners = 76'd6567617595785084206188;
3120 12'd3108: quad_corners = 76'd6567617596060231072877;
3121 12'd3109: quad_corners = 76'd6567761711248575364205;
3122 12'd3110: quad_corners = 76'd6567761852261478496877;
3123 12'd3111: quad_corners = 76'd6567905967724432259694;
3124 12'd3112: quad_corners = 76'd6567906108737066956910;
3125 12'd3113: quad_corners = 76'd6568050364662899603567;
3126 12'd3114: quad_corners = 76'd6568194620863610156655;

3127 12'd3115: quad_corners = 76'd6568194621138488063600;
3128 12'd3116: quad_corners = 76'd6568339018076686972017;
3129 12'd3117: quad_corners = 76'd6568483274277129089649;
3130 12'd3118: quad_corners = 76'd6568627530202961736306;
3131 12'd3119: quad_corners = 76'd6568771786403403853939;
3132 12'd3120: quad_corners = 76'd6569060298529410182772;
3133 12'd3121: quad_corners = 76'd6569204554729852300404;
3134 12'd3122: quad_corners = 76'd6569348951667782773365;
3135 12'd3123: quad_corners = 76'd6569493348605444810358;
3136 12'd3124: quad_corners = 76'd6569781719993962783863;
3137 12'd3125: quad_corners = 76'd6569781719993962783863;
3138 12'd3126: quad_corners = 76'd6569781719993962783863;
3139 12'd3127: quad_corners = 76'd6569781719993962783863;
3140 12'd3128: quad_corners = 76'd6569781719993962783863;
3141 12'd3129: quad_corners = 76'd6569781719993962783863;
3142 12'd3130: quad_corners = 76'd6569781719993962783863;
3143 12'd3131: quad_corners = 76'd6569781719993962783863;
3144 12'd3132: quad_corners = 76'd6569781719993962783863;
3145 12'd3133: quad_corners = 76'd6569781719993962783863;
3146 12'd3134: quad_corners = 76'd6569781719993962783863;
3147 12'd3135: quad_corners = 76'd6569781719993962783863;
3148 12'd3136: quad_corners = 76'd7306352608131693211767;
3149 12'd3137: quad_corners = 76'd7306352608131693211767;
3150 12'd3138: quad_corners = 76'd7306352608131693211767;
3151 12'd3139: quad_corners = 76'd7306352608131693211767;
3152 12'd3140: quad_corners = 76'd7306352608131693211767;
3153 12'd3141: quad_corners = 76'd7306352608131693211767;
3154 12'd3142: quad_corners = 76'd7306352608131693211767;
3155 12'd3143: quad_corners = 76'd7306352608131693211767;
3156 12'd3144: quad_corners = 76'd7306352608131693211767;
3157 12'd3145: quad_corners = 76'd7306352608131693211767;
3158 12'd3146: quad_corners = 76'd7306352608131693211767;
3159 12'd3147: quad_corners = 76'd7306352608131693211767;
3160 12'd3148: quad_corners = 76'd7306352608131693211767;
3161 12'd3149: quad_corners = 76'd7306352608131693211767;
3162 12'd3150: quad_corners = 76'd7306352608131693211767;
3163 12'd3151: quad_corners = 76'd7306352608131693211767;
3164 12'd3152: quad_corners = 76'd7306352608131693211767;
3165 12'd3153: quad_corners = 76'd7306352608131693211767;
3166 12'd3154: quad_corners = 76'd7306352608131693211767;
3167 12'd3155: quad_corners = 76'd7306352608131693211767;
3168 12'd3156: quad_corners = 76'd7306352608131693211767;
3169 12'd3157: quad_corners = 76'd7306352608131693211767;
3170 12'd3158: quad_corners = 76'd7306352608131693211767;
3171 12'd3159: quad_corners = 76'd7306352608131693211767;
3172 12'd3160: quad_corners = 76'd7306352608131693211767;

```

3173      12'd3161: quad_corners = 76'd7306208493219301618295;
3174      12'd3162: quad_corners = 76'd7306208352482350658167;
3175      12'd3163: quad_corners = 76'd7306064237569690104950;
3176      12'd3164: quad_corners = 76'd7232277120812778845302;
3177      12'd3165: quad_corners = 76'd7232133005625240909430;
3178      12'd3166: quad_corners = 76'd7232133005900387776118;
3179      12'd3167: quad_corners = 76'd7158346029606086964854;
3180      12'd3168: quad_corners = 76'd7158345889143746000502;
3181      12'd3169: quad_corners = 76'd7158201774231085447286;
3182      12'd3170: quad_corners = 76'd7158201914968842762357;
3183      12'd3171: quad_corners = 76'd7158201915243989629045;
3184      12'd3172: quad_corners = 76'd7084414939224567248501;
3185      12'd3173: quad_corners = 76'd7084415079962324563574;
3186      12'd3174: quad_corners = 76'd7084415080237202994806;
3187      12'd3175: quad_corners = 76'd7084415221249838216822;
3188      12'd3176: quad_corners = 76'd7010772360143344301174;
3189      12'd3177: quad_corners = 76'd7010772501155978998902;
3190      12'd3178: quad_corners = 76'd7010772642168345260662;
3191      12'd3179: quad_corners = 76'd7010916898094177907318;
3192      12'd3180: quad_corners = 76'd7010917039106544169079;
3193      12'd3181: quad_corners = 76'd6937130203824072224375;
3194      12'd3182: quad_corners = 76'd6937274460024514342007;
3195      12'd3183: quad_corners = 76'd6937418856687566908024;
3196      12'd3184: quad_corners = 76'd6937418997699933169784;
3197      12'd3185: quad_corners = 76'd6937563394637863642744;
3198      12'd3186: quad_corners = 76'd6937707650838305760377;
3199      12'd3187: quad_corners = 76'd6937707932313014035065;
3200      12'd3188: quad_corners = 76'd6864065352956106301050;
3201      12'd3189: quad_corners = 76'd6864065352956106301050;
3202      12'd3190: quad_corners = 76'd6864065352956106301050;
3203      12'd3191: quad_corners = 76'd6864065352956106301050;
3204      12'd3192: quad_corners = 76'd6864065352956106301050;
3205      12'd3193: quad_corners = 76'd6864065352956106301050;
3206      12'd3194: quad_corners = 76'd6864065352956106301050;
3207      12'd3195: quad_corners = 76'd6864065352956106301050;
3208      12'd3196: quad_corners = 76'd6864065352956106301050;
3209      12'd3197: quad_corners = 76'd6864065352956106301050;
3210      12'd3198: quad_corners = 76'd6864065352956106301050;
3211      12'd3199: quad_corners = 76'd6864065352956106301050;
3212      12'd3200: quad_corners = 76'd7971300364638276086403;
3213      12'd3201: quad_corners = 76'd7971300364638276086403;
3214      12'd3202: quad_corners = 76'd7971300364638276086403;
3215      12'd3203: quad_corners = 76'd7971300364638276086403;
3216      12'd3204: quad_corners = 76'd7971300364638276086403;
3217      12'd3205: quad_corners = 76'd7971300364638276086403;
3218      12'd3206: quad_corners = 76'd7971300364638276086403;

```

3219 12'd3207: quad_corners = 76'd7971300364638276086403;
3220 12'd3208: quad_corners = 76'd7971300364638276086403;
3221 12'd3209: quad_corners = 76'd7971300364638276086403;
3222 12'd3210: quad_corners = 76'd7971300364638276086403;
3223 12'd3211: quad_corners = 76'd7971300364638276086403;
3224 12'd3212: quad_corners = 76'd7971300364638276086403;
3225 12'd3213: quad_corners = 76'd7971300364638276086403;
3226 12'd3214: quad_corners = 76'd7971300364638276086403;
3227 12'd3215: quad_corners = 76'd7971300364638276086403;
3228 12'd3216: quad_corners = 76'd7971300364638276086403;
3229 12'd3217: quad_corners = 76'd7971300364638276086403;
3230 12'd3218: quad_corners = 76'd7971300364638276086403;
3231 12'd3219: quad_corners = 76'd7971300364638276086403;
3232 12'd3220: quad_corners = 76'd7971300364638276086403;
3233 12'd3221: quad_corners = 76'd7971300364638276086403;
3234 12'd3222: quad_corners = 76'd7971300364638276086403;
3235 12'd3223: quad_corners = 76'd7971300364638276086403;
3236 12'd3224: quad_corners = 76'd7971300364638276086403;
3237 12'd3225: quad_corners = 76'd7897369273430777850498;
3238 12'd3226: quad_corners = 76'd7897225017505751035010;
3239 12'd3227: quad_corners = 76'd7897225017781166861953;
3240 12'd3228: quad_corners = 76'd7897080902868506308225;
3241 12'd3229: quad_corners = 76'd7823149670648641286273;
3242 12'd3230: quad_corners = 76'd7823149670923788677248;
3243 12'd3231: quad_corners = 76'd7749218579441412010112;
3244 12'd3232: quad_corners = 76'd7749074464528483545728;
3245 12'd3233: quad_corners = 76'd7749074464528752505471;
3246 12'd3234: quad_corners = 76'd7675143373320985309823;
3247 12'd3235: quad_corners = 76'd7675143373596401136255;
3248 12'd3236: quad_corners = 76'd7675143514333890015871;
3249 12'd3237: quad_corners = 76'd7601212423126122820222;
3250 12'd3238: quad_corners = 76'd7601212563863880659070;
3251 12'd3239: quad_corners = 76'd7601212564139027525758;
3252 12'd3240: quad_corners = 76'd7527281613668480774270;
3253 12'd3241: quad_corners = 76'd7527281754406238088830;
3254 12'd3242: quad_corners = 76'd7527281895418604874877;
3255 12'd3243: quad_corners = 76'd7453495060136132930685;
3256 12'd3244: quad_corners = 76'd7453495200873621285501;
3257 12'd3245: quad_corners = 76'd7379708365591149340797;
3258 12'd3246: quad_corners = 76'd7379708506328637695613;
3259 12'd3247: quad_corners = 76'd7379708647341003957373;
3260 12'd3248: quad_corners = 76'd7305921812058532012669;
3261 12'd3249: quad_corners = 76'd7305922093533240287356;
3262 12'd3250: quad_corners = 76'd7232135258250768342652;
3263 12'd3251: quad_corners = 76'd7232135540000354524284;
3264 12'd3252: quad_corners = 76'd7232135821475062798460;

```

3265      12'd3253: quad_corners = 76'd7232135821475062798460;
3266      12'd3254: quad_corners = 76'd7232135821475062798460;
3267      12'd3255: quad_corners = 76'd7232135821475062798460;
3268      12'd3256: quad_corners = 76'd7232135821475062798460;
3269      12'd3257: quad_corners = 76'd7232135821475062798460;
3270      12'd3258: quad_corners = 76'd7232135821475062798460;
3271      12'd3259: quad_corners = 76'd7232135821475062798460;
3272      12'd3260: quad_corners = 76'd7232135821475062798460;
3273      12'd3261: quad_corners = 76'd7232135821475062798460;
3274      12'd3262: quad_corners = 76'd7232135821475062798460;
3275      12'd3263: quad_corners = 76'd7232135821475062798460;
3276      12'd3264: quad_corners = 76'd8562461285587240673934;
3277      12'd3265: quad_corners = 76'd8562461285587240673934;
3278      12'd3266: quad_corners = 76'd8562461285587240673934;
3279      12'd3267: quad_corners = 76'd8562461285587240673934;
3280      12'd3268: quad_corners = 76'd8562461285587240673934;
3281      12'd3269: quad_corners = 76'd8562461285587240673934;
3282      12'd3270: quad_corners = 76'd8562461285587240673934;
3283      12'd3271: quad_corners = 76'd8562461285587240673934;
3284      12'd3272: quad_corners = 76'd8562461285587240673934;
3285      12'd3273: quad_corners = 76'd8562461285587240673934;
3286      12'd3274: quad_corners = 76'd8562461285587240673934;
3287      12'd3275: quad_corners = 76'd8562461285587240673934;
3288      12'd3276: quad_corners = 76'd8562461285587240673934;
3289      12'd3277: quad_corners = 76'd8562461285587240673934;
3290      12'd3278: quad_corners = 76'd8562461285587240673934;
3291      12'd3279: quad_corners = 76'd8562461285587240673934;
3292      12'd3280: quad_corners = 76'd8562461285587240673934;
3293      12'd3281: quad_corners = 76'd8562461285587240673934;
3294      12'd3282: quad_corners = 76'd8562461285587240673934;
3295      12'd3283: quad_corners = 76'd8562461285587240673934;
3296      12'd3284: quad_corners = 76'd8562461285587240673934;
3297      12'd3285: quad_corners = 76'd8562461285587240673934;
3298      12'd3286: quad_corners = 76'd8562461285587240673934;
3299      12'd3287: quad_corners = 76'd8562461285587240673934;
3300      12'd3288: quad_corners = 76'd8562461285587240673934;
3301      12'd3289: quad_corners = 76'd8562317029662214382734;
3302      12'd3290: quad_corners = 76'd8562172914749553829517;
3303      12'd3291: quad_corners = 76'd8488241682804567238285;
3304      12'd3292: quad_corners = 76'd8488097567617028778124;
3305      12'd3293: quad_corners = 76'd8487953452704100313228;
3306      12'd3294: quad_corners = 76'd8414022361221723646603;
3307      12'd3295: quad_corners = 76'd8414022361496871037579;
3308      12'd3296: quad_corners = 76'd8340091270014225935498;
3309      12'd3297: quad_corners = 76'd8339947155101297470602;
3310      12'd3298: quad_corners = 76'd8266016063618652368009;

```

3311 12'd3299: quad_corners = 76'd8265871948705723903113;
3312 12'd3300: quad_corners = 76'd8191940997960567155848;
3313 12'd3301: quad_corners = 76'd8191940998235714546824;
3314 12'd3302: quad_corners = 76'd8118010047765167270535;
3315 12'd3303: quad_corners = 76'd8117865932577360374406;
3316 12'd3304: quad_corners = 76'd8043934982106813622918;
3317 12'd3305: quad_corners = 76'd8043935122844302502021;
3318 12'd3306: quad_corners = 76'd7970004031636267394693;
3319 12'd3307: quad_corners = 76'd7969860057185680418436;
3320 12'd3308: quad_corners = 76'd7896073221903209522308;
3321 12'd3309: quad_corners = 76'd7822142271157784339075;
3322 12'd3310: quad_corners = 76'd7822142552907638956162;
3323 12'd3311: quad_corners = 76'd7748211602161944813186;
3324 12'd3312: quad_corners = 76'd7748067627986235219073;
3325 12'd3313: quad_corners = 76'd7674280933166105287297;
3326 12'd3314: quad_corners = 76'd7600349982695289050752;
3327 12'd3315: quad_corners = 76'd7526563287875159118975;
3328 12'd3316: quad_corners = 76'd7526419313699181088895;
3329 12'd3317: quad_corners = 76'd7526419313699181088895;
3330 12'd3318: quad_corners = 76'd7526419313699181088895;
3331 12'd3319: quad_corners = 76'd7526419313699181088895;
3332 12'd3320: quad_corners = 76'd7526419313699181088895;
3333 12'd3321: quad_corners = 76'd7526419313699181088895;
3334 12'd3322: quad_corners = 76'd7526419313699181088895;
3335 12'd3323: quad_corners = 76'd7526419313699181088895;
3336 12'd3324: quad_corners = 76'd7526419313699181088895;
3337 12'd3325: quad_corners = 76'd7526419313699181088895;
3338 12'd3326: quad_corners = 76'd7526419313699181088895;
3339 12'd3327: quad_corners = 76'd7526419313699181088895;
3340 12'd3328: quad_corners = 76'd9227553297744240892571;
3341 12'd3329: quad_corners = 76'd9227553297744240892571;
3342 12'd3330: quad_corners = 76'd9227553297744240892571;
3343 12'd3331: quad_corners = 76'd9227553297744240892571;
3344 12'd3332: quad_corners = 76'd9227553297744240892571;
3345 12'd3333: quad_corners = 76'd9227553297744240892571;
3346 12'd3334: quad_corners = 76'd9227553297744240892571;
3347 12'd3335: quad_corners = 76'd9227553297744240892571;
3348 12'd3336: quad_corners = 76'd9227553297744240892571;
3349 12'd3337: quad_corners = 76'd9227553297744240892571;
3350 12'd3338: quad_corners = 76'd9227553297744240892571;
3351 12'd3339: quad_corners = 76'd9227553297744240892571;
3352 12'd3340: quad_corners = 76'd9227553297744240892571;
3353 12'd3341: quad_corners = 76'd9227553297744240892571;
3354 12'd3342: quad_corners = 76'd9227553297744240892571;
3355 12'd3343: quad_corners = 76'd9227553297744240892571;
3356 12'd3344: quad_corners = 76'd9227553297744240892571;

3357 12'd3345: quad_corners = 76'd9227553297744240892571;
3358 12'd3346: quad_corners = 76'd9227553297744240892571;
3359 12'd3347: quad_corners = 76'd9227553297744240892571;
3360 12'd3348: quad_corners = 76'd9227553297744240892571;
3361 12'd3349: quad_corners = 76'd9227553297744240892571;
3362 12'd3350: quad_corners = 76'd9227553297744240892571;
3363 12'd3351: quad_corners = 76'd9227553297744240892571;
3364 12'd3352: quad_corners = 76'd9227553297744240892571;
3365 12'd3353: quad_corners = 76'd9227409042094092508314;
3366 12'd3354: quad_corners = 76'd9227264926906554048154;
3367 12'd3355: quad_corners = 76'd9153333694961567457433;
3368 12'd3356: quad_corners = 76'd9153045464585685229720;
3369 12'd3357: quad_corners = 76'd9079114373378186470040;
3370 12'd3358: quad_corners = 76'd9078970258190380098199;
3371 12'd3359: quad_corners = 76'd9005039166982613427350;
3372 12'd3360: quad_corners = 76'd9004750936606730675349;
3373 12'd3361: quad_corners = 76'd8930819845398964003988;
3374 12'd3362: quad_corners = 76'd8930675730211157107860;
3375 12'd3363: quad_corners = 76'd8856744779740878791827;
3376 12'd3364: quad_corners = 76'd8782669573069889922194;
3377 12'd3365: quad_corners = 76'd8782525458156960933009;
3378 12'd3366: quad_corners = 76'd8708594507411536274576;
3379 12'd3367: quad_corners = 76'd8634663416203500643471;
3380 12'd3368: quad_corners = 76'd8634519441752914190990;
3381 12'd3369: quad_corners = 76'd8560444375819413152397;
3382 12'd3370: quad_corners = 76'd8486513425348866400396;
3383 12'd3371: quad_corners = 76'd8412582333865952861835;
3384 12'd3372: quad_corners = 76'd8412438359690244316298;
3385 12'd3373: quad_corners = 76'd8338507408944551221897;
3386 12'd3374: quad_corners = 76'd8264432483748538538120;
3387 12'd3375: quad_corners = 76'd8190501533277723350663;
3388 12'd3376: quad_corners = 76'd8116570582532029731974;
3389 12'd3377: quad_corners = 76'd8042783746974411444869;
3390 12'd3378: quad_corners = 76'd7968997052429159419524;
3391 12'd3379: quad_corners = 76'd7895210216871541132419;
3392 12'd3380: quad_corners = 76'd7821423522051142764673;
3393 12'd3381: quad_corners = 76'd7821423522051142764673;
3394 12'd3382: quad_corners = 76'd7821423522051142764673;
3395 12'd3383: quad_corners = 76'd7821423522051142764673;
3396 12'd3384: quad_corners = 76'd7821423522051142764673;
3397 12'd3385: quad_corners = 76'd7821423522051142764673;
3398 12'd3386: quad_corners = 76'd7821423522051142764673;
3399 12'd3387: quad_corners = 76'd7821423522051142764673;
3400 12'd3388: quad_corners = 76'd7821423522051142764673;
3401 12'd3389: quad_corners = 76'd7821423522051142764673;
3402 12'd3390: quad_corners = 76'd7821423522051142764673;

3403 12'd3391: quad_corners = 76'd7821423522051142764673;
3404 12'd3392: quad_corners = 76'd9227553297744240892571;
3405 12'd3393: quad_corners = 76'd9227553297744240892571;
3406 12'd3394: quad_corners = 76'd9227553297744240892571;
3407 12'd3395: quad_corners = 76'd9227553297744240892571;
3408 12'd3396: quad_corners = 76'd9227553297744240892571;
3409 12'd3397: quad_corners = 76'd9227553297744240892571;
3410 12'd3398: quad_corners = 76'd9227553297744240892571;
3411 12'd3399: quad_corners = 76'd9227553297744240892571;
3412 12'd3400: quad_corners = 76'd9227553297744240892571;
3413 12'd3401: quad_corners = 76'd9227553297744240892571;
3414 12'd3402: quad_corners = 76'd9227553297744240892571;
3415 12'd3403: quad_corners = 76'd9227553297744240892571;
3416 12'd3404: quad_corners = 76'd9227553297744240892571;
3417 12'd3405: quad_corners = 76'd9227553297744240892571;
3418 12'd3406: quad_corners = 76'd9227553297744240892571;
3419 12'd3407: quad_corners = 76'd9227553297744240892571;
3420 12'd3408: quad_corners = 76'd9227553297744240892571;
3421 12'd3409: quad_corners = 76'd9227553297744240892571;
3422 12'd3410: quad_corners = 76'd9227553297744240892571;
3423 12'd3411: quad_corners = 76'd9227553297744240892571;
3424 12'd3412: quad_corners = 76'd9227553297744240892571;
3425 12'd3413: quad_corners = 76'd9227553297744240892571;
3426 12'd3414: quad_corners = 76'd9227553297744240892571;
3427 12'd3415: quad_corners = 76'd9227553297744240892571;
3428 12'd3416: quad_corners = 76'd9227553297744240892571;
3429 12'd3417: quad_corners = 76'd9227409042094092508314;
3430 12'd3418: quad_corners = 76'd9227264926906554048154;
3431 12'd3419: quad_corners = 76'd9153333694961567457433;
3432 12'd3420: quad_corners = 76'd9153045464585685229720;
3433 12'd3421: quad_corners = 76'd9079114373378186470040;
3434 12'd3422: quad_corners = 76'd9078970258190380098199;
3435 12'd3423: quad_corners = 76'd9005039166982613427350;
3436 12'd3424: quad_corners = 76'd9004750936606730675349;
3437 12'd3425: quad_corners = 76'd8930819845398964003988;
3438 12'd3426: quad_corners = 76'd8930675730211157107860;
3439 12'd3427: quad_corners = 76'd8856744779740878791827;
3440 12'd3428: quad_corners = 76'd8782669573069889922194;
3441 12'd3429: quad_corners = 76'd8782525458156960933009;
3442 12'd3430: quad_corners = 76'd8708594507411536274576;
3443 12'd3431: quad_corners = 76'd8634663416203500643471;
3444 12'd3432: quad_corners = 76'd8634519441752914190990;
3445 12'd3433: quad_corners = 76'd8560444375819413152397;
3446 12'd3434: quad_corners = 76'd8486513425348866400396;
3447 12'd3435: quad_corners = 76'd8412582333865952861835;
3448 12'd3436: quad_corners = 76'd8412438359690244316298;

```

3449      12'd3437: quad_corners = 76'd8338507408944551221897;
3450      12'd3438: quad_corners = 76'd8264432483748538538120;
3451      12'd3439: quad_corners = 76'd8190501533277723350663;
3452      12'd3440: quad_corners = 76'd8116570582532029731974;
3453      12'd3441: quad_corners = 76'd8042783746974411444869;
3454      12'd3442: quad_corners = 76'd7968997052429159419524;
3455      12'd3443: quad_corners = 76'd7895210216871541132419;
3456      12'd3444: quad_corners = 76'd7821423522051142764673;
3457      12'd3445: quad_corners = 76'd7821423522051142764673;
3458      12'd3446: quad_corners = 76'd7821423522051142764673;
3459      12'd3447: quad_corners = 76'd7821423522051142764673;
3460      12'd3448: quad_corners = 76'd7821423522051142764673;
3461      12'd3449: quad_corners = 76'd7821423522051142764673;
3462      12'd3450: quad_corners = 76'd7821423522051142764673;
3463      12'd3451: quad_corners = 76'd7821423522051142764673;
3464      12'd3452: quad_corners = 76'd7821423522051142764673;
3465      12'd3453: quad_corners = 76'd7821423522051142764673;
3466      12'd3454: quad_corners = 76'd7821423522051142764673;
3467      12'd3455: quad_corners = 76'd7821423522051142764673;
3468      12'd3456: quad_corners = 76'd9227553297744240892571;
3469      12'd3457: quad_corners = 76'd9227553297744240892571;
3470      12'd3458: quad_corners = 76'd9227553297744240892571;
3471      12'd3459: quad_corners = 76'd9227553297744240892571;
3472      12'd3460: quad_corners = 76'd9227553297744240892571;
3473      12'd3461: quad_corners = 76'd9227553297744240892571;
3474      12'd3462: quad_corners = 76'd9227553297744240892571;
3475      12'd3463: quad_corners = 76'd9227553297744240892571;
3476      12'd3464: quad_corners = 76'd9227553297744240892571;
3477      12'd3465: quad_corners = 76'd9227553297744240892571;
3478      12'd3466: quad_corners = 76'd9227553297744240892571;
3479      12'd3467: quad_corners = 76'd9227553297744240892571;
3480      12'd3468: quad_corners = 76'd9227553297744240892571;
3481      12'd3469: quad_corners = 76'd9227553297744240892571;
3482      12'd3470: quad_corners = 76'd9227553297744240892571;
3483      12'd3471: quad_corners = 76'd9227553297744240892571;
3484      12'd3472: quad_corners = 76'd9227553297744240892571;
3485      12'd3473: quad_corners = 76'd9227553297744240892571;
3486      12'd3474: quad_corners = 76'd9227553297744240892571;
3487      12'd3475: quad_corners = 76'd9227553297744240892571;
3488      12'd3476: quad_corners = 76'd9227553297744240892571;
3489      12'd3477: quad_corners = 76'd9227553297744240892571;
3490      12'd3478: quad_corners = 76'd9227553297744240892571;
3491      12'd3479: quad_corners = 76'd9227553297744240892571;
3492      12'd3480: quad_corners = 76'd9227553297744240892571;
3493      12'd3481: quad_corners = 76'd9227409042094092508314;
3494      12'd3482: quad_corners = 76'd9227264926906554048154;

```

3495 12'd3483: quad_corners = 76'd9153333694961567457433;
3496 12'd3484: quad_corners = 76'd9153045464585685229720;
3497 12'd3485: quad_corners = 76'd9079114373378186470040;
3498 12'd3486: quad_corners = 76'd9078970258190380098199;
3499 12'd3487: quad_corners = 76'd9005039166982613427350;
3500 12'd3488: quad_corners = 76'd9004750936606730675349;
3501 12'd3489: quad_corners = 76'd8930819845398964003988;
3502 12'd3490: quad_corners = 76'd8930675730211157107860;
3503 12'd3491: quad_corners = 76'd8856744779740878791827;
3504 12'd3492: quad_corners = 76'd8782669573069889922194;
3505 12'd3493: quad_corners = 76'd8782525458156960933009;
3506 12'd3494: quad_corners = 76'd8708594507411536274576;
3507 12'd3495: quad_corners = 76'd8634663416203500643471;
3508 12'd3496: quad_corners = 76'd8634519441752914190990;
3509 12'd3497: quad_corners = 76'd8560444375819413152397;
3510 12'd3498: quad_corners = 76'd8486513425348866400396;
3511 12'd3499: quad_corners = 76'd8412582333865952861835;
3512 12'd3500: quad_corners = 76'd8412438359690244316298;
3513 12'd3501: quad_corners = 76'd8338507408944551221897;
3514 12'd3502: quad_corners = 76'd8264432483748538538120;
3515 12'd3503: quad_corners = 76'd8190501533277723350663;
3516 12'd3504: quad_corners = 76'd8116570582532029731974;
3517 12'd3505: quad_corners = 76'd8042783746974411444869;
3518 12'd3506: quad_corners = 76'd7968997052429159419524;
3519 12'd3507: quad_corners = 76'd7895210216871541132419;
3520 12'd3508: quad_corners = 76'd7821423522051142764673;
3521 12'd3509: quad_corners = 76'd7821423522051142764673;
3522 12'd3510: quad_corners = 76'd7821423522051142764673;
3523 12'd3511: quad_corners = 76'd7821423522051142764673;
3524 12'd3512: quad_corners = 76'd7821423522051142764673;
3525 12'd3513: quad_corners = 76'd7821423522051142764673;
3526 12'd3514: quad_corners = 76'd7821423522051142764673;
3527 12'd3515: quad_corners = 76'd7821423522051142764673;
3528 12'd3516: quad_corners = 76'd7821423522051142764673;
3529 12'd3517: quad_corners = 76'd7821423522051142764673;
3530 12'd3518: quad_corners = 76'd7821423522051142764673;
3531 12'd3519: quad_corners = 76'd7821423522051142764673;
3532 12'd3520: quad_corners = 76'd9227553297744240892571;
3533 12'd3521: quad_corners = 76'd9227553297744240892571;
3534 12'd3522: quad_corners = 76'd9227553297744240892571;
3535 12'd3523: quad_corners = 76'd9227553297744240892571;
3536 12'd3524: quad_corners = 76'd9227553297744240892571;
3537 12'd3525: quad_corners = 76'd9227553297744240892571;
3538 12'd3526: quad_corners = 76'd9227553297744240892571;
3539 12'd3527: quad_corners = 76'd9227553297744240892571;
3540 12'd3528: quad_corners = 76'd9227553297744240892571;

3541 12'd3529: quad_corners = 76'd9227553297744240892571;
3542 12'd3530: quad_corners = 76'd9227553297744240892571;
3543 12'd3531: quad_corners = 76'd9227553297744240892571;
3544 12'd3532: quad_corners = 76'd9227553297744240892571;
3545 12'd3533: quad_corners = 76'd9227553297744240892571;
3546 12'd3534: quad_corners = 76'd9227553297744240892571;
3547 12'd3535: quad_corners = 76'd9227553297744240892571;
3548 12'd3536: quad_corners = 76'd9227553297744240892571;
3549 12'd3537: quad_corners = 76'd9227553297744240892571;
3550 12'd3538: quad_corners = 76'd9227553297744240892571;
3551 12'd3539: quad_corners = 76'd9227553297744240892571;
3552 12'd3540: quad_corners = 76'd9227553297744240892571;
3553 12'd3541: quad_corners = 76'd9227553297744240892571;
3554 12'd3542: quad_corners = 76'd9227553297744240892571;
3555 12'd3543: quad_corners = 76'd9227553297744240892571;
3556 12'd3544: quad_corners = 76'd9227553297744240892571;
3557 12'd3545: quad_corners = 76'd9227409042094092508314;
3558 12'd3546: quad_corners = 76'd9227264926906554048154;
3559 12'd3547: quad_corners = 76'd9153333694961567457433;
3560 12'd3548: quad_corners = 76'd9153045464585685229270;
3561 12'd3549: quad_corners = 76'd9079114373378186470040;
3562 12'd3550: quad_corners = 76'd9078970258190380098199;
3563 12'd3551: quad_corners = 76'd9005039166982613427350;
3564 12'd3552: quad_corners = 76'd9004750936606730675349;
3565 12'd3553: quad_corners = 76'd8930819845398964003988;
3566 12'd3554: quad_corners = 76'd8930675730211157107860;
3567 12'd3555: quad_corners = 76'd8856744779740878791827;
3568 12'd3556: quad_corners = 76'd8782669573069889922194;
3569 12'd3557: quad_corners = 76'd8782525458156960933009;
3570 12'd3558: quad_corners = 76'd8708594507411536274576;
3571 12'd3559: quad_corners = 76'd8634663416203500643471;
3572 12'd3560: quad_corners = 76'd8634519441752914190990;
3573 12'd3561: quad_corners = 76'd8560444375819413152397;
3574 12'd3562: quad_corners = 76'd8486513425348866400396;
3575 12'd3563: quad_corners = 76'd8412582333865952861835;
3576 12'd3564: quad_corners = 76'd8412438359690244316298;
3577 12'd3565: quad_corners = 76'd8338507408944551221897;
3578 12'd3566: quad_corners = 76'd8264432483748538538120;
3579 12'd3567: quad_corners = 76'd8190501533277723350663;
3580 12'd3568: quad_corners = 76'd8116570582532029731974;
3581 12'd3569: quad_corners = 76'd8042783746974411444869;
3582 12'd3570: quad_corners = 76'd7968997052429159419524;
3583 12'd3571: quad_corners = 76'd7895210216871541132419;
3584 12'd3572: quad_corners = 76'd7821423522051142764673;
3585 12'd3573: quad_corners = 76'd7821423522051142764673;
3586 12'd3574: quad_corners = 76'd7821423522051142764673;

3587 12'd3575: quad_corners = 76'd7821423522051142764673;
3588 12'd3576: quad_corners = 76'd7821423522051142764673;
3589 12'd3577: quad_corners = 76'd7821423522051142764673;
3590 12'd3578: quad_corners = 76'd7821423522051142764673;
3591 12'd3579: quad_corners = 76'd7821423522051142764673;
3592 12'd3580: quad_corners = 76'd7821423522051142764673;
3593 12'd3581: quad_corners = 76'd7821423522051142764673;
3594 12'd3582: quad_corners = 76'd7821423522051142764673;
3595 12'd3583: quad_corners = 76'd7821423522051142764673;
3596 12'd3584: quad_corners = 76'd9227553297744240892571;
3597 12'd3585: quad_corners = 76'd9227553297744240892571;
3598 12'd3586: quad_corners = 76'd9227553297744240892571;
3599 12'd3587: quad_corners = 76'd9227553297744240892571;
3600 12'd3588: quad_corners = 76'd9227553297744240892571;
3601 12'd3589: quad_corners = 76'd9227553297744240892571;
3602 12'd3590: quad_corners = 76'd9227553297744240892571;
3603 12'd3591: quad_corners = 76'd9227553297744240892571;
3604 12'd3592: quad_corners = 76'd9227553297744240892571;
3605 12'd3593: quad_corners = 76'd9227553297744240892571;
3606 12'd3594: quad_corners = 76'd9227553297744240892571;
3607 12'd3595: quad_corners = 76'd9227553297744240892571;
3608 12'd3596: quad_corners = 76'd9227553297744240892571;
3609 12'd3597: quad_corners = 76'd9227553297744240892571;
3610 12'd3598: quad_corners = 76'd9227553297744240892571;
3611 12'd3599: quad_corners = 76'd9227553297744240892571;
3612 12'd3600: quad_corners = 76'd9227553297744240892571;
3613 12'd3601: quad_corners = 76'd9227553297744240892571;
3614 12'd3602: quad_corners = 76'd9227553297744240892571;
3615 12'd3603: quad_corners = 76'd9227553297744240892571;
3616 12'd3604: quad_corners = 76'd9227553297744240892571;
3617 12'd3605: quad_corners = 76'd9227553297744240892571;
3618 12'd3606: quad_corners = 76'd9227553297744240892571;
3619 12'd3607: quad_corners = 76'd9227553297744240892571;
3620 12'd3608: quad_corners = 76'd9227553297744240892571;
3621 12'd3609: quad_corners = 76'd9227409042094092508314;
3622 12'd3610: quad_corners = 76'd9227264926906554048154;
3623 12'd3611: quad_corners = 76'd9153333694961567457433;
3624 12'd3612: quad_corners = 76'd9153045464585685229720;
3625 12'd3613: quad_corners = 76'd9079114373378186470040;
3626 12'd3614: quad_corners = 76'd9078970258190380098199;
3627 12'd3615: quad_corners = 76'd9005039166982613427350;
3628 12'd3616: quad_corners = 76'd9004750936606730675349;
3629 12'd3617: quad_corners = 76'd8930819845398964003988;
3630 12'd3618: quad_corners = 76'd8930675730211157107860;
3631 12'd3619: quad_corners = 76'd8856744779740878791827;
3632 12'd3620: quad_corners = 76'd8782669573069889922194;

3633 12'd3621: quad_corners = 76'd8782525458156960933009;
3634 12'd3622: quad_corners = 76'd8708594507411536274576;
3635 12'd3623: quad_corners = 76'd8634663416203500643471;
3636 12'd3624: quad_corners = 76'd8634519441752914190990;
3637 12'd3625: quad_corners = 76'd8560444375819413152397;
3638 12'd3626: quad_corners = 76'd8486513425348866400396;
3639 12'd3627: quad_corners = 76'd8412582333865952861835;
3640 12'd3628: quad_corners = 76'd8412438359690244316298;
3641 12'd3629: quad_corners = 76'd8338507408944551221897;
3642 12'd3630: quad_corners = 76'd8264432483748538538120;
3643 12'd3631: quad_corners = 76'd8190501533277723350663;
3644 12'd3632: quad_corners = 76'd8116570582532029731974;
3645 12'd3633: quad_corners = 76'd8042783746974411444869;
3646 12'd3634: quad_corners = 76'd7968997052429159419524;
3647 12'd3635: quad_corners = 76'd7895210216871541132419;
3648 12'd3636: quad_corners = 76'd7821423522051142764673;
3649 12'd3637: quad_corners = 76'd7821423522051142764673;
3650 12'd3638: quad_corners = 76'd7821423522051142764673;
3651 12'd3639: quad_corners = 76'd7821423522051142764673;
3652 12'd3640: quad_corners = 76'd7821423522051142764673;
3653 12'd3641: quad_corners = 76'd7821423522051142764673;
3654 12'd3642: quad_corners = 76'd7821423522051142764673;
3655 12'd3643: quad_corners = 76'd7821423522051142764673;
3656 12'd3644: quad_corners = 76'd7821423522051142764673;
3657 12'd3645: quad_corners = 76'd7821423522051142764673;
3658 12'd3646: quad_corners = 76'd7821423522051142764673;
3659 12'd3647: quad_corners = 76'd7821423522051142764673;
3660 12'd3648: quad_corners = 76'd9227553297744240892571;
3661 12'd3649: quad_corners = 76'd9227553297744240892571;
3662 12'd3650: quad_corners = 76'd9227553297744240892571;
3663 12'd3651: quad_corners = 76'd9227553297744240892571;
3664 12'd3652: quad_corners = 76'd9227553297744240892571;
3665 12'd3653: quad_corners = 76'd9227553297744240892571;
3666 12'd3654: quad_corners = 76'd9227553297744240892571;
3667 12'd3655: quad_corners = 76'd9227553297744240892571;
3668 12'd3656: quad_corners = 76'd9227553297744240892571;
3669 12'd3657: quad_corners = 76'd9227553297744240892571;
3670 12'd3658: quad_corners = 76'd9227553297744240892571;
3671 12'd3659: quad_corners = 76'd9227553297744240892571;
3672 12'd3660: quad_corners = 76'd9227553297744240892571;
3673 12'd3661: quad_corners = 76'd9227553297744240892571;
3674 12'd3662: quad_corners = 76'd9227553297744240892571;
3675 12'd3663: quad_corners = 76'd9227553297744240892571;
3676 12'd3664: quad_corners = 76'd9227553297744240892571;
3677 12'd3665: quad_corners = 76'd9227553297744240892571;
3678 12'd3666: quad_corners = 76'd9227553297744240892571;

3679 12'd3667: quad_corners = 76'd9227553297744240892571;
3680 12'd3668: quad_corners = 76'd9227553297744240892571;
3681 12'd3669: quad_corners = 76'd9227553297744240892571;
3682 12'd3670: quad_corners = 76'd9227553297744240892571;
3683 12'd3671: quad_corners = 76'd9227553297744240892571;
3684 12'd3672: quad_corners = 76'd9227553297744240892571;
3685 12'd3673: quad_corners = 76'd9227409042094092508314;
3686 12'd3674: quad_corners = 76'd9227264926906554048154;
3687 12'd3675: quad_corners = 76'd9153333694961567457433;
3688 12'd3676: quad_corners = 76'd9153045464585685229720;
3689 12'd3677: quad_corners = 76'd9079114373378186470040;
3690 12'd3678: quad_corners = 76'd9078970258190380098199;
3691 12'd3679: quad_corners = 76'd9005039166982613427350;
3692 12'd3680: quad_corners = 76'd9004750936606730675349;
3693 12'd3681: quad_corners = 76'd8930819845398964003988;
3694 12'd3682: quad_corners = 76'd8930675730211157107860;
3695 12'd3683: quad_corners = 76'd8856744779740878791827;
3696 12'd3684: quad_corners = 76'd8782669573069889922194;
3697 12'd3685: quad_corners = 76'd8782525458156960933009;
3698 12'd3686: quad_corners = 76'd8708594507411536274576;
3699 12'd3687: quad_corners = 76'd8634663416203500643471;
3700 12'd3688: quad_corners = 76'd8634519441752914190990;
3701 12'd3689: quad_corners = 76'd8560444375819413152397;
3702 12'd3690: quad_corners = 76'd8486513425348866400396;
3703 12'd3691: quad_corners = 76'd8412582333865952861835;
3704 12'd3692: quad_corners = 76'd8412438359690244316298;
3705 12'd3693: quad_corners = 76'd8338507408944551221897;
3706 12'd3694: quad_corners = 76'd8264432483748538538120;
3707 12'd3695: quad_corners = 76'd8190501533277723350663;
3708 12'd3696: quad_corners = 76'd8116570582532029731974;
3709 12'd3697: quad_corners = 76'd8042783746974411444869;
3710 12'd3698: quad_corners = 76'd7968997052429159419524;
3711 12'd3699: quad_corners = 76'd7895210216871541132419;
3712 12'd3700: quad_corners = 76'd7821423522051142764673;
3713 12'd3701: quad_corners = 76'd7821423522051142764673;
3714 12'd3702: quad_corners = 76'd7821423522051142764673;
3715 12'd3703: quad_corners = 76'd7821423522051142764673;
3716 12'd3704: quad_corners = 76'd7821423522051142764673;
3717 12'd3705: quad_corners = 76'd7821423522051142764673;
3718 12'd3706: quad_corners = 76'd7821423522051142764673;
3719 12'd3707: quad_corners = 76'd7821423522051142764673;
3720 12'd3708: quad_corners = 76'd7821423522051142764673;
3721 12'd3709: quad_corners = 76'd7821423522051142764673;
3722 12'd3710: quad_corners = 76'd7821423522051142764673;
3723 12'd3711: quad_corners = 76'd7821423522051142764673;
3724 12'd3712: quad_corners = 76'd9227553297744240892571;

3725 12'd3713: quad_corners = 76'd9227553297744240892571;
3726 12'd3714: quad_corners = 76'd9227553297744240892571;
3727 12'd3715: quad_corners = 76'd9227553297744240892571;
3728 12'd3716: quad_corners = 76'd9227553297744240892571;
3729 12'd3717: quad_corners = 76'd9227553297744240892571;
3730 12'd3718: quad_corners = 76'd9227553297744240892571;
3731 12'd3719: quad_corners = 76'd9227553297744240892571;
3732 12'd3720: quad_corners = 76'd9227553297744240892571;
3733 12'd3721: quad_corners = 76'd9227553297744240892571;
3734 12'd3722: quad_corners = 76'd9227553297744240892571;
3735 12'd3723: quad_corners = 76'd9227553297744240892571;
3736 12'd3724: quad_corners = 76'd9227553297744240892571;
3737 12'd3725: quad_corners = 76'd9227553297744240892571;
3738 12'd3726: quad_corners = 76'd9227553297744240892571;
3739 12'd3727: quad_corners = 76'd9227553297744240892571;
3740 12'd3728: quad_corners = 76'd9227553297744240892571;
3741 12'd3729: quad_corners = 76'd9227553297744240892571;
3742 12'd3730: quad_corners = 76'd9227553297744240892571;
3743 12'd3731: quad_corners = 76'd9227553297744240892571;
3744 12'd3732: quad_corners = 76'd9227553297744240892571;
3745 12'd3733: quad_corners = 76'd9227553297744240892571;
3746 12'd3734: quad_corners = 76'd9227553297744240892571;
3747 12'd3735: quad_corners = 76'd9227553297744240892571;
3748 12'd3736: quad_corners = 76'd9227553297744240892571;
3749 12'd3737: quad_corners = 76'd9227409042094092508314;
3750 12'd3738: quad_corners = 76'd9227264926906554048154;
3751 12'd3739: quad_corners = 76'd9153333694961567457433;
3752 12'd3740: quad_corners = 76'd9153045464585685229720;
3753 12'd3741: quad_corners = 76'd9079114373378186470040;
3754 12'd3742: quad_corners = 76'd9078970258190380098199;
3755 12'd3743: quad_corners = 76'd9005039166982613427350;
3756 12'd3744: quad_corners = 76'd9004750936606730675349;
3757 12'd3745: quad_corners = 76'd8930819845398964003988;
3758 12'd3746: quad_corners = 76'd8930675730211157107860;
3759 12'd3747: quad_corners = 76'd8856744779740878791827;
3760 12'd3748: quad_corners = 76'd8782669573069889922194;
3761 12'd3749: quad_corners = 76'd8782525458156960933009;
3762 12'd3750: quad_corners = 76'd8708594507411536274576;
3763 12'd3751: quad_corners = 76'd8634663416203500643471;
3764 12'd3752: quad_corners = 76'd8634519441752914190990;
3765 12'd3753: quad_corners = 76'd8560444375819413152397;
3766 12'd3754: quad_corners = 76'd8486513425348866400396;
3767 12'd3755: quad_corners = 76'd8412582333865952861835;
3768 12'd3756: quad_corners = 76'd8412438359690244316298;
3769 12'd3757: quad_corners = 76'd8338507408944551221897;
3770 12'd3758: quad_corners = 76'd8264432483748538538120;

3771 12'd3759: quad_corners = 76'd8190501533277723350663;
3772 12'd3760: quad_corners = 76'd8116570582532029731974;
3773 12'd3761: quad_corners = 76'd8042783746974411444869;
3774 12'd3762: quad_corners = 76'd7968997052429159419524;
3775 12'd3763: quad_corners = 76'd7895210216871541132419;
3776 12'd3764: quad_corners = 76'd7821423522051142764673;
3777 12'd3765: quad_corners = 76'd7821423522051142764673;
3778 12'd3766: quad_corners = 76'd7821423522051142764673;
3779 12'd3767: quad_corners = 76'd7821423522051142764673;
3780 12'd3768: quad_corners = 76'd7821423522051142764673;
3781 12'd3769: quad_corners = 76'd7821423522051142764673;
3782 12'd3770: quad_corners = 76'd7821423522051142764673;
3783 12'd3771: quad_corners = 76'd7821423522051142764673;
3784 12'd3772: quad_corners = 76'd7821423522051142764673;
3785 12'd3773: quad_corners = 76'd7821423522051142764673;
3786 12'd3774: quad_corners = 76'd7821423522051142764673;
3787 12'd3775: quad_corners = 76'd7821423522051142764673;
3788 12'd3776: quad_corners = 76'd9227553297744240892571;
3789 12'd3777: quad_corners = 76'd9227553297744240892571;
3790 12'd3778: quad_corners = 76'd9227553297744240892571;
3791 12'd3779: quad_corners = 76'd9227553297744240892571;
3792 12'd3780: quad_corners = 76'd9227553297744240892571;
3793 12'd3781: quad_corners = 76'd9227553297744240892571;
3794 12'd3782: quad_corners = 76'd9227553297744240892571;
3795 12'd3783: quad_corners = 76'd9227553297744240892571;
3796 12'd3784: quad_corners = 76'd9227553297744240892571;
3797 12'd3785: quad_corners = 76'd9227553297744240892571;
3798 12'd3786: quad_corners = 76'd9227553297744240892571;
3799 12'd3787: quad_corners = 76'd9227553297744240892571;
3800 12'd3788: quad_corners = 76'd9227553297744240892571;
3801 12'd3789: quad_corners = 76'd9227553297744240892571;
3802 12'd3790: quad_corners = 76'd9227553297744240892571;
3803 12'd3791: quad_corners = 76'd9227553297744240892571;
3804 12'd3792: quad_corners = 76'd9227553297744240892571;
3805 12'd3793: quad_corners = 76'd9227553297744240892571;
3806 12'd3794: quad_corners = 76'd9227553297744240892571;
3807 12'd3795: quad_corners = 76'd9227553297744240892571;
3808 12'd3796: quad_corners = 76'd9227553297744240892571;
3809 12'd3797: quad_corners = 76'd9227553297744240892571;
3810 12'd3798: quad_corners = 76'd9227553297744240892571;
3811 12'd3799: quad_corners = 76'd9227553297744240892571;
3812 12'd3800: quad_corners = 76'd9227553297744240892571;
3813 12'd3801: quad_corners = 76'd9227409042094092508314;
3814 12'd3802: quad_corners = 76'd9227264926906554048154;
3815 12'd3803: quad_corners = 76'd9153333694961567457433;
3816 12'd3804: quad_corners = 76'd9153045464585685229720;

3817 12'd3805: quad_corners = 76'd9079114373378186470040;
3818 12'd3806: quad_corners = 76'd9078970258190380098199;
3819 12'd3807: quad_corners = 76'd9005039166982613427350;
3820 12'd3808: quad_corners = 76'd9004750936606730675349;
3821 12'd3809: quad_corners = 76'd8930819845398964003988;
3822 12'd3810: quad_corners = 76'd8930675730211157107860;
3823 12'd3811: quad_corners = 76'd8856744779740878791827;
3824 12'd3812: quad_corners = 76'd8782669573069889922194;
3825 12'd3813: quad_corners = 76'd8782525458156960933009;
3826 12'd3814: quad_corners = 76'd8708594507411536274576;
3827 12'd3815: quad_corners = 76'd8634663416203500643471;
3828 12'd3816: quad_corners = 76'd8634519441752914190990;
3829 12'd3817: quad_corners = 76'd8560444375819413152397;
3830 12'd3818: quad_corners = 76'd8486513425348866400396;
3831 12'd3819: quad_corners = 76'd8412582333865952861835;
3832 12'd3820: quad_corners = 76'd8412438359690244316298;
3833 12'd3821: quad_corners = 76'd8338507408944551221897;
3834 12'd3822: quad_corners = 76'd8264432483748538538120;
3835 12'd3823: quad_corners = 76'd8190501533277723350663;
3836 12'd3824: quad_corners = 76'd8116570582532029731974;
3837 12'd3825: quad_corners = 76'd8042783746974411444869;
3838 12'd3826: quad_corners = 76'd7968997052429159419524;
3839 12'd3827: quad_corners = 76'd7895210216871541132419;
3840 12'd3828: quad_corners = 76'd7821423522051142764673;
3841 12'd3829: quad_corners = 76'd7821423522051142764673;
3842 12'd3830: quad_corners = 76'd7821423522051142764673;
3843 12'd3831: quad_corners = 76'd7821423522051142764673;
3844 12'd3832: quad_corners = 76'd7821423522051142764673;
3845 12'd3833: quad_corners = 76'd7821423522051142764673;
3846 12'd3834: quad_corners = 76'd7821423522051142764673;
3847 12'd3835: quad_corners = 76'd7821423522051142764673;
3848 12'd3836: quad_corners = 76'd7821423522051142764673;
3849 12'd3837: quad_corners = 76'd7821423522051142764673;
3850 12'd3838: quad_corners = 76'd7821423522051142764673;
3851 12'd3839: quad_corners = 76'd7821423522051142764673;
3852 12'd3840: quad_corners = 76'd9227553297744240892571;
3853 12'd3841: quad_corners = 76'd9227553297744240892571;
3854 12'd3842: quad_corners = 76'd9227553297744240892571;
3855 12'd3843: quad_corners = 76'd9227553297744240892571;
3856 12'd3844: quad_corners = 76'd9227553297744240892571;
3857 12'd3845: quad_corners = 76'd9227553297744240892571;
3858 12'd3846: quad_corners = 76'd9227553297744240892571;
3859 12'd3847: quad_corners = 76'd9227553297744240892571;
3860 12'd3848: quad_corners = 76'd9227553297744240892571;
3861 12'd3849: quad_corners = 76'd9227553297744240892571;
3862 12'd3850: quad_corners = 76'd9227553297744240892571;

3863 12'd3851: quad_corners = 76'd9227553297744240892571;
3864 12'd3852: quad_corners = 76'd9227553297744240892571;
3865 12'd3853: quad_corners = 76'd9227553297744240892571;
3866 12'd3854: quad_corners = 76'd9227553297744240892571;
3867 12'd3855: quad_corners = 76'd9227553297744240892571;
3868 12'd3856: quad_corners = 76'd9227553297744240892571;
3869 12'd3857: quad_corners = 76'd9227553297744240892571;
3870 12'd3858: quad_corners = 76'd9227553297744240892571;
3871 12'd3859: quad_corners = 76'd9227553297744240892571;
3872 12'd3860: quad_corners = 76'd9227553297744240892571;
3873 12'd3861: quad_corners = 76'd9227553297744240892571;
3874 12'd3862: quad_corners = 76'd9227553297744240892571;
3875 12'd3863: quad_corners = 76'd9227553297744240892571;
3876 12'd3864: quad_corners = 76'd9227553297744240892571;
3877 12'd3865: quad_corners = 76'd9227409042094092508314;
3878 12'd3866: quad_corners = 76'd9227264926906554048154;
3879 12'd3867: quad_corners = 76'd9153333694961567457433;
3880 12'd3868: quad_corners = 76'd9153045464585685229720;
3881 12'd3869: quad_corners = 76'd9079114373378186470040;
3882 12'd3870: quad_corners = 76'd9078970258190380098199;
3883 12'd3871: quad_corners = 76'd9005039166982613427350;
3884 12'd3872: quad_corners = 76'd9004750936606730675349;
3885 12'd3873: quad_corners = 76'd8930819845398964003988;
3886 12'd3874: quad_corners = 76'd8930675730211157107860;
3887 12'd3875: quad_corners = 76'd8856744779740878791827;
3888 12'd3876: quad_corners = 76'd8782669573069889922194;
3889 12'd3877: quad_corners = 76'd8782525458156960933009;
3890 12'd3878: quad_corners = 76'd8708594507411536274576;
3891 12'd3879: quad_corners = 76'd8634663416203500643471;
3892 12'd3880: quad_corners = 76'd8634519441752914190990;
3893 12'd3881: quad_corners = 76'd8560444375819413152397;
3894 12'd3882: quad_corners = 76'd8486513425348866400396;
3895 12'd3883: quad_corners = 76'd8412582333865952861835;
3896 12'd3884: quad_corners = 76'd8412438359690244316298;
3897 12'd3885: quad_corners = 76'd8338507408944551221897;
3898 12'd3886: quad_corners = 76'd8264432483748538538120;
3899 12'd3887: quad_corners = 76'd8190501533277723350663;
3900 12'd3888: quad_corners = 76'd8116570582532029731974;
3901 12'd3889: quad_corners = 76'd8042783746974411444869;
3902 12'd3890: quad_corners = 76'd7968997052429159419524;
3903 12'd3891: quad_corners = 76'd7895210216871541132419;
3904 12'd3892: quad_corners = 76'd7821423522051142764673;
3905 12'd3893: quad_corners = 76'd7821423522051142764673;
3906 12'd3894: quad_corners = 76'd7821423522051142764673;
3907 12'd3895: quad_corners = 76'd7821423522051142764673;
3908 12'd3896: quad_corners = 76'd7821423522051142764673;

3909 12'd3897: quad_corners = 76'd7821423522051142764673;
3910 12'd3898: quad_corners = 76'd7821423522051142764673;
3911 12'd3899: quad_corners = 76'd7821423522051142764673;
3912 12'd3900: quad_corners = 76'd7821423522051142764673;
3913 12'd3901: quad_corners = 76'd7821423522051142764673;
3914 12'd3902: quad_corners = 76'd7821423522051142764673;
3915 12'd3903: quad_corners = 76'd7821423522051142764673;
3916 12'd3904: quad_corners = 76'd9227553297744240892571;
3917 12'd3905: quad_corners = 76'd9227553297744240892571;
3918 12'd3906: quad_corners = 76'd9227553297744240892571;
3919 12'd3907: quad_corners = 76'd9227553297744240892571;
3920 12'd3908: quad_corners = 76'd9227553297744240892571;
3921 12'd3909: quad_corners = 76'd9227553297744240892571;
3922 12'd3910: quad_corners = 76'd9227553297744240892571;
3923 12'd3911: quad_corners = 76'd9227553297744240892571;
3924 12'd3912: quad_corners = 76'd9227553297744240892571;
3925 12'd3913: quad_corners = 76'd9227553297744240892571;
3926 12'd3914: quad_corners = 76'd9227553297744240892571;
3927 12'd3915: quad_corners = 76'd9227553297744240892571;
3928 12'd3916: quad_corners = 76'd9227553297744240892571;
3929 12'd3917: quad_corners = 76'd9227553297744240892571;
3930 12'd3918: quad_corners = 76'd9227553297744240892571;
3931 12'd3919: quad_corners = 76'd9227553297744240892571;
3932 12'd3920: quad_corners = 76'd9227553297744240892571;
3933 12'd3921: quad_corners = 76'd9227553297744240892571;
3934 12'd3922: quad_corners = 76'd9227553297744240892571;
3935 12'd3923: quad_corners = 76'd9227553297744240892571;
3936 12'd3924: quad_corners = 76'd9227553297744240892571;
3937 12'd3925: quad_corners = 76'd9227553297744240892571;
3938 12'd3926: quad_corners = 76'd9227553297744240892571;
3939 12'd3927: quad_corners = 76'd9227553297744240892571;
3940 12'd3928: quad_corners = 76'd9227553297744240892571;
3941 12'd3929: quad_corners = 76'd9227409042094092508314;
3942 12'd3930: quad_corners = 76'd9227264926906554048154;
3943 12'd3931: quad_corners = 76'd9153333694961567457433;
3944 12'd3932: quad_corners = 76'd9153045464585685229720;
3945 12'd3933: quad_corners = 76'd9079114373378186470040;
3946 12'd3934: quad_corners = 76'd9078970258190380098199;
3947 12'd3935: quad_corners = 76'd9005039166982613427350;
3948 12'd3936: quad_corners = 76'd9004750936606730675349;
3949 12'd3937: quad_corners = 76'd8930819845398964003988;
3950 12'd3938: quad_corners = 76'd8930675730211157107860;
3951 12'd3939: quad_corners = 76'd8856744779740878791827;
3952 12'd3940: quad_corners = 76'd8782669573069889922194;
3953 12'd3941: quad_corners = 76'd8782525458156960933009;
3954 12'd3942: quad_corners = 76'd8708594507411536274576;

3955 12'd3943: quad_corners = 76'd8634663416203500643471;
3956 12'd3944: quad_corners = 76'd8634519441752914190990;
3957 12'd3945: quad_corners = 76'd8560444375819413152397;
3958 12'd3946: quad_corners = 76'd8486513425348866400396;
3959 12'd3947: quad_corners = 76'd8412582333865952861835;
3960 12'd3948: quad_corners = 76'd8412438359690244316298;
3961 12'd3949: quad_corners = 76'd8338507408944551221897;
3962 12'd3950: quad_corners = 76'd8264432483748538538120;
3963 12'd3951: quad_corners = 76'd8190501533277723350663;
3964 12'd3952: quad_corners = 76'd8116570582532029731974;
3965 12'd3953: quad_corners = 76'd8042783746974411444869;
3966 12'd3954: quad_corners = 76'd7968997052429159419524;
3967 12'd3955: quad_corners = 76'd7895210216871541132419;
3968 12'd3956: quad_corners = 76'd7821423522051142764673;
3969 12'd3957: quad_corners = 76'd7821423522051142764673;
3970 12'd3958: quad_corners = 76'd7821423522051142764673;
3971 12'd3959: quad_corners = 76'd7821423522051142764673;
3972 12'd3960: quad_corners = 76'd7821423522051142764673;
3973 12'd3961: quad_corners = 76'd7821423522051142764673;
3974 12'd3962: quad_corners = 76'd7821423522051142764673;
3975 12'd3963: quad_corners = 76'd7821423522051142764673;
3976 12'd3964: quad_corners = 76'd7821423522051142764673;
3977 12'd3965: quad_corners = 76'd7821423522051142764673;
3978 12'd3966: quad_corners = 76'd7821423522051142764673;
3979 12'd3967: quad_corners = 76'd7821423522051142764673;
3980 12'd3968: quad_corners = 76'd9227553297744240892571;
3981 12'd3969: quad_corners = 76'd9227553297744240892571;
3982 12'd3970: quad_corners = 76'd9227553297744240892571;
3983 12'd3971: quad_corners = 76'd9227553297744240892571;
3984 12'd3972: quad_corners = 76'd9227553297744240892571;
3985 12'd3973: quad_corners = 76'd9227553297744240892571;
3986 12'd3974: quad_corners = 76'd9227553297744240892571;
3987 12'd3975: quad_corners = 76'd9227553297744240892571;
3988 12'd3976: quad_corners = 76'd9227553297744240892571;
3989 12'd3977: quad_corners = 76'd9227553297744240892571;
3990 12'd3978: quad_corners = 76'd9227553297744240892571;
3991 12'd3979: quad_corners = 76'd9227553297744240892571;
3992 12'd3980: quad_corners = 76'd9227553297744240892571;
3993 12'd3981: quad_corners = 76'd9227553297744240892571;
3994 12'd3982: quad_corners = 76'd9227553297744240892571;
3995 12'd3983: quad_corners = 76'd9227553297744240892571;
3996 12'd3984: quad_corners = 76'd9227553297744240892571;
3997 12'd3985: quad_corners = 76'd9227553297744240892571;
3998 12'd3986: quad_corners = 76'd9227553297744240892571;
3999 12'd3987: quad_corners = 76'd9227553297744240892571;
4000 12'd3988: quad_corners = 76'd9227553297744240892571;

4001 12'd3989: quad_corners = 76'd9227553297744240892571;
4002 12'd3990: quad_corners = 76'd9227553297744240892571;
4003 12'd3991: quad_corners = 76'd9227553297744240892571;
4004 12'd3992: quad_corners = 76'd9227553297744240892571;
4005 12'd3993: quad_corners = 76'd9227409042094092508314;
4006 12'd3994: quad_corners = 76'd9227264926906554048154;
4007 12'd3995: quad_corners = 76'd9153333694961567457433;
4008 12'd3996: quad_corners = 76'd9153045464585685229720;
4009 12'd3997: quad_corners = 76'd9079114373378186470040;
4010 12'd3998: quad_corners = 76'd9078970258190380098199;
4011 12'd3999: quad_corners = 76'd9005039166982613427350;
4012 12'd4000: quad_corners = 76'd9004750936606730675349;
4013 12'd4001: quad_corners = 76'd8930819845398964003988;
4014 12'd4002: quad_corners = 76'd8930675730211157107860;
4015 12'd4003: quad_corners = 76'd8856744779740878791827;
4016 12'd4004: quad_corners = 76'd8782669573069889922194;
4017 12'd4005: quad_corners = 76'd8782525458156960933009;
4018 12'd4006: quad_corners = 76'd8708594507411536274576;
4019 12'd4007: quad_corners = 76'd8634663416203500643471;
4020 12'd4008: quad_corners = 76'd8634519441752914190990;
4021 12'd4009: quad_corners = 76'd8560444375819413152397;
4022 12'd4010: quad_corners = 76'd8486513425348866400396;
4023 12'd4011: quad_corners = 76'd8412582333865952861835;
4024 12'd4012: quad_corners = 76'd8412438359690244316298;
4025 12'd4013: quad_corners = 76'd8338507408944551221897;
4026 12'd4014: quad_corners = 76'd8264432483748538538120;
4027 12'd4015: quad_corners = 76'd8190501533277723350663;
4028 12'd4016: quad_corners = 76'd8116570582532029731974;
4029 12'd4017: quad_corners = 76'd8042783746974411444869;
4030 12'd4018: quad_corners = 76'd7968997052429159419524;
4031 12'd4019: quad_corners = 76'd7895210216871541132419;
4032 12'd4020: quad_corners = 76'd7821423522051142764673;
4033 12'd4021: quad_corners = 76'd7821423522051142764673;
4034 12'd4022: quad_corners = 76'd7821423522051142764673;
4035 12'd4023: quad_corners = 76'd7821423522051142764673;
4036 12'd4024: quad_corners = 76'd7821423522051142764673;
4037 12'd4025: quad_corners = 76'd7821423522051142764673;
4038 12'd4026: quad_corners = 76'd7821423522051142764673;
4039 12'd4027: quad_corners = 76'd7821423522051142764673;
4040 12'd4028: quad_corners = 76'd7821423522051142764673;
4041 12'd4029: quad_corners = 76'd7821423522051142764673;
4042 12'd4030: quad_corners = 76'd7821423522051142764673;
4043 12'd4031: quad_corners = 76'd7821423522051142764673;
4044 12'd4032: quad_corners = 76'd9227553297744240892571;
4045 12'd4033: quad_corners = 76'd9227553297744240892571;
4046 12'd4034: quad_corners = 76'd9227553297744240892571;

4047 12'd4035: quad_corners = 76'd9227553297744240892571;
4048 12'd4036: quad_corners = 76'd9227553297744240892571;
4049 12'd4037: quad_corners = 76'd9227553297744240892571;
4050 12'd4038: quad_corners = 76'd9227553297744240892571;
4051 12'd4039: quad_corners = 76'd9227553297744240892571;
4052 12'd4040: quad_corners = 76'd9227553297744240892571;
4053 12'd4041: quad_corners = 76'd9227553297744240892571;
4054 12'd4042: quad_corners = 76'd9227553297744240892571;
4055 12'd4043: quad_corners = 76'd9227553297744240892571;
4056 12'd4044: quad_corners = 76'd9227553297744240892571;
4057 12'd4045: quad_corners = 76'd9227553297744240892571;
4058 12'd4046: quad_corners = 76'd9227553297744240892571;
4059 12'd4047: quad_corners = 76'd9227553297744240892571;
4060 12'd4048: quad_corners = 76'd9227553297744240892571;
4061 12'd4049: quad_corners = 76'd9227553297744240892571;
4062 12'd4050: quad_corners = 76'd9227553297744240892571;
4063 12'd4051: quad_corners = 76'd9227553297744240892571;
4064 12'd4052: quad_corners = 76'd9227553297744240892571;
4065 12'd4053: quad_corners = 76'd9227553297744240892571;
4066 12'd4054: quad_corners = 76'd9227553297744240892571;
4067 12'd4055: quad_corners = 76'd9227553297744240892571;
4068 12'd4056: quad_corners = 76'd9227553297744240892571;
4069 12'd4057: quad_corners = 76'd9227409042094092508314;
4070 12'd4058: quad_corners = 76'd9227264926906554048154;
4071 12'd4059: quad_corners = 76'd9153333694961567457433;
4072 12'd4060: quad_corners = 76'd9153045464585685229720;
4073 12'd4061: quad_corners = 76'd9079114373378186470040;
4074 12'd4062: quad_corners = 76'd9078970258190380089199;
4075 12'd4063: quad_corners = 76'd9005039166982613427350;
4076 12'd4064: quad_corners = 76'd9004750936606730675349;
4077 12'd4065: quad_corners = 76'd8930819845398964003988;
4078 12'd4066: quad_corners = 76'd8930675730211157107860;
4079 12'd4067: quad_corners = 76'd8856744779740878791827;
4080 12'd4068: quad_corners = 76'd8782669573069889922194;
4081 12'd4069: quad_corners = 76'd8782525458156960933009;
4082 12'd4070: quad_corners = 76'd8708594507411536274576;
4083 12'd4071: quad_corners = 76'd8634663416203500643471;
4084 12'd4072: quad_corners = 76'd8634519441752914190990;
4085 12'd4073: quad_corners = 76'd8560444375819413152397;
4086 12'd4074: quad_corners = 76'd8486513425348866400396;
4087 12'd4075: quad_corners = 76'd8412582333865952861835;
4088 12'd4076: quad_corners = 76'd8412438359690244316298;
4089 12'd4077: quad_corners = 76'd8338507408944551221897;
4090 12'd4078: quad_corners = 76'd8264432483748538538120;
4091 12'd4079: quad_corners = 76'd8190501533277723350663;
4092 12'd4080: quad_corners = 76'd8116570582532029731974;

```

4093         12'd4081: quad_corners = 76'd8042783746974411444869;
4094         12'd4082: quad_corners = 76'd7968997052429159419524;
4095         12'd4083: quad_corners = 76'd7895210216871541132419;
4096         12'd4084: quad_corners = 76'd7821423522051142764673;
4097         12'd4085: quad_corners = 76'd7821423522051142764673;
4098         12'd4086: quad_corners = 76'd7821423522051142764673;
4099         12'd4087: quad_corners = 76'd7821423522051142764673;
4100         12'd4088: quad_corners = 76'd7821423522051142764673;
4101         12'd4089: quad_corners = 76'd7821423522051142764673;
4102         12'd4090: quad_corners = 76'd7821423522051142764673;
4103         12'd4091: quad_corners = 76'd7821423522051142764673;
4104         12'd4092: quad_corners = 76'd7821423522051142764673;
4105         12'd4093: quad_corners = 76'd7821423522051142764673;
4106         12'd4094: quad_corners = 76'd7821423522051142764673;
4107         12'd4095: quad_corners = 76'd7821423522051142764673;
4108     endcase
4109 end
4110 endmodule

```

A.3.3 accel_lut.jl

```

1  #=
2  Copyright (C) {2014} {Ganesh Ajjanagadde} <gajjanagadde@gmail.com>
3
4  This program is free software: you can redistribute it and/or modify
5  it under the terms of the GNU General Public License as published by
6  the Free Software Foundation, either version 3 of the License, or
7  (at your option) any later version.
8
9  This program is distributed in the hope that it will be useful,
10 but WITHOUT ANY WARRANTY; without even the implied warranty of
11 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12 GNU General Public License for more details.
13
14 You should have received a copy of the GNU General Public License
15 along with this program. If not, see <http://www.gnu.org/licenses/>.
16
17 This script generates an accel_lut.v file.
18 accel_lut.v contains a verilog implementation of a lookup table,
19 which takes in an accelerometer reading (6 bit x dir, 6 bit y dir),
20 and looks up a 76 bit value (4 corners of quadrilateral)
21 This script requires a input file accel_lut.txt containing data points.
22 It then interpolates the data points using 2D splines,
23 and creates the desired lookup table.
24
25 Format of accel_lut.txt:

```



```

26 x_accel, y_accel, x1, y1, x2, y2, x3, y3, x4, y4
27
28 i.e it is a csv file, each line denoting a reading
29 for ease of entry, all values are hex (just read off hex display)
30 leading zeros don't have to be specified, but can be if desired
31
32 NOTE: accel_lut.v and accel_lut.txt are suggested names
33 General installation:
34 1) Install julia
35 2) open Julia interpreter, i.e julia at cmd line
36 3) install gfortran (gcc frontend for fortran), needed to compile interpolation package
37 4) install Dierckx (for the spline interpolation) by 'Pkg.add("Dierckx")' at julia prompt
38 5) Dierckx details (docs, installation help, etc): https://github.com/kbarbary/Dierckx.jl
39
40 General usage:
41 accel_lut(input_path, output_path) at julia prompt
42
43 input_path is path to the csv file, and output_path is path to the desired .v file
44 NOTE: in order to run the command, you first need to include this file, so type:
45 include("accel_lut.jl") at the julia prompt prior to running the above
46 NOTE: THIS CODE WILL OVERWRITE THE FILE AT OUTPUT_PATH!!!
47
48 Suggested usage:
49 accel_lut("./accel_lut.txt", "./accel_lut.v")
50 =#
51
52 using Dierckx # interpolation package
53
54 function read_file(path)
55     return readcsv(path, String)
56 end
57
58 function parse_data(path)
59     raw_data = read_file(path)
60     num_samples = size(raw_data)[1]
61     x_accel = zeros(Int64, num_samples)
62     y_accel = zeros(Int64, num_samples)
63     x1 = zeros(Int64, num_samples)
64     y1 = zeros(Int64, num_samples)
65     x2 = zeros(Int64, num_samples)
66     y2 = zeros(Int64, num_samples)
67     x3 = zeros(Int64, num_samples)
68     y3 = zeros(Int64, num_samples)
69     x4 = zeros(Int64, num_samples)
70     y4 = zeros(Int64, num_samples)
71     base = 16

```

```

72     for i = 1:num_samples
73         x_accel[i] = parseint(raw_data[i,1], base)
74         y_accel[i] = parseint(raw_data[i,2], base)
75         x1[i] = parseint(raw_data[i,3], base)
76         y1[i] = parseint(raw_data[i,4], base)
77         x2[i] = parseint(raw_data[i,5], base)
78         y2[i] = parseint(raw_data[i,6], base)
79         x3[i] = parseint(raw_data[i,7], base)
80         y3[i] = parseint(raw_data[i,8], base)
81         x4[i] = parseint(raw_data[i,9], base)
82         y4[i] = parseint(raw_data[i,10], base)
83     end
84     return float(x_accel), float(y_accel), float(x1), float(y1), float(x2), float(y2), float(x3), float(y3), float(x4), float(y4)
85 end
86
87 function saturate!(vec, low, upp)
88     for i=1:length(vec)
89         if (vec[i] < low)
90             vec[i] = low
91         elseif (vec[i] > upp)
92             vec[i] = upp
93         end
94     end
95     return vec
96 end
97
98 function write_file(path, x_accel, y_accel, x1, y1, x2, y2, x3, y3, x4, y4)
99     # compute grid
100     x = zeros(2^12)
101     y = zeros(2^12)
102     quad_corners = zeros(Int128, 2^12)
103     for i=0:2^12-1
104         x[i+1] = i >> 6
105         y[i+1] = i & ((1 << 6) - 1)
106     end
107
108     # do the spline interpolation
109     # we do linear fits for now
110     # as we add more points, we can do something more sophisticated
111     x_deg = 2;
112     y_deg = 2;
113     # we also use a smoothing factor
114     # this trades off exact interpolation vs weighted least squares
115     # for more details, see doc at: https://github.com/kbarbary/Dierckx.jl
116     smooth_factor = 454.0;
117     spline_x1 = Spline2D(x_accel, y_accel, x1; kx=x_deg, ky=y_deg, s=smooth_factor)

```

```

118 spline_x2 = Spline2D(x_accel, y_accel, x2; kx=x_deg, ky=y_deg, s=smooth_factor)
119 spline_x3 = Spline2D(x_accel, y_accel, x3; kx=x_deg, ky=y_deg, s=smooth_factor)
120 spline_x4 = Spline2D(x_accel, y_accel, x4; kx=x_deg, ky=y_deg, s=smooth_factor)
121 spline_y1 = Spline2D(x_accel, y_accel, y1; kx=x_deg, ky=y_deg, s=smooth_factor)
122 spline_y2 = Spline2D(x_accel, y_accel, y2; kx=x_deg, ky=y_deg, s=smooth_factor)
123 spline_y3 = Spline2D(x_accel, y_accel, y3; kx=x_deg, ky=y_deg, s=smooth_factor)
124 spline_y4 = Spline2D(x_accel, y_accel, y4; kx=x_deg, ky=y_deg, s=smooth_factor)
125 x1_interp = int128(evaluate(spline_x1, x, y))
126 x2_interp = int128(evaluate(spline_x2, x, y))
127 x3_interp = int128(evaluate(spline_x3, x, y))
128 x4_interp = int128(evaluate(spline_x4, x, y))
129 y1_interp = int128(evaluate(spline_y1, x, y))
130 y2_interp = int128(evaluate(spline_y2, x, y))
131 y3_interp = int128(evaluate(spline_y3, x, y))
132 y4_interp = int128(evaluate(spline_y4, x, y))
133
134 # threshold x, y coords at appropriate values
135 # this is to guarantee we are not putting garbage into the lut
136 low_x = 0
137 low_y = 0
138 upp_x = 639
139 upp_y = 479
140 saturate!(x1_interp, low_x, upp_x)
141 saturate!(x2_interp, low_x, upp_x)
142 saturate!(x3_interp, low_x, upp_x)
143 saturate!(x4_interp, low_x, upp_x)
144 saturate!(y1_interp, low_y, upp_y)
145 saturate!(y2_interp, low_y, upp_y)
146 saturate!(y3_interp, low_y, upp_y)
147 saturate!(y4_interp, low_y, upp_y)
148
149 # compute quad_corners
150 for i=1:2^12
151     quad_corners[i] = y4_interp[i] + (x4_interp[i] << 9) + (y3_interp[i] << 19) + (x3_in
152     quad_corners[i] += (y2_interp[i] << 38) + (x2_interp[i] << 47) + (y1_interp[i] << 5
153 end
154
155 # write header
156 comment_head = "////////////////////////////////////"
157 comment_body1 = "//This file was autogenerated by accel_lut.jl.\n"
158 comment_body2 = "//DO NOT MANUALLY EDIT THIS FILE!!!\n\n"
159 comment_body3 = "//This file implements accel_lut rom for lookup of quadrilateral corner
160 comment_tail = "////////////////////////////////////"
161 code_preamble1 = "module accel_lut(input clk, input[11:0] accel_val, output reg[75:0] qu
162 code_preamble2 = "always @(posedge clk) begin\n"
163 code_preamble3 = "\tcase (accel_val)\n";

```

```

164     fs = open(path, "w")
165     write(fs, string(comment_head, comment_body1, comment_body2, comment_body3, comment_tail))
166
167     # write body
168     for i=0:2^12-1
169         val = quad_corners[i+1]
170         line_str = string("\t\t12'd", i, ": quad_corners = 76'd", val, ";\n")
171         write(fs, line_str)
172     end
173
174     # write footer
175     code_end = "\tendcase\nend\nendmodule\n"
176     write(fs, code_end)
177     close(fs)
178 end
179
180 function accel_lut(in_path, out_path)
181     x_accel, y_accel, x1, y1, x2, y2, x3, y3, x4, y4 = parse_data(in_path)
182     write_file(out_path, x_accel, y_accel, x1, y1, x2, y2, x3, y3, x4, y4)
183 end

```

A.3.4 accel_lut.txt

```

1  20,20,      0,0,      0,1df,      27f,1df,      27f,0
2  20,24,      a,d,      0,1df,      27f,1df,      275,d
3  20,28,     12,1a,     0,1df,      27f,1df,      26e,1a
4  20,2c,     1b,39,     0,1df,      27f,1df,      265,39
5  20,30,     1f,43,     0,1df,      27f,1df,      263,43
6  20,34,     27,4d,     0,1df,      27f,1df,      25b,4d
7  28,20,     28,0,      0,1b9,      25c,1df,      27f,1e
8  30,20,     5a,0,      0,176,      22c,1df,      27f,72
9  34,30,     6e,0,      16,155,     20f,1df,      26a,86
10 18,20,      0,33,      3b,1df,     27f,1a5,      249,0
11 10,20,      0,07e,     60,1df,     27f,158,      21b,0
12 20,18,      8,0,      16,1df,     26a,1df,      27f,0

```

A.3.5 pixels_kept.v

```

1  /*
2  Copyright (C) {2014} {Ganesh Ajjanagadde} <gajjanagadde@gmail.com>
3
4  This program is free software: you can redistribute it and/or modify
5  it under the terms of the GNU General Public License as published by
6  the Free Software Foundation, either version 3 of the License, or
7  (at your option) any later version.
8

```



```

55 assign sy2 = {1'b0, y2};
56 assign sy3 = {1'b0, y3};
57 assign sy4 = {1'b0, y4};
58
59 // difference terms
60 assign d_x1_x3 = sx1 - sx3;
61 assign d_x2_x4 = sx2 - sx4;
62 assign d_y1_y3 = sy1 - sy3;
63 assign d_y2_y4 = sy2 - sy4;
64
65 // multipliers
66 assign prod0 = d_x1_x3 * d_y2_y4;
67 assign prod1 = d_y1_y3 * d_x2_x4;
68
69 // final area calculation
70 assign prod = prod0 - prod1; // this is twice the area
71
72 // but first, we need to take its absolute value
73 assign abs_prod = (prod < 0) ? -prod : prod;
74 assign unsigned_prod = abs_prod;
75
76 // to compute the percentage of pixels covered, here is the calculation
77 // we want (100*A)/(640*480), or A/(64*48)
78 // what we have is temp=2*A
79 // thus, we need temp/(128*48) = temp/(6144) = temp/(2^11 * 3) = (temp >> 11) / 3
80 // to avoid the division by 3, we approximate 3 ~ 21/64 (accurate to
81 // within 1%)
82 // thus, we want ((temp >> 11)*21) >> 6
83 // but mult by 21 is same as mult by (16 + 4 + 1)
84 // thus, our final calculation is ((temp >> 7) + (temp >> 9) + (temp >> 11))>>6
85 assign shift_prod_7 = unsigned_prod >> 7;
86 assign shift_prod_9 = unsigned_prod >> 9;
87 assign shift_prod_11 = unsigned_prod >> 11;
88 assign sum_shift_prod = shift_prod_7 + shift_prod_9 + shift_prod_11;
89 assign percent_kept = sum_shift_prod >> 6;
90
91 endmodule

```

A.3.6 bram.v

```

1 /*
2 Copyright (C) {2014} {Ganesh Ajjanagadde} <gajjanagadde@gmail.com>
3
4 This program is free software: you can redistribute it and/or modify
5 it under the terms of the GNU General Public License as published by
6 the Free Software Foundation, either version 3 of the License, or

```



```

55
56 parameter XSPEED = 1'd1;
57 parameter YSPEED = 1'd1;
58
59 // 640 x 480 screen
60 parameter SCR_WIDTH = 10'd639;
61 parameter SCR_HEIGHT = 9'd479;
62
63 reg cur_state = ~OVERRIDE;
64
65 always @(posedge clk) begin
66     case (switch)
67         2'b00: begin
68             display_x <= x1;
69             display_y <= y1;
70         end
71         2'b01: begin
72             display_x <= x2;
73             display_y <= y2;
74         end
75         2'b10: begin
76             display_x <= x3;
77             display_y <= y3;
78         end
79         2'b11: begin
80             display_x <= x4;
81             display_y <= y4;
82         end
83     endcase
84 end
85
86 always @(posedge clk) begin
87     if (override && !(cur_state == OVERRIDE)) begin
88         cur_state <= OVERRIDE;
89         x1 <= x1_raw;
90         y1 <= y1_raw;
91         x2 <= x2_raw;
92         y2 <= y2_raw;
93         x3 <= x3_raw;
94         y3 <= y3_raw;
95         x4 <= x4_raw;
96         y4 <= y4_raw;
97     end
98     else if (override) begin
99         case (switch)
100             2'b00: begin

```

```

101         if (down) begin
102             y1 <= (y1 <= SCR_HEIGHT-YSPEED) ? (y1 + YSPEED) : y1;
103         end
104         else if (up) begin
105             y1 <= (y1 >= YSPEED) ? (y1 - YSPEED) : y1;
106         end
107         else if (left) begin
108             x1 <= (x1 >= XSPEED) ? (x1 - XSPEED) : x1;
109         end
110         else if (right) begin
111             x1 <= (x1 <= SCR_WIDTH-XSPEED) ? (x1 + XSPEED) : x1;
112         end
113     end
114     2'b01: begin
115         if (down) begin
116             y2 <= (y2 <= SCR_HEIGHT-YSPEED) ? (y2 + YSPEED) : y2;
117         end
118         else if (up) begin
119             y2 <= (y2 >= YSPEED) ? (y2 - YSPEED) : y2;
120         end
121         else if (left) begin
122             x2 <= (x2 >= XSPEED) ? (x2 - XSPEED) : x2;
123         end
124         else if (right) begin
125             x2 <= (x2 <= SCR_WIDTH-XSPEED) ? (x2 + XSPEED) : x2;
126         end
127     end
128     2'b10: begin
129         if (down) begin
130             y3 <= (y3 <= SCR_HEIGHT-YSPEED) ? (y3 + YSPEED) : y3;
131         end
132         else if (up) begin
133             y3 <= (y3 >= YSPEED) ? (y3 - YSPEED) : y3;
134         end
135         else if (left) begin
136             x3 <= (x3 >= XSPEED) ? (x3 - XSPEED) : x3;
137         end
138         else if (right) begin
139             x3 <= (x3 <= SCR_WIDTH-XSPEED) ? (x3 + XSPEED) : x3;
140         end
141     end
142     2'b11: begin
143         if (down) begin
144             y4 <= (y4 <= SCR_HEIGHT-YSPEED) ? (y4 + YSPEED) : y4;
145         end
146         else if (up) begin

```



```

19 // rectangle to the quadrilateral inside it
20 // Note that this is the forward mapping
21 // The math is described as follows
22 // Let (x1, y1), (x2, y2), (x3, y3), (x4, y4) be the four points inside the
23 // screen
24 // Let the (forward) perspective map be given by:
25 // (X, Y) = ((p1*x + p2*y + p3)/(p7*x + p8*y + p9), (p4*x + p5*y + p6)/(p7*x
26 // + p8*y + p9))
27 // Then our task is to determine the values of p_i given the values of the x_i
28 // This is a system of equations in 8 unknowns
29 // It turns out that a pretty simple closed form solution exists, given by
30 //
31 // p7 = 3((x1-x4)(y2-y3) + 3(y1-y4)(x3-x2))
32 // p8 = 4((x1-x2)(y3-y4) + 4(x4-x3)(y1-y2))
33 // denom = x4(y2-y3) + x2(y3-y4) + x3(y4-y2)
34 // p9 = 1920*denom (2^7 * 15 * denom)
35 // p3 = 1920*x1*denom (2^7 * 15 * x1 * denom)
36 // p6 = 1920*y1*denom (2^7 * 15 * y1 * denom)
37 // p1 = x4*p7 + 3(x4-x1)*denom
38 // p2 = x2*p8 + 4(x2-x1)*denom
39 // p4 = y4*p7 + 3(y4-y1)*denom
40 // p5 = y2*p8 + 4(y2-y1)*denom
41 //
42 // inverse mapping
43 // p1_inv = p6*p8 - p5*p9
44 // p2_inv = p2*p9 - p3*p8
45 // p3_inv = p3*p5 - p2*p6
46 // p4_inv = p4*p9 - p6*p7
47 // p5_inv = p3*p7 - p1*p9
48 // p6_inv = p1*p6 - p3*p4
49 // p7_inv = p5*p7 - p4*p8
50 // p8_inv = p1*p8 - p2*p7
51 // p9_inv = p2*p4 - p1*p5
52 // dec_numx_horiz = p1_inv * 639
53 // dec_numy_horiz = p4_inv * 639
54 // dec_denom_horiz = p7_inv * 639
55 //
56 // Future improvements:
57 // 1)
58 // This module uses over 120 out of 144 available 18x18
59 // multipliers!!!
60 // By reducing bitwidths and avoiding needless multiplies, e.g shifting
61 // whenever multiplying by constant, resource utilization could be improved
62 // Even with those improvements, I estimate the need of at least 80-100 18x18
63 // multipliers to avoid precision loss
64 //

```



```

111 // difference values for computation
112 wire signed[10:0] d_x1_x2,d_x2_x3,d_x3_x4,d_x4_x1;
113 wire signed[9:0] d_y1_y2, d_y2_y3, d_y3_y4, d_y4_y1, d_y4_y2;
114 assign d_x1_x2 = sx1 - sx2;
115 assign d_x2_x3 = sx2 - sx3;
116 assign d_x3_x4 = sx3 - sx4;
117 assign d_x4_x1 = sx4 - sx1;
118 assign d_y1_y2 = sy1 - sy2;
119 assign d_y2_y3 = sy2 - sy3;
120 assign d_y3_y4 = sy3 - sy4;
121 assign d_y4_y1 = sy4 - sy1;
122 assign d_y4_y2 = sy4 - sy2;
123
124 // computation of p7, p8
125 wire signed[20:0] num0, num1, num2, num3;
126 wire signed[21:0] p7_temp, p8_temp;
127 wire signed[23:0] p7, p8;
128 assign num0 = -(d_x4_x1 * d_y2_y3);
129 assign num1 = d_y4_y1 * d_x2_x3;
130 assign num2 = d_x1_x2 * d_y3_y4;
131 assign num3 = -(d_x3_x4 * d_y1_y2);
132 assign p7_temp = num0 + num1;
133 assign p8_temp = num2 + num3;
134 assign p7 = (p7_temp <<< 1) + p7_temp;
135 assign p8 = (p8_temp <<< 2);
136
137 // computation of denom
138 wire signed[20:0] denom0, denom1, denom2;
139 wire signed[21:0] denom;
140 assign denom0 = sx4 * d_y2_y3;
141 assign denom1 = sx2 * d_y3_y4;
142 assign denom2 = sx3 * d_y4_y2;
143 assign denom = denom0 + denom1 + denom2;
144
145 // computation of p3, p6, p9
146 // observe that 1920 = 2^7 * 15
147 wire signed[25:0] denom_15;
148 wire signed[32:0] p9;
149 wire signed[32:0] x1_denom;
150 wire signed[36:0] x1_denom_15;
151 wire signed[43:0] p3;
152 wire signed[31:0] y1_denom;
153 wire signed[35:0] y1_denom_15;
154 wire signed[42:0] p6;
155 assign denom_15 = (denom <<< 4) - denom; // denom * 15
156 assign p9 = denom_15 <<< 7; // denom * 1920

```



```

157 assign x1_denom = sx1 * denom; // x1 * denom
158 assign x1_denom_15 = (x1_denom <<< 4) - x1_denom; // x1 * denom * 15
159 assign p3 = x1_denom_15 <<< 7; // x1 * denom * 1920
160 assign y1_denom = sy1 * denom; // y1 * denom
161 assign y1_denom_15 = (y1_denom <<< 4) - y1_denom; // y1 * denom * 15
162 assign p6 = y1_denom_15 <<< 7; // y1 * denom * 1920
163
164 // computation of p1, p2, p4, p5
165 wire signed[32:0] d_x1_x2_denom;
166 wire signed[32:0] d_x4_x1_denom;
167 wire signed[31:0] d_y4_y1_denom;
168 wire signed[31:0] d_y1_y2_denom;
169 wire signed[34:0] d_x1_x2_denom_scale;
170 wire signed[34:0] d_x4_x1_denom_scale;
171 wire signed[33:0] d_y4_y1_denom_scale;
172 wire signed[33:0] d_y1_y2_denom_scale;
173 wire signed[34:0] x4_p7;
174 wire signed[34:0] x2_p8;
175 wire signed[33:0] y4_p7;
176 wire signed[33:0] y2_p8;
177 wire signed[35:0] p1, p2;
178 wire signed[34:0] p4, p5;
179 assign d_x1_x2_denom = d_x1_x2 * denom;
180 assign d_x4_x1_denom = d_x4_x1 * denom;
181 assign d_y4_y1_denom = d_y4_y1 * denom;
182 assign d_y1_y2_denom = d_y1_y2 * denom;
183 assign d_x4_x1_denom_scale = (d_x4_x1_denom <<< 1) + d_x4_x1_denom; // d_x4_x1_denom*3
184 assign d_x1_x2_denom_scale = (d_x1_x2_denom <<< 2); // d_x1_x2_denom*4
185 assign d_y4_y1_denom_scale = (d_y4_y1_denom <<< 1) + d_y4_y1_denom; // d_y4_y1_denom*3
186 assign d_y1_y2_denom_scale = (d_y1_y2_denom <<< 2); // d_y1_y2_denom*4
187 assign x4_p7 = sx4 * p7;
188 assign x2_p8 = sx2 * p8;
189 assign y4_p7 = sy4 * p7;
190 assign y2_p8 = sy2 * p8;
191 assign p1 = x4_p7 + d_x4_x1_denom_scale;
192 assign p2 = x2_p8 - d_x1_x2_denom_scale;
193 assign p4 = y4_p7 + d_y4_y1_denom_scale;
194 assign p5 = y2_p8 - d_y1_y2_denom_scale;
195
196 // 36, 36, 44, 35, 35, 43, 24, 24, 33
197 // computation of inverse mapping
198 wire signed[67:0] p1_inv_wire;
199 wire signed[68:0] p2_inv_wire;
200 wire signed[78:0] p3_inv_wire;
201 wire signed[67:0] p4_inv_wire;
202 wire signed[68:0] p5_inv_wire;

```

```

203 wire signed[78:0] p6_inv_wire;
204 wire signed[58:0] p7_inv_wire;
205 wire signed[59:0] p8_inv_wire;
206 wire signed[70:0] p9_inv_wire;
207 assign p1_inv_wire = p6*p8 - p5*p9;
208 assign p2_inv_wire = p2*p9 - p3*p8;
209 assign p3_inv_wire = p3*p5 - p2*p6;
210 assign p4_inv_wire = p4*p9 - p6*p7;
211 assign p5_inv_wire = p3*p7 - p1*p9;
212 assign p6_inv_wire = p1*p6 - p3*p4;
213 assign p7_inv_wire = p5*p7 - p4*p8;
214 assign p8_inv_wire = p1*p8 - p2*p7;
215 assign p9_inv_wire = p2*p4 - p1*p5;
216
217 // computation of dec_numx_horiz, dec_nумы_horiz, dec_denom_horiz
218 wire signed[78:0] dec_numx_horiz_wire;
219 wire signed[78:0] dec_nумы_horiz_wire;
220 wire signed[70:0] dec_denom_horiz_wire;
221 // multiply stuff by 639 = 512 + 128 - 1
222 assign dec_numx_horiz_wire = (p1_inv_wire <<< 9) + (p1_inv_wire <<< 7) - p1_inv_wire;
223 assign dec_nумы_horiz_wire = (p4_inv_wire <<< 9) + (p4_inv_wire <<< 7) - p4_inv_wire;
224 assign dec_denom_horiz_wire = (p7_inv_wire <<< 9) + (p7_inv_wire <<< 7) - p7_inv_wire;
225
226 always @(posedge clk) begin
227     p1_inv <= p1_inv_wire;
228     p2_inv <= p2_inv_wire;
229     p3_inv <= p3_inv_wire;
230     p4_inv <= p4_inv_wire;
231     p5_inv <= p5_inv_wire;
232     p6_inv <= p6_inv_wire;
233     p7_inv <= p7_inv_wire;
234     p8_inv <= p8_inv_wire;
235     p9_inv <= p9_inv_wire;
236     dec_numx_horiz <= dec_numx_horiz_wire;
237     dec_nумы_horiz <= dec_nумы_horiz_wire;
238     dec_denom_horiz <= dec_denom_horiz_wire;
239 end
240
241 endmodule

```

A.3.11 pixel_map.v

```

1  /*
2  Copyright (C) {2014} {Ganesh Ajjanagadde} <gajjanagadde@gmail.com>
3
4  This program is free software: you can redistribute it and/or modify

```



```

51         input[11:0] pixel_in,
52         output reg[11:0] pixel_out,
53         output[16:0] ntsc_out_addr,
54         output reg vga_in_wr,
55         output[16:0] vga_in_addr);
56
57     // internal registers for numerator and denominator computation
58     // see perspective_params.v for the equations
59     reg signed[78:0] num_x = 0;
60     reg signed[78:0] num_y = 0;
61     reg signed[78:0] denom = 0;
62
63     // internal registers for pixel index
64     reg[9:0] cur_x = 0;
65     reg[9:0] cur_y = 0;
66
67     // divider outputs
68     wire signed[78:0] inv_x;
69     wire signed[78:0] inv_y;
70     wire signed[78:0] dummy_remx;
71     wire signed[78:0] dummy_remy;
72     reg div_start;
73     wire div_done_x;
74     wire div_done_y;
75
76     // instantiate dividers
77     divider #(.WIDTH(79)) divider_x(.clk(clk),
78                                     .sign(1'b1),
79                                     .start(div_start),
80                                     .dividend(num_x),
81                                     .divider(denom),
82                                     .quotient(inv_x),
83                                     .remainder(dummy_remx),
84                                     .ready(div_done_x));
85
86     divider #(.WIDTH(79)) divider_y(.clk(clk),
87                                     .sign(1'b1),
88                                     .start(div_start),
89                                     .dividend(num_y),
90                                     .divider(denom),
91                                     .quotient(inv_y),
92                                     .remainder(dummy_remy),
93                                     .ready(div_done_y));
94
95     // instantiate an address mapper (for the vga_in)
96     addr_map addr_map_vga(.hcount(cur_x),

```

```

97         .vcount(cur_y),
98         .addr(vga_in_addr));
99
100 // instantiate an address mapper (for the ntsc_out)
101 addr_map addr_map_ntsc(.hcount(inv_x[9:0]),
102                       .vcount(inv_y[9:0]),
103                       .addr(ntsc_out_addr));
104
105 parameter NEXT_PIXEL_ST = 2'b00;
106 parameter WAIT_FOR_DIV_ST = 2'b01;
107 parameter WAIT_FOR_MEM_ST = 2'b10;
108 parameter BLACK = 12'd0;
109 reg[1:0] cur_state = NEXT_PIXEL_ST;
110 always @(posedge clk) begin
111     case (cur_state)
112         NEXT_PIXEL_ST: begin
113             vga_in_wr <= 0;
114             div_start <= 1;
115             cur_state <= WAIT_FOR_DIV_ST;
116             if ((cur_x == 639) && (cur_y == 479)) begin
117                 cur_x <= 0;
118                 cur_y <= 0;
119                 num_x <= p3_inv;
120                 num_y <= p6_inv;
121                 denom <= p9_inv;
122             end
123             else if ((cur_x == 639) && (cur_y != 479)) begin
124                 cur_x <= 0;
125                 cur_y <= cur_y + 1;
126                 num_x <= num_x - dec_numx_horiz + p2_inv;
127                 num_y <= num_y - dec_namy_horiz + p5_inv;
128                 denom <= denom - dec_denom_horiz + p8_inv;
129             end
130             else if (cur_x != 639) begin
131                 cur_x <= cur_x + 1;
132                 cur_y <= cur_y;
133                 num_x <= num_x + p1_inv;
134                 num_y <= num_y + p4_inv;
135                 denom <= denom + p7_inv;
136             end
137         end
138
139         WAIT_FOR_DIV_ST: begin
140             vga_in_wr <= 0;
141             div_start <= 0;
142             if (div_done_x == 1) begin

```

```

143         cur_state <= WAIT_FOR_MEM_ST;
144     end
145 end
146
147 WAIT_FOR_MEM_ST: begin
148     if ((inv_x < 0) || (inv_x > 639) || (inv_y < 0) || (inv_y > 479)) begin
149         pixel_out <= BLACK;
150         vga_in_wr <= 1;
151         cur_state <= NEXT_PIXEL_ST;
152     end
153     else begin
154         pixel_out <= pixel_in;
155         vga_in_wr <= 1;
156         cur_state <= NEXT_PIXEL_ST;
157     end
158 end
159 endcase
160 end
161 endmodule

```

A.3.12 audioManager.v

```

1  /*
2  Copyright (C) {2014} {Shaun Jain} <shaunjain.08@gmail.com>
3
4  This program is free software: you can redistribute it and/or modify
5  it under the terms of the GNU General Public License as published by
6  the Free Software Foundation, either version 3 of the License, or
7  (at your option) any later version.
8
9  This program is distributed in the hope that it will be useful,
10 but WITHOUT ANY WARRANTY; without even the implied warranty of
11 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12 GNU General Public License for more details.
13
14 You should have received a copy of the GNU General Public License
15 along with this program. If not, see <http://www.gnu.org/licenses/>.
16 */
17 `default_nettype none
18 // Receives audio samples via FTDI UM245R USB-to-FIFO, stores to
19 // onboard flash memory. Plays back the percentage of pixels used
20 // when audioTrigger is pulsed, where the percent is set by
21 // audioSelector. Internally it queues upto four separate tracks
22 // to be played. To save memory and for a faster transfer, the
23 // system constructs all numbers 1-100 out of a subset of the
24 // digits.

```

```

25
26 module audioManager(
27     input wire clock, // 27mhz system clock
28     input wire reset, // 1 to reset to initial state
29
30     // User I/O
31     input wire startSwitch,
32     input wire [6:0] audioSelector,
33     input wire writeSwitch, // 1=Write, 0=Read
34     output wire [63:0] hexdisp,
35     input wire audioTrigger, // 1=Begin Playback as determined by audioSelector
36
37     // AC97 I/O
38     input wire ready, // 1 when AC97 data is available
39     input wire [7:0] from_ac97_data, // 8-bit PCM data from mic
40     output reg [7:0] to_ac97_data, // 8-bit PCM data to headphone
41
42     // Flash I/O
43     output wire [15:0] flash_data,
44     output wire [23:0] flash_address,
45     output wire flash_ce_b,
46     output wire flash_oe_b,
47     output wire flash_we_b,
48     output wire flash_reset_b,
49     output wire flash_byte_b,
50     input wire flash_sts,
51     output wire busy,
52
53     // USB I/O
54     input wire [7:0] data, // the data pins from the USB fifo
55     input wire rxf, // the rxf pin from the USB fifo
56     output wire rd // the rd pin from the USB fifo (OUTPUT)
57 );
58
59 // Playback addresses:
60 parameter TRACK_LENGTH = 69000; // approx 1 sec
61
62 parameter ONE_INDEX = 23'd0;
63 parameter TWO_INDEX = 23'd1;
64 parameter THREE_INDEX = 23'd2;
65 parameter FOUR_INDEX = 23'd3;
66 parameter FIVE_INDEX = 23'd4;
67 parameter SIX_INDEX = 23'd5;
68 parameter SEVEN_INDEX = 23'd6;
69 parameter EIGHT_INDEX = 23'd7;
70 parameter NINE_INDEX = 23'd8;

```

```

71 parameter TEN_INDEX = 23'd9;
72 parameter ELEVEN_INDEX = 23'd10; // A
73 parameter TWELVE_INDEX = 23'd11; // B
74 parameter THIRTEEN_INDEX = 23'd12; // C
75 parameter FOURTEEN_INDEX = 23'd13; // D
76 parameter FIFTEEN_INDEX = 23'd14; // E
77 parameter TWENTY_INDEX = 23'd15; // F
78 parameter THIRTY_INDEX = 23'd16; // 10
79 parameter FORTY_INDEX = 23'd17; // 11
80 parameter FIFTY_INDEX = 23'd18; // 12
81 parameter SIXTY_INDEX = 23'd19; // 13
82 parameter SEVENTY_INDEX = 23'd20; // 14
83 parameter EIGHTY_INDEX = 23'd21; // 15
84 parameter NINETY_INDEX = 23'd22; // 16
85 parameter HUNDRED_INDEX = 23'd23; // 17
86 parameter TEEN_INDEX = 23'd24; // 18
87 parameter PERCENT_INDEX = 23'd25; // 19
88 parameter USED_INDEX = 23'd26; // 1A
89 parameter HELP_AUDIO_INDEX = 23'd27; // 1B
90 parameter SKIP_INDEX = 23'd28; // 1C
91 parameter UNUSED_INDEX = 23'd31; // 1F
92
93 reg writemode = 0; // 1=write mode; 0=read mode
94 reg [15:0] wdata = 0; // writeData
95 reg dowrite = 0; // 1=new data, write it
96 reg [22:0] raddr = 2; // readAddress
97 wire [15:0] frdata; // readData
98 reg doread = 0; // 1=execute read
99
100 flash_manager fm(
101     .clock(clock),
102     .reset(reset),
103
104     // Interface I/O
105     .writemode(writemode),
106     .wdata(wdata),
107     .dowrite(dowrite),
108     .raddr(raddr),
109     .frdata(frdata),
110     .doread(doread),
111     .busy(busy),
112
113     // Flash I/O
114     .flash_data(flash_data),
115     .flash_address(flash_address),
116     .flash_ce_b(flash_ce_b),

```



```

117     .flash_oe_b(flash_oe_b),
118     .flash_we_b(flash_we_b),
119     .flash_reset_b(flash_reset_b),
120     .flash_sts(flash_sts),
121     .flash_byte_b(flash_byte_b)
122 );
123
124 wire [7:0] out; // data from FIFO (OUTPUT)
125 wire newout; // newout=1 out contains new data (OUTPUT)
126 wire hold; // hold=1 the module will not accept new data from the FIFO
127
128 assign hold = 1'b0;
129
130 usb_input usbttest(
131     .clk(clock),
132     .reset(reset),
133
134     // USB FTDI I/O
135     .data(data[7:0]),
136     .rxf(rxf),
137     .rd(rd),
138
139     // Interface
140     .out(out[7:0]),
141     .newout(newout),
142     .hold(hold)
143 );
144
145 wire [3:0] hundreds;
146 wire [3:0] tens;
147 wire [3:0] ones;
148
149 BCD inputToBCD(
150     .number({1'b0, audioSelector}),
151     .hundreds(hundreds),
152     .tens(tens),
153     .ones(ones)
154 );
155
156 reg lastAudioTrigger;
157 reg [2:0] third = 0;
158 reg lastReady;
159
160 // Set of 4 addresses that represent a playback sequence
161 // First track in bottom 23 bits[22:0]. Last track in top bits [91:68].
162 reg [91:0] playbackSeq = 2;

```

```

163 reg [22:0] trackEndAddr = 0;
164 reg playing = 0;
165 reg lastPlaying = 0;
166 reg [15:0] bytesRxed = 0;
167
168 assign hexdisp = {playbackSeq[30:23], playbackSeq[7:0], 1'h0 ,trackEndAddr, 1'h0, raddr[22:0]};
169
170 reg [7:0] dataFromFifo;
171 always @ (posedge rd) begin
172     dataFromFifo <= out; // out & data have same results
173 end
174
175 always @ (posedge clock) begin
176     lastAudioTrigger <= audioTrigger;
177     lastReady <= ready;
178     lastPlaying <= playing;
179
180     if (startSwitch) begin
181         // write USB RX data if switch is up
182         if (writeSwitch) begin
183             writemode <= 1'b1;
184             doread <= 1'b0;
185             //dowrite <= 1'b0; // only write on new data // WATCH OUT!!
186             if (newout) begin
187                 bytesRxed <= bytesRxed + 1;
188                 wdata <= {dataFromFifo, 8'b0};/{out, 8'b0};
189                 dowrite <= 1'b1;
190             end
191         end
192
193         // if button is DOWN - scroll through addresses via buttons
194         if (~writeSwitch) begin
195             dowrite <= 1'b0;
196             writemode <= 1'b0;
197             doread <= 1'b1;
198
199             if (playing & ready) begin // REMOVE audioTrigger
200                 if (raddr < trackEndAddr) begin
201                     // Normal 48K Playback
202                     raddr <= raddr + 1;
203                     to_ac97_data <= frdata[15:8]; // PUT BACK
204                 end
205                 else begin
206                     if (playbackSeq[45:23] < UNUSED_INDEX) begin
207                         // change raddr to next track
208                         raddr <= playbackSeq[45:23] * TRACK_LENGTH;

```

```

209         // shift playbackSeq down
210         playbackSeq <= {UNUSED_INDEX, playbackSeq[91:23]};
211         // update trackEndAddr
212         trackEndAddr <= playbackSeq[45:23] * TRACK_LENGTH + TRACK_LENGTH;
213     end
214     else if (playbackSeq[45:23] == UNUSED_INDEX) begin
215         playing <= 0;
216         raddr <= 0; // reset for safety - lower than UNUSED_ADDR
217     end
218 end
219 end // if (playing & audioTrigger & ready)
220
221 // if entering this state, assign start address
222 if (audioTrigger & ~lastAudioTrigger) begin
223     playing <= 1;
224     case(ones)
225         0: playbackSeq[91:23] <= {UNUSED_INDEX, USED_INDEX, PERCENT_INDEX};
226         1: playbackSeq[91:23] <= {USED_INDEX, PERCENT_INDEX, ONE_INDEX};
227         2: playbackSeq[91:23] <= {USED_INDEX, PERCENT_INDEX, TWO_INDEX};
228         3: playbackSeq[91:23] <= {USED_INDEX, PERCENT_INDEX, THREE_INDEX};
229         4: playbackSeq[91:23] <= {USED_INDEX, PERCENT_INDEX, FOUR_INDEX};
230         5: playbackSeq[91:23] <= {USED_INDEX, PERCENT_INDEX, FIVE_INDEX};
231         6: playbackSeq[91:23] <= {USED_INDEX, PERCENT_INDEX, SIX_INDEX};
232         7: playbackSeq[91:23] <= {USED_INDEX, PERCENT_INDEX, SEVEN_INDEX};
233         8: playbackSeq[91:23] <= {USED_INDEX, PERCENT_INDEX, EIGHT_INDEX};
234         9: playbackSeq[91:23] <= {USED_INDEX, PERCENT_INDEX, NINE_INDEX};
235         default: playbackSeq <= {USED_INDEX, PERCENT_INDEX, UNUSED_INDEX}; // error
236     endcase
237     case (tens)
238         0: playbackSeq[22:0] <= SKIP_INDEX;
239         1: playbackSeq[22:0] <= TEN_INDEX;
240         2: playbackSeq[22:0] <= TWENTY_INDEX;
241         3: playbackSeq[22:0] <= THIRTY_INDEX;
242         4: playbackSeq[22:0] <= FORTY_INDEX;
243         5: playbackSeq[22:0] <= FIFTY_INDEX;
244         6: playbackSeq[22:0] <= SIXTY_INDEX;
245         7: playbackSeq[22:0] <= SEVENTY_INDEX;
246         8: playbackSeq[22:0] <= EIGHTY_INDEX;
247         9: playbackSeq[22:0] <= NINETY_INDEX;
248         default: playbackSeq[22:0] <= UNUSED_INDEX;
249     endcase
250     case (hundreds)
251         0: begin end
252         1: playbackSeq <= {UNUSED_INDEX, USED_INDEX, PERCENT_INDEX, HUNDRED_INDEX}; //
253     endcase
254     case (audioSelector)

```

```

255     11: playbackSeq <= {UNUSED_INDEX, USED_INDEX, PERCENT_INDEX, ELEVEN_INDEX};
256     12: playbackSeq <= {UNUSED_INDEX, USED_INDEX, PERCENT_INDEX, TWELVE_INDEX};
257     13: playbackSeq <= {UNUSED_INDEX, USED_INDEX, PERCENT_INDEX, THIRTEEN_INDEX};
258     14: playbackSeq <= {UNUSED_INDEX, USED_INDEX, PERCENT_INDEX, FOURTEEN_INDEX};
259     15: playbackSeq <= {UNUSED_INDEX, USED_INDEX, PERCENT_INDEX, FIFTEEN_INDEX};
260     16: playbackSeq <= {USED_INDEX, PERCENT_INDEX, TEEN_INDEX, SIX_INDEX};
261     17: playbackSeq <= {USED_INDEX, PERCENT_INDEX, TEEN_INDEX, SEVEN_INDEX};
262     18: playbackSeq <= {USED_INDEX, PERCENT_INDEX, TEEN_INDEX, EIGHT_INDEX};
263     19: playbackSeq <= {USED_INDEX, PERCENT_INDEX, TEEN_INDEX, NINE_INDEX};
264     default: begin end
265     endcase
266     end // if (audioTrigger & ~lastAudioTrigger)
267
268     // just started playing - need to set raddr
269     // Assuming this happens once playbackSeq has been properly set
270     if (playing & ~lastPlaying) begin
271         if (playbackSeq[22:0] == SKIP_INDEX) begin
272             playbackSeq <= {UNUSED_INDEX, playbackSeq[91:23]};
273             raddr <= playbackSeq[45:23] * TRACK_LENGTH;
274             trackEndAddr <= playbackSeq[45:23] * TRACK_LENGTH + TRACK_LENGTH;
275         end
276         else begin
277             raddr <= playbackSeq[22:0] * TRACK_LENGTH;
278             trackEndAddr <= playbackSeq[22:0] * TRACK_LENGTH + TRACK_LENGTH;
279         end
280     end
281     end // if (~writeSwitch)
282 end // if (startSwitch)
283 else begin
284     // TO ENABLE RESET:
285     // writemode <= 1
286     // dowrite <= 0
287     // doread <= 0 // to be safe
288
289     // Reset First, Write Second, Read Later
290     writemode <= 1'h1;
291     doread <= 1'h0;
292     dowrite <= 1'h0;
293 end
294 end // always @
295 endmodule

```

A.3.13 binaryToDecimal.py

```

1 # Shawn Jain
2 # Python script to generate LUT for BCD.v

```

```

3
4 for i in range(100):
5     print str(i) + ': begin ' + 'ones <= ' + str(i%10) + '; ' + 'tens <= ' + str(i/10) +
6 print 'default: begin ones <= 0; tens <= 0; end'

```

A.3.14 BCD.v

```

1 /*
2 Copyright (C) {2014} {Shaun Jain} <shaunjain.08@gmail.com>
3
4 This program is free software: you can redistribute it and/or modify
5 it under the terms of the GNU General Public License as published by
6 the Free Software Foundation, either version 3 of the License, or
7 (at your option) any later version.
8
9 This program is distributed in the hope that it will be useful,
10 but WITHOUT ANY WARRANTY; without even the implied warranty of
11 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12 GNU General Public License for more details.
13
14 You should have received a copy of the GNU General Public License
15 along with this program. If not, see <http://www.gnu.org/licenses/>.
16 */
17 // Converts an 8-bit binary number into a decimal representation
18 // LUT generated from assets/binaryToDecimal.py
19
20 module BCD(
21     input wire [7:0] number,
22     output reg [3:0] hundreds,
23     output reg [3:0] tens,
24     output reg [3:0] ones);
25
26     always @ (number) begin
27         case(number)
28             0: begin ones <= 0; tens <= 0; end
29             1: begin ones <= 1; tens <= 0; end
30             2: begin ones <= 2; tens <= 0; end
31             3: begin ones <= 3; tens <= 0; end
32             4: begin ones <= 4; tens <= 0; end
33             5: begin ones <= 5; tens <= 0; end
34             6: begin ones <= 6; tens <= 0; end
35             7: begin ones <= 7; tens <= 0; end
36             8: begin ones <= 8; tens <= 0; end
37             9: begin ones <= 9; tens <= 0; end
38             10: begin ones <= 0; tens <= 1; end
39             11: begin ones <= 1; tens <= 1; end

```

```
40:   12: begin ones <= 2; tens <= 1; end
41:   13: begin ones <= 3; tens <= 1; end
42:   14: begin ones <= 4; tens <= 1; end
43:   15: begin ones <= 5; tens <= 1; end
44:   16: begin ones <= 6; tens <= 1; end
45:   17: begin ones <= 7; tens <= 1; end
46:   18: begin ones <= 8; tens <= 1; end
47:   19: begin ones <= 9; tens <= 1; end
48:   20: begin ones <= 0; tens <= 2; end
49:   21: begin ones <= 1; tens <= 2; end
50:   22: begin ones <= 2; tens <= 2; end
51:   23: begin ones <= 3; tens <= 2; end
52:   24: begin ones <= 4; tens <= 2; end
53:   25: begin ones <= 5; tens <= 2; end
54:   26: begin ones <= 6; tens <= 2; end
55:   27: begin ones <= 7; tens <= 2; end
56:   28: begin ones <= 8; tens <= 2; end
57:   29: begin ones <= 9; tens <= 2; end
58:   30: begin ones <= 0; tens <= 3; end
59:   31: begin ones <= 1; tens <= 3; end
60:   32: begin ones <= 2; tens <= 3; end
61:   33: begin ones <= 3; tens <= 3; end
62:   34: begin ones <= 4; tens <= 3; end
63:   35: begin ones <= 5; tens <= 3; end
64:   36: begin ones <= 6; tens <= 3; end
65:   37: begin ones <= 7; tens <= 3; end
66:   38: begin ones <= 8; tens <= 3; end
67:   39: begin ones <= 9; tens <= 3; end
68:   40: begin ones <= 0; tens <= 4; end
69:   41: begin ones <= 1; tens <= 4; end
70:   42: begin ones <= 2; tens <= 4; end
71:   43: begin ones <= 3; tens <= 4; end
72:   44: begin ones <= 4; tens <= 4; end
73:   45: begin ones <= 5; tens <= 4; end
74:   46: begin ones <= 6; tens <= 4; end
75:   47: begin ones <= 7; tens <= 4; end
76:   48: begin ones <= 8; tens <= 4; end
77:   49: begin ones <= 9; tens <= 4; end
78:   50: begin ones <= 0; tens <= 5; end
79:   51: begin ones <= 1; tens <= 5; end
80:   52: begin ones <= 2; tens <= 5; end
81:   53: begin ones <= 3; tens <= 5; end
82:   54: begin ones <= 4; tens <= 5; end
83:   55: begin ones <= 5; tens <= 5; end
84:   56: begin ones <= 6; tens <= 5; end
85:   57: begin ones <= 7; tens <= 5; end
```

```

86      58: begin ones <= 8; tens <= 5; end
87      59: begin ones <= 9; tens <= 5; end
88      60: begin ones <= 0; tens <= 6; end
89      61: begin ones <= 1; tens <= 6; end
90      62: begin ones <= 2; tens <= 6; end
91      63: begin ones <= 3; tens <= 6; end
92      64: begin ones <= 4; tens <= 6; end
93      65: begin ones <= 5; tens <= 6; end
94      66: begin ones <= 6; tens <= 6; end
95      67: begin ones <= 7; tens <= 6; end
96      68: begin ones <= 8; tens <= 6; end
97      69: begin ones <= 9; tens <= 6; end
98      70: begin ones <= 0; tens <= 7; end
99      71: begin ones <= 1; tens <= 7; end
100     72: begin ones <= 2; tens <= 7; end
101     73: begin ones <= 3; tens <= 7; end
102     74: begin ones <= 4; tens <= 7; end
103     75: begin ones <= 5; tens <= 7; end
104     76: begin ones <= 6; tens <= 7; end
105     77: begin ones <= 7; tens <= 7; end
106     78: begin ones <= 8; tens <= 7; end
107     79: begin ones <= 9; tens <= 7; end
108     80: begin ones <= 0; tens <= 8; end
109     81: begin ones <= 1; tens <= 8; end
110     82: begin ones <= 2; tens <= 8; end
111     83: begin ones <= 3; tens <= 8; end
112     84: begin ones <= 4; tens <= 8; end
113     85: begin ones <= 5; tens <= 8; end
114     86: begin ones <= 6; tens <= 8; end
115     87: begin ones <= 7; tens <= 8; end
116     88: begin ones <= 8; tens <= 8; end
117     89: begin ones <= 9; tens <= 8; end
118     90: begin ones <= 0; tens <= 9; end
119     91: begin ones <= 1; tens <= 9; end
120     92: begin ones <= 2; tens <= 9; end
121     93: begin ones <= 3; tens <= 9; end
122     94: begin ones <= 4; tens <= 9; end
123     95: begin ones <= 5; tens <= 9; end
124     96: begin ones <= 6; tens <= 9; end
125     97: begin ones <= 7; tens <= 9; end
126     98: begin ones <= 8; tens <= 9; end
127     99: begin ones <= 9; tens <= 9; end
128     default: begin ones <= 0; tens <= 0; end
129     endcase
130     hundreds <= 0;
131 end

```

```

132 endmodule
133
134 // Note: a computational logic based binary to BCD is found at:
135 // http://www.deathbylogic.com/2013/12/binary-to-binary-coded-decimal-bcd-converter/
136
137 module BCDTest;
138     reg [7:0] number = 8'd65;
139     wire [3:0] hundreds;
140     wire [3:0] tens;
141     wire [3:0] ones;
142
143     BCD bc(number, hundreds, tens, ones);
144     initial begin
145         #100
146         $display("%d, %d, %d", hundreds, tens, ones);
147         $stop();
148     end
149 endmodule

```

A.3.15 ClockDivider.v

```

1  /*
2  Copyright (C) {2014} {Shaun Jain} <shaunjain.08@gmail.com>
3
4  This program is free software: you can redistribute it and/or modify
5  it under the terms of the GNU General Public License as published by
6  the Free Software Foundation, either version 3 of the License, or
7  (at your option) any later version.
8
9  This program is distributed in the hope that it will be useful,
10 but WITHOUT ANY WARRANTY; without even the implied warranty of
11 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12 GNU General Public License for more details.
13
14 You should have received a copy of the GNU General Public License
15 along with this program. If not, see <http://www.gnu.org/licenses/>.
16 */
17 // From Lab 4
18 // Sends a pulse on oneHertz_enable every Hz clock cycles
19
20 module ClockDivider #(parameter Hz = 27000000)(
21     input clock, reset, fastMode,
22     output reg oneHertz_enable
23 );
24
25     reg [24:0] counter = 25'b0;

```



```

26
27     always @ (posedge clock) begin
28         if (reset) begin
29             counter <= 25'b0;
30             oneHertz_enable <= 1'b0;
31         end
32         else if (counter == (fastMode ? 3:Hz)) begin
33             oneHertz_enable <= 1'b1;
34             counter <= 25'b0;
35         end
36         else begin
37             counter <= counter + 1;
38             oneHertz_enable <= 1'b0;
39         end
40     end
41
42 endmodule

```

A.3.16 Square.v

```

1  /*
2  Copyright (C) {2014} {Shaun Jain} <shaunjain.08@gmail.com>
3
4  This program is free software: you can redistribute it and/or modify
5  it under the terms of the GNU General Public License as published by
6  the Free Software Foundation, either version 3 of the License, or
7  (at your option) any later version.
8
9  This program is distributed in the hope that it will be useful,
10 but WITHOUT ANY WARRANTY; without even the implied warranty of
11 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12 GNU General Public License for more details.
13
14 You should have received a copy of the GNU General Public License
15 along with this program. If not, see <http://www.gnu.org/licenses/>.
16 */
17 // From Lab 4
18 // Generates a square wave that flips every Hz clock cycles
19
20 module Square #(parameter Hz = 27000000) (
21     input clock, reset,
22     output reg square = 0);
23
24     wire oneHertz_enable;
25
26     ClockDivider #(.Hz(Hz)) Sqr (

```

```
27         .clock(clock),
28         .reset(reset),
29         .fastMode(1'b0),
30         .oneHertz_enable(oneHertz_enable)
31     );
32
33     always @ (posedge oneHertz_enable) begin
34         square <= ~square;
35     end
36 endmodule
```