Music Visualization with Audio Beat-Matching

Final Report

Maggie Reagan and Liz Schell

Table of Contents

1. Introduction and Motivation

The goal for this project is to implement music visualization on a computer monitor using audio beat detection for real-time music playing. Most commercially available applications for this purpose are implemented in software, and are also limited in the visuals they provide. Hardware implementations that exist for this purpose are entirely custom and are even more visually limited; the most common music visualization is done with LED strips. This project would offer a hardware implementation for music beat detection and a more interesting and visually pleasing visualization than currently exists on the market.

We chose this project because we enjoy music and thought that it would be interesting to implement a colorful and unique visualization to go with our favorite music. In order for this project to be useful for a party or social event, we wanted to be able to process a song in real time or as close to real time as possible. We decided to go with an algorithm that takes a 2.2 second sample of a song to start generating the tempo and phase, to allow for beat detection as low as 60 bpm with some wiggle room. As new music samples come in, we will keep calculating the phase and tempo with the new data to keep up with changing tempos.

The development of the audio processing section of this project was done by Liz, and the development of the visualization section was done by Maggie. Therefore, the sections of this paper describing the design details and debugging process of these sections have been written by the person who worked on that section of the project.

2. Implementation Overview

Our project has two major parts: audio beat detection and visual generator. The beat detection is based on an algorithm described in a paper from the Media Lab (“Tempo and beat analysis of acoustic musical signals” by Eric D. Scheirer in 1997). It starts by smoothing the audio signal to make the beats stand out more and then convolving that with various comb filters to determine which tempo the music matches. The beat detection output will be used as an input to a visual display generator that will have several stages. The visual generator creates a series of expanding nested circles originating from the center and corners of the screen, that change in time with the beat of the song. The circles appear to expand until reaching the corners of the screen, at which they reappear in the center of the screen. The speed at which this expansion is performed is correlated with the tempo of the music.

3. Audio Block Implementation

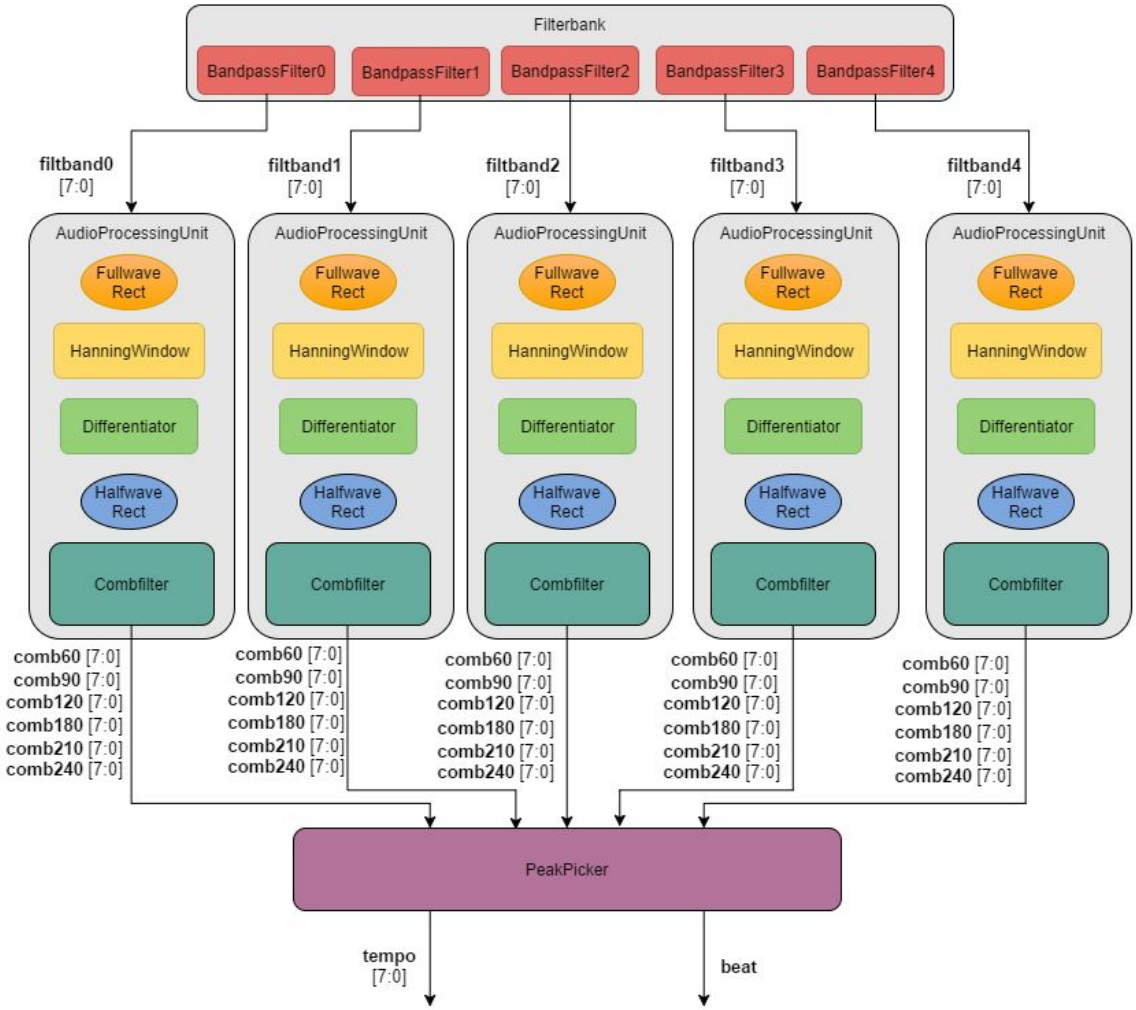

Figure 1: Audio Processing Block Diagram

The purpose of the audio block is to process the input audio and output the tempo and phase of the audio to the visual block. The frequency tells the visual block what the tempo of the music is, and the phase is represented by a signal “beat”, which is high when the beat happens and low otherwise. This will allow the visualization to be synced up with the music. The full block diagram for the Audio Block is shown in Figure 1.

3.1 Filterbank

To achieve these goals, the audio input from the ac97 module is first low passed to 3000 Hz and then sampled at 6000 Hz. The low passed signal is shown in the first row of logic analyzer outputs in Figure 8. Next, this signal is split up into five different frequency bands (0-200,

200-400, 400-800, 800-1600, 1600-3200 Hz) by the frequency filterbank so that they can be processed separately. The frequency responses for each of these filters are shown in Figures 2-6. Each of these different bands is sent to its own audio processing unit. Figure 8 also shows the output from the highest frequency band, filtband4.

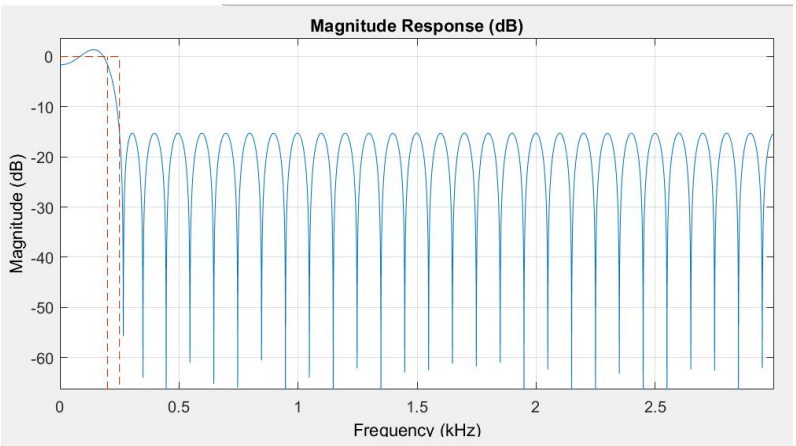

Figure 2: Frequency Response of 200 Hz Low Pass Filter

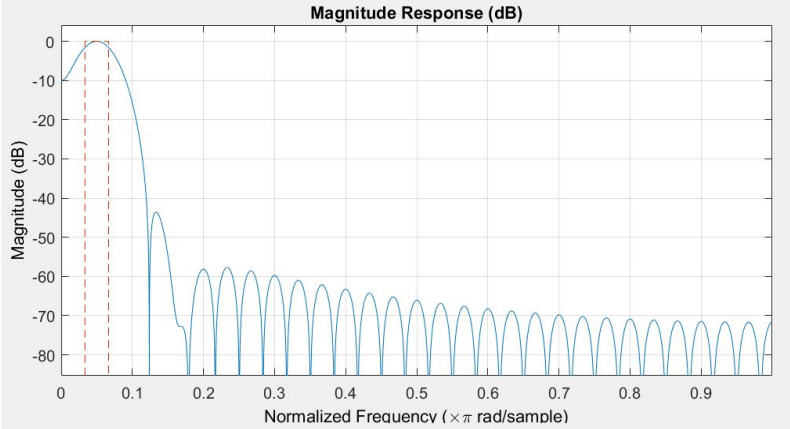

Figure 3: Frequency Response of 200-400 Hz Bandpass Filter

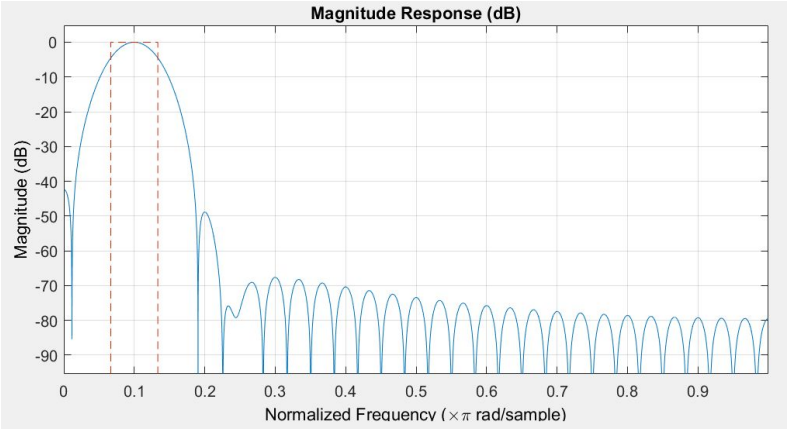

Figure 4: Frequency Response of 400-800 Hz Bandpass Filter

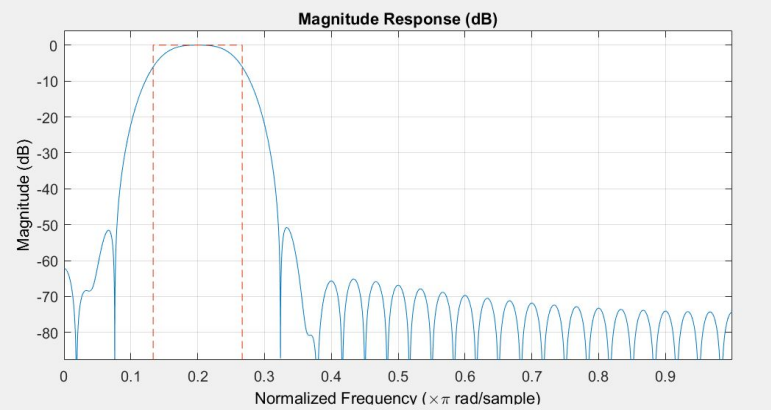

Figure 5: Frequency Response of 800-1600 Hz Bandpass Filter

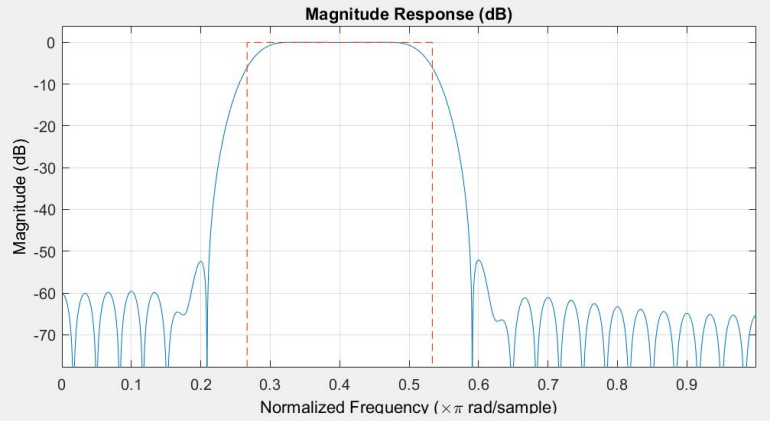

Figure 6: Frequency Response of 1600-3200 Hz Bandpass Filter

3.2 AudioProcessingUnit

The audio processing unit contains an envelope extractor, a differentiator, a half-wave rectifier, and then a comb filter. The envelope extractor is implemented with a full-wave rectifier and then a Hann window, which low passes the signal to around 20 Hz. The Hann window that was used is shown in Figure 7. This window filter smooths the signal to accentuate the amplitude of the sound. The output of this Hann window for the filtband4 signal is shown as hann_out in Figure 8. After the envelope is extracted, the signal is differentiated and passed through a half-wave rectifier. This isolates the large positive jumps in amplitude that tend to happen at the beats. The differentiated signal is shown as diff_out in Figure 8. Once the audio has been processed to bring out the beats of the song, the comb filters act as resonators for different possible frequencies of the audio. This implementation tested the input audio for 60, 90, 120, 180, 210, and 240 bpm. With the sampling rate of 6000 Hz, this required impulses spaced by 6000, 4000, 3000, 2000, 1714, and 1500 samples, respectively. The comb filter is implemented with an instantiation of the bram module that stores the input at each sample. Between samples, it cycles through the different addresses that are needed to tap out the

correct samples for the different comb filters. The samples for a given frequency comb filter are summed and passed on to the peakpicker module.

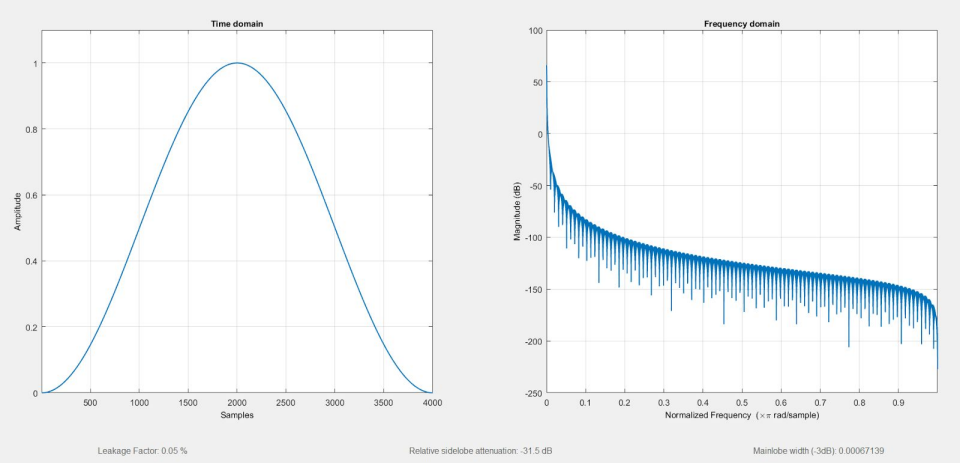

Figure 7: Hann Window in Time and Frequency Domain

3.3 Peakpicker

The peakpicker module receives 30 different signals, which correspond to the six different tempo comb filter outputs for each of the five frequency bands. After the comb filters, the peakpicker module squares the comb filter outputs for a given tempo from all of the five frequency bands and adds those together to find the energy for a given sample at a given tempo. These byte energies are then summed over time to find the total energy of the song at 60 bpm or one of the other tempos. These different summed energies are compared to determine which tempo is the fundamental tempo of the audio. This will be the tempo whose comb filter outputs have the highest energy. The phase can be determined by looking at the energy of the individual samples that are coming in for the selected tempo. These single byte energies can be compared to a predetermined threshold to determine when the beats lined up with the comb filter. This will show when the beat happened with some phase lag between when the audio sample was taken in and when it arrived at this stage of the system.

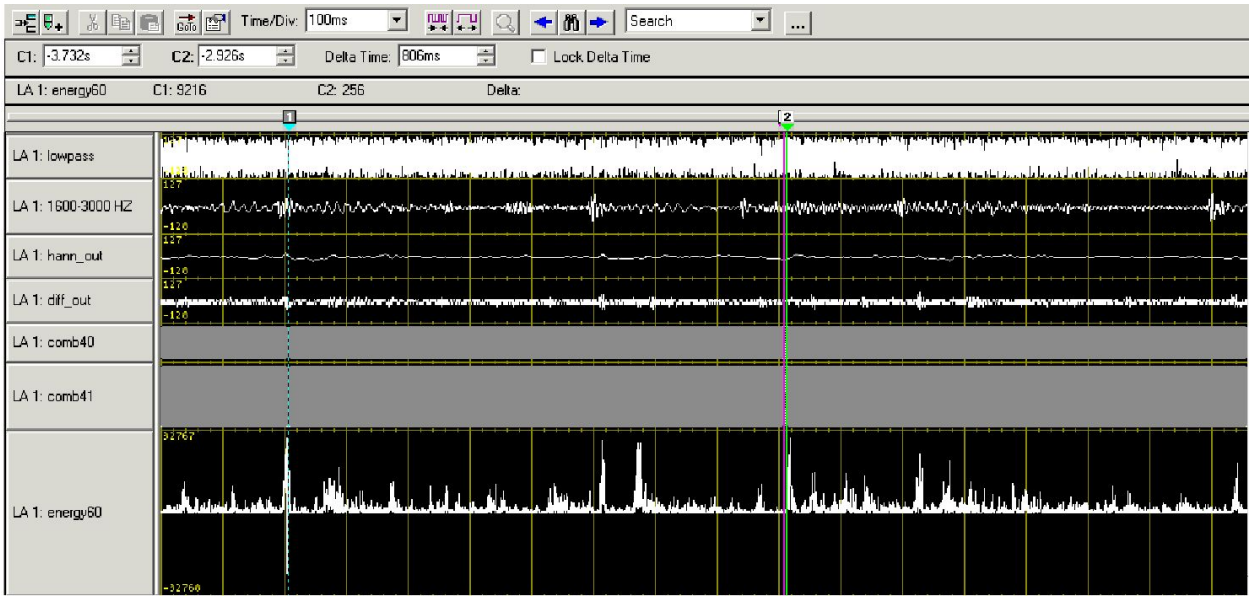

Figure 8: Signals from different stages of the system

3.4 Peakfinder

The Peakpicker module was not functional in time for the project check-off and demo. In order to make the project more functional, I decided to try a different approach at sending information from the Audio Block to the Visual Block. This module cannot determine the fundamental tempo of the song, however it can detect pulses of high energy in the song, which correlate with beats or high amplitudes of sound. This module calculates the byte energy in the same way as the Peakpicker, but depends on the fundamental tempo being input by the user. Given this input tempo, the Peakfinder can look at the byte energies coming from the comb filters of a given tempo, for example 120 bpm, and sets the beat signal high whenever this energy is above a certain threshold energy.

4. Audio Block Obstacles

4.1 Bugs and Fixes

One of the more difficult obstacles of this project was designing the filters. The filters suggested in the paper that this design is based on are sixth order elliptic filters. However, implementing FIR filters is easier, so I decided to try that. Initially, I tried making 31 tap FIR filters, and these filters didn't work for a number of reasons. One reason was that they didn't have enough taps to effectively filter the frequencies that I was trying to filter. The lowest filterband has a cutoff of 200 Hz, which is very small compared to the sampling rate of 6000 Hz. An FIR filter order of at least 60 is needed to filter frequencies this low. Unfortunately just increasing all of my bandpass filters to 61 tap didn't completely fix the problem. The signals appeared to be passing the correct frequencies. However, there was clipping, and the signals

looked like the peaks of the wave were transposed to the bottom of the signal. I discovered that the other problem was that I had forgotten to low pass my incoming audio at 3 kHz before sampling it at 6 kHz. This was introducing aliasing into the signals. This aliasing was fixed by including a 3 kHz low pass filter between the audio input from the ac97 and the one-eighth downsampling that happens as the signal is fed into the filterbank.

Similarly, when I first implemented the low pass Hann window, I also did not make it with enough coefficients. Since the purpose of the Hann window is to act as a low pass filter and find the envelope of the signal, it needs to have a cutoff frequency of around 10 Hz. In order to achieve this low of a cutoff frequency, the filter needs 4800 coefficients. Since the system samples at 6000 Hz and uses a 27 MHz clock, there are 4500 clock cycles between samples which limits the number of coefficients that the filter can have. I decided to use 4000 coefficients which low passed with a cutoff frequency of about 20 Hz. Once I started using filters with so many coefficients, it became impractical to write out the code for the case statement with over 4000 lines. Instead, I wrote a python script to read the coefficients from a file and format them into the case statement.

With these filters of higher orders, there are some considerations that need to be made. When we made the 31 tap FIR filter for lab 5, we had an 8 bit input and an 18 bit accumulator, because the coefficients were all scaled up by 2^{10}. However, with more coefficients, it can become necessary to have a larger accumulator. Without this, the signals clip, because of overflow. It took me several tries to determine how wide the output of the Hann window needed to be in order to see as much of the wave as possible without clipping. I started by calculating the maximum number of bits that could be filled by multiplying the number of coefficients by the highest possible coefficient by the highest possible input. This proved to be much higher than any of the outputs, because many of the beginning and end coefficients are zero or numbers under ten and the maximum possible coefficient is 1024. I found the correct width for the accumulator by shifting the window of eight bits that were displayed on the logic analyzer until the signal's amplitude was as large as possible without clipping.

The comb filter module also presented many problems during development. I implemented the comb filter by convolving the signal over time with evenly spaced impulses. My first idea involved using a shift register to store the past samples so that they could be accessed again at the right time. I thought that this would be easier than using bram, because I wouldn't have to shift the address around to access the samples when I needed them. However, because over two seconds of audio are needed to get 3 impulses at 60 bpm, I needed an 8 bit wide shift register with 13200 registers. Unfortunately, Verilog doesn't allow partial declarations of 2D arrays, and so I couldn't easily assign the past 13199 bytes to the next registers with one line. Next, I tried to use a for loop to assign the byte samples one-by-one to their next registers. That attempt produced an error saying that iterations of greater than 10000 could not be performed. After this, I decided to change tactics and implement the comb filter with bram which introduced its own new set of bugs and difficulties.

I spent about four entire days debugging the comb filter module. During those four days, I found many potential bugs, but I kept getting the same results. All of my comb filters for a given frequency bands kept outputting the same exact signal. Eventually, I decided to change how I wrote to memory. Originally, I was assigning the input to memin and then just pulsing write enable when it was time to write the new input to memory. To fix the problem, I changed memin to a register instead of a wire and set memin to the input at the same time as I set write enable high. I still don't understand exactly why connecting the input to memin and pulsing write enable doesn't work, but I assume it must be a clocking issue where memin is unstable, because it's a wire instead of a reg.

4.1 Testing and Debugging Methods

During early development, I used simulations to in order to have a proof of concept and determine if my modules were working as expected. However, simulations rarely react the same way as implementation on the labkit. Almost all of the debugging was done with actual audio input on the labkit. I displayed the signals on the logic analyser as I stepped through the different stages of the audio block to determine that each module was reacting correctly. These tests were done at different volumes and with different songs in order to get a better understand of the capabilities of the system.

5. Visual Block Modules

Figure 9 is an overall block diagram of the music visualization part of the project. All of the modules were clocked at a 40 MHz clock cycle generated by the ramclock module, a TA-provided module, which was necessary to use the ZBT memory banks.

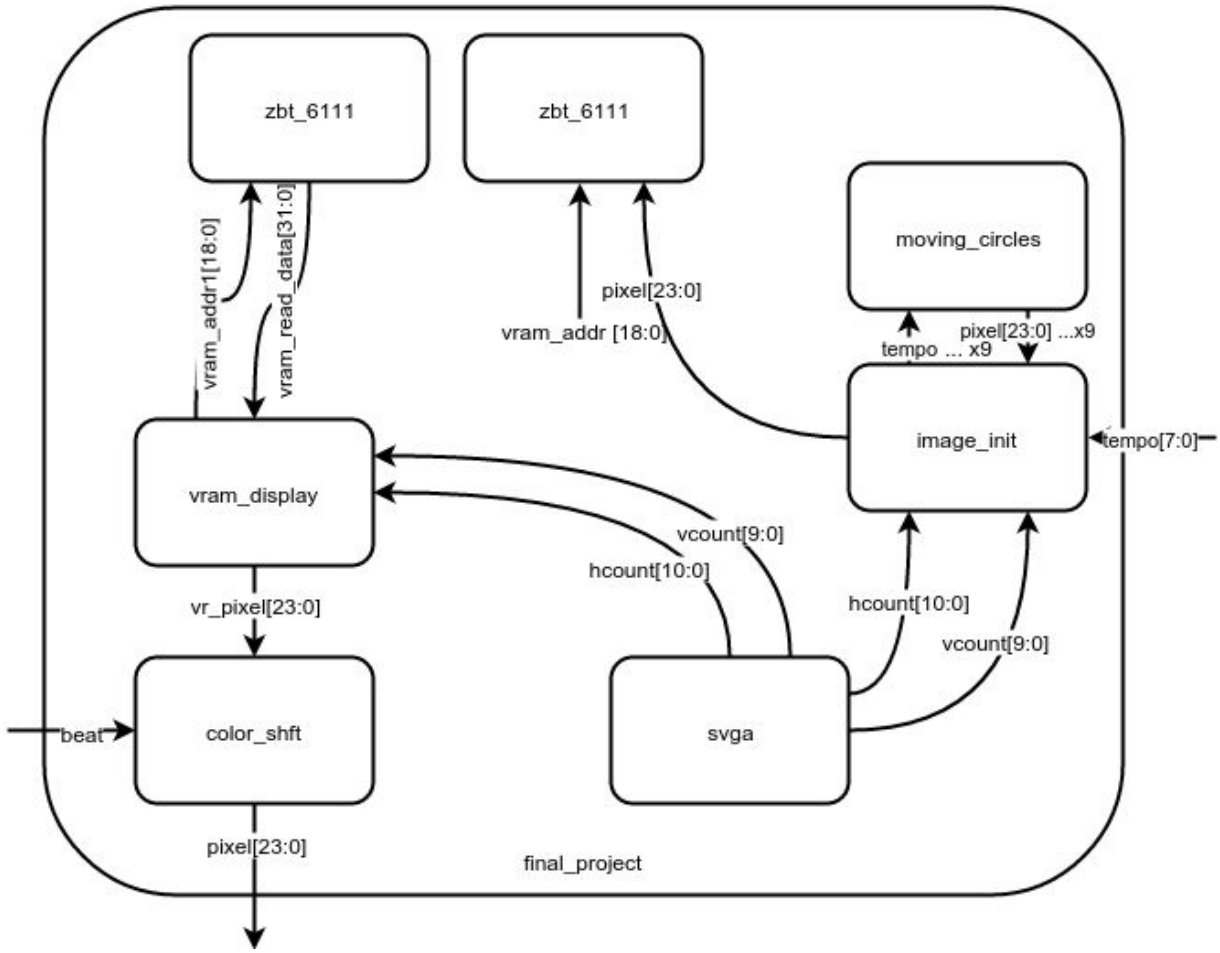

Figure 9: Overall block diagram of visualization modules

5.1 final_project

The top-level module of the project, final_project, contained all of the other visual modules as well as the arbitration between the ZBT SRAM modules, and the connection between the beat detection aspect of the project and the visualization aspect of the project. The arbitration between the ZBTs was such that while the current ZBT was being displayed via the TA-supplied vram_display module, the next ZBT was being written to with the next image to be displayed. The ZBT being displayed from would switch every vsync, or frame refresh, and the next image would be again written to the rendering ZBT which had been previously displayed. In the first form of our project, I had originally been connecting the data from the ZBTs together as well; that is, in addition to the read data of the current ZBT memory bank being displayed, it

was also being written to the rendering ZBT at a different location given by the graphics_location module. This form of ZBT switching was successfully implemented; however, due to timing issues between the signals which will be explained further later on, this did not end up in the final form of our project.

In addition, the top-level module connected the music beat detection part of the project to the visualization part of the project. While connecting the tempo to the visualization blocks was easy, in that tempo was a continuous value output from the audio modules, the beat detection was less so due to the signals being clocked at different speeds. This was solved by a finite state machine which set two states, beat and reset, and switched to the beat state on the beat and only switched to reset state after the vsync had occurred. This was necessary due to the fact that we wanted to catch the beat signal before the frame updated, since they would not be in sync and we wanted to display a color change at the approximate beat time.

5.2 graphics_rotation

The final form of the project did not contain rotation as originally proposed due to several issues that will be explained in detail in this section. Still, a large amount of time was spent on writing and debugging this aspect of the project, and a switch on the labkit allows us to go into "rotation mode", so it merits an explanation. An overall block diagram for the graphics module is shown in Figure 10.

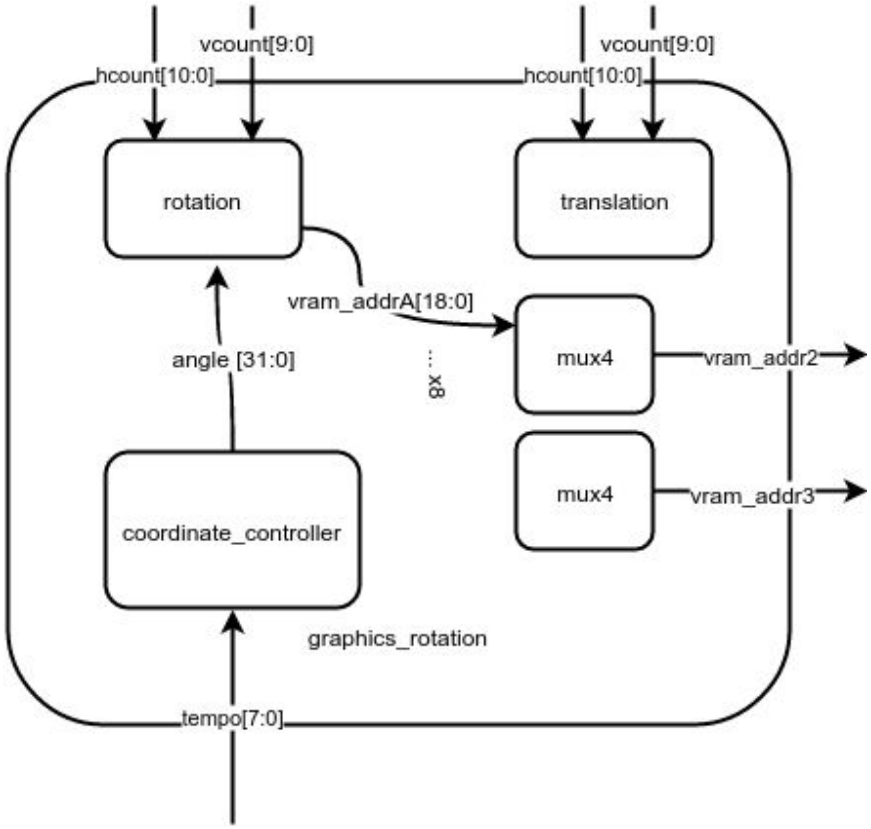

Figure 10: Rotation graphics block diagram.

The graphics_rotation module takes in the current vcount and hcount given by the svga module, and turns them into forecasted hcount and vcount values to be given as x and y coordinates to be given to the rotation module. The rotation module computes the new rotated pixel x and y coordinates, x' and y'. This calculation takes 11 clock cycles, which is why the hcount and vcount must be turned into forecasted hcount and vcount values. These x and y outputs are then turned into an address for the pixel data from the non-rotated x and y coordinate to be written to. In creating this address for the rotated pixel data, I also create 6 other addresses which correspond to pixels surrounding the original pixel: the coordinates (x'-1, y'), (x'+1, y'), (x', y'-1), (x', y'+1), (x'+1, y'+1), (x'-1, y'-1). These addresses are split up and given to two four-input muxes, which select the correct address to output given the address count variable. In the original implementation of the project involving rotation, the ZBT ram was switched only every 4 vsync signals, making the frame rate 15 Hz, and allowing the rendering ZBT to be written over multiple times. In writing data to the rotated pixel location as well as the surrounding pixels over multiple svga cycles, I tried to ensure that the rounding issues caused by the fact that pixels cannot be floating point numbers, and that a rotation without floating point numbers would necessarily be inexact and pixels would get dropped, would be mitigated.

5.2.1 coordinate_controller

The coordinate controller module takes in the tempo of the song and outputs the angle by which the image will be rotated. The tempo is mapped directly to an angle ranging from 1 degree to 16 degrees, not due to the lack of ability of the rotation algorithm to handle larger angles, but because if the image rotates by too large an angle the object will appear to "jump" to the new angle rather than move to the new angle.

5.2.2. rotation

In this block, x and y values are given on a 0 to 800, or 0 to 600 scale according to the hcount and vcount they correspond to. First they are converted to values on a standard axis; that is, values from -400 to 400 and -300 to 300. This was necessary to create rotation about the center of the screen, rather than the top left corner, since the rotation occurs about the origin. Next, the converted x and y coordinates are rotated using the CORDIC rotation algorithm. This algorithm is an iterative algorithm, and basically calculates a rotation based on shifting and addition of x, y, and angle values. The equations for the algorithm are shown below:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$
$$z_{i+1} = z_i - d_i \cdot tan(2^{-i})$$

In this algorithm, z is initialized to the angle which you want to rotate by, and x and y are initialized to the non-rotated x and y coordinate values. For each iteration, d is given by the

sign of the current z value. The algorithm is iterated for the number of bits that can be stored for x and y. In my implementation, I implemented this inside of a generated for loop that incremented i once per clock cycle. The multiplication by 2^{-i} can be implemented by a signed right shift by i, meaning that no actual multiplication is used during the iteration. The tangent values of 2^{-i} were stored in a table within the module, as a 32 bit number, representing the rounded value of $2^{32} \cdot \frac{tan(2^{-i})}{360}$, to allow for the maximal amount of tangent data to be stored. The reason this module takes 11 clock cycles to complete is that the algorithm must iterate 11 times; this is the maximum size of signed x and y, so it doesn't make sense to calculate further than that. Once the rotated values are calculated, they must be divided by a gain of 1.647. Fortunately, the inverse of this is .6072, which can be approximated to .6074, or a multiplication by 311 and a right shift by 9 bits. The x and y rotated coordinates are then transformed back into positive values on the 0-800 and 0-600 scale, and given as the output.

This module worked well in testing; that is, in simulation when given an x, y value and an angle it would rotate to the correct x' and y' values. However, the actual use of this algorithm to rotate an entire image pixel by pixel did not work for reasons described in section 5.2.3.

5.2.3. translation

In the process of testing the integrated visual aspect of these modules, before I implemented the moving circles contingency plan, I discovered it was difficult to tell whether or not the rotating image was staying in the correct place on the screen, and whether or not switching between reading ZBTs was implemented correctly due to the timing issues in the rotation algorithm itself. Therefore the translation module was written to both allow for testing of the arbitration between ZBT modules, and testing of the timing for the vram_display module. The translation module simply takes in an hcount and a vcount, and writes the data at that address to an address that has moved by some distance. Using this module, I was able to test whether or not writing data from the read data of one ZBT to another would work. I addition, I needed to forecast the hcount and vcount in the vram_display module by some amount, since the ZBT takes two clock cycles to be read from. I was able to test whether the timing of this was correct by checking that the square blob did not move when I gave it a distance of zero to move; I knew then that I was forecasting the correct amount by hcount and vcount to be read from the ZBT, as if it was off slightly the read data given to the rendering ZBT would be off by some small number of pixels, causing the blob to appear to drift along the screen.

5.2.4. Obstacles with rotation

The original concept for our project involved rotating nested squares about the center of the screen. While I was able to come up with an image that would do this, the quality of the image was poor due to several reasons. One problem was inherent to the original design idea, and the other was a bug that I was unable to figure out after many hours and days in lab.

This first issue, the one inherent in the design, I had considered beforehand, but hadn't realized it would cause the extent of the issues it did. This was the fact that because I was implementing a rotation of every single pixel on the screen, and the rotation algorithm and Verilog does not allow for floating point pixel values, I would have to round to the nearest neighbor location of the pixel. What this did would cause most of the image to rotate, leaving some pixels behind where they were not overwritten by the background color, and causing pixels of the background color to bleed into the shape as they were not overwritten by the rotated image due to rounding. I knew the rounding would happen to some degree, causing the image to be slightly degraded over time; however, in the implementation I had working this degradation of the image happened extremely quickly, such that the image would not even make it halfway around the screen before disappearing or turning into a shapeless blob.

The next problem I had was one of timing. In writing the translation module, originally written to test the switching between ZBTs was working properly, I noticed that grey lines would appear on the top of the screen as the square blob test image moved down the screen. I figured the appearance of these lines had to be a timing issue somewhere which caused either the wrong pixels to be written to the ZBT or the wrong pixels to be read from the ZBT. After many hours of testing, I discovered the issue could be fixed by manipulating the forecasting of hcount and vcount in the vram_display module; however, the same issue appeared the next day when I recompiled and ran my code. This was the source of much confusion and frustration; after several additional hours of testing I discovered that the timing of the vram_display had changed between compiles, and that the best was to debug to proper timing constraints (i.e. the forecasting of hcount and vcount) I needed to set the travel distance of the square blob to zero and test different forecasting amounts until the image was able to remain stationary on the screen. Figure 11 below shows what this problem looked like when it occurred

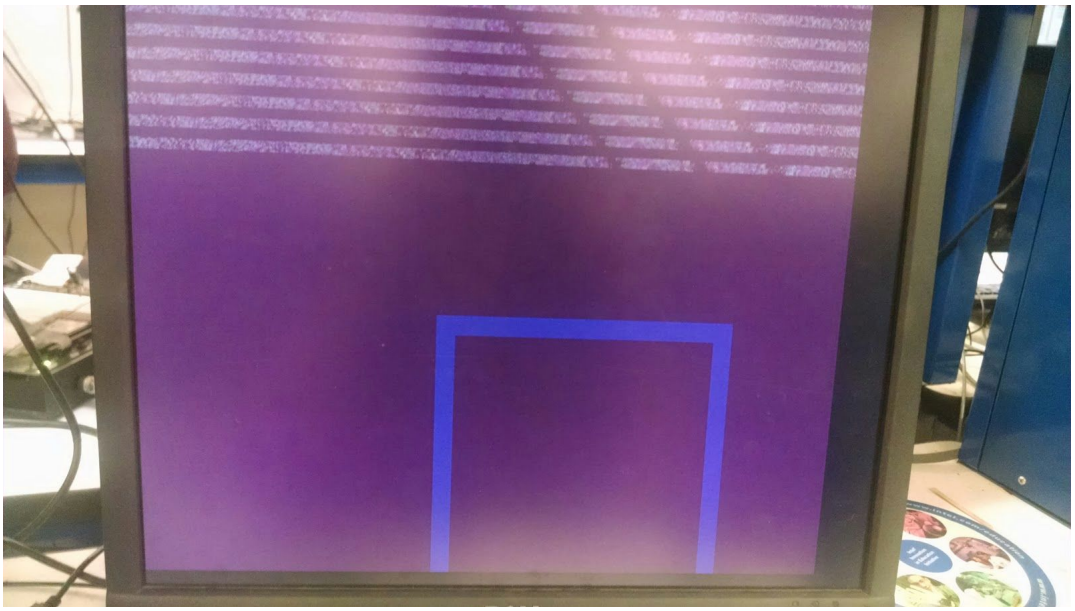

Figure 11: Grey lines caused by timing problems

This same method was used in an attempt to debug the timing issues of the rotation module; I would set the angle of rotation to be zero, and adjust the forecasted vcount and vcount being sent to the rotation module accordingly until the actual image was stationary. One problem I discovered was that despite modifying both the values sent to the rotation block as well as the values to be displayed via the vram_display module, I was unable to get the timing exactly right. Figure 12 shows another test image rotated an angle of zero via the rotation algorithm.

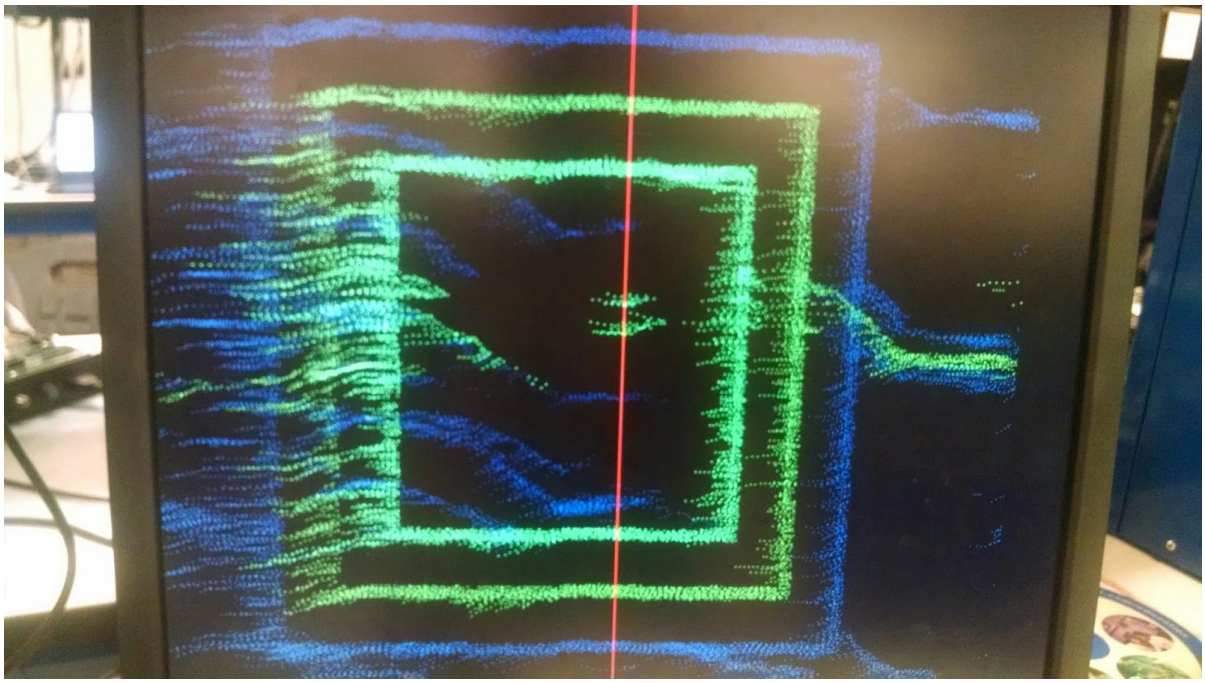

Figure 12: Image rotated by zero degrees with timing problems.

As you can see, the timing of the values given to the rotation module were very important; the image appears to be melting when it has incorrect timing. Unfortunately, this was not able to be resolved; changing the forecasting of both the vram_display coordinates and the coordinates given to the rotation module did have an effect on the image, reducing the melting; however the melting never disappeared completely, and the image would only melt in the other direction (to the right as opposed to the left) when I overcorrected the forecasted hcount and vcount. If I had more time to work on the project, one thing I would like to do would be to discover the source of this issue. I took into account the 11 clock cycles it would take to rotate the x and y coordinate in the rotation module, as well as the time it took to read from the ZBT ram. Obviously there was some other timing problem or other issue in there that I was unable to discover, and would need to be accounted for if implementing my original idea for the visualization.

5.3 image_init

After deciding that I would be unable to perform my original rotation idea to the extent that any image I rotated would be distorted due to the aforementioned issues, I decided instead to have expanding nested circles as my visualization image. This was implemented by initializing several moving_circle modules in the center and in the corners of the screen; the next image would be loaded to the rendering ZBT by passing hcount and vcount directly through these modules and writing the subsequent pixel value to the ZBT, while the contents of the current ZBT was displayed.

5.3.1 moving_circles

The moving_circles module takes in an hcount and vcount and checks if the values was within a certain ring in the display, and if so colors that pixel in. The radius for the ring of values expands at every vsync, and is tied to the tempo produced by the audio part of this project, so that a faster tempo means the radius of the circle gets larger at every vsync than it does for a slower tempo.

In addition, there is a delay for the circle expansion that can be set, so that I could initialize several of these circles to the same location-- the middle of the screen-- and have them start expanding at different times in order to create the nested circles effect. This was done because I wanted the circles to expand until they reached the edge of the screen, and then reappear in the same place in the center of the screen when the value for the radius overflowed. This image was successfully implemented, and the speed of the circle expansion was easily tied to the tempo value given to me by the audio block, and proportional so that a larger bpm meant a faster expansion. Figures 13-15 display the final image with the moving circles, as well as the “rotation mode.”

Figure 13: Final image

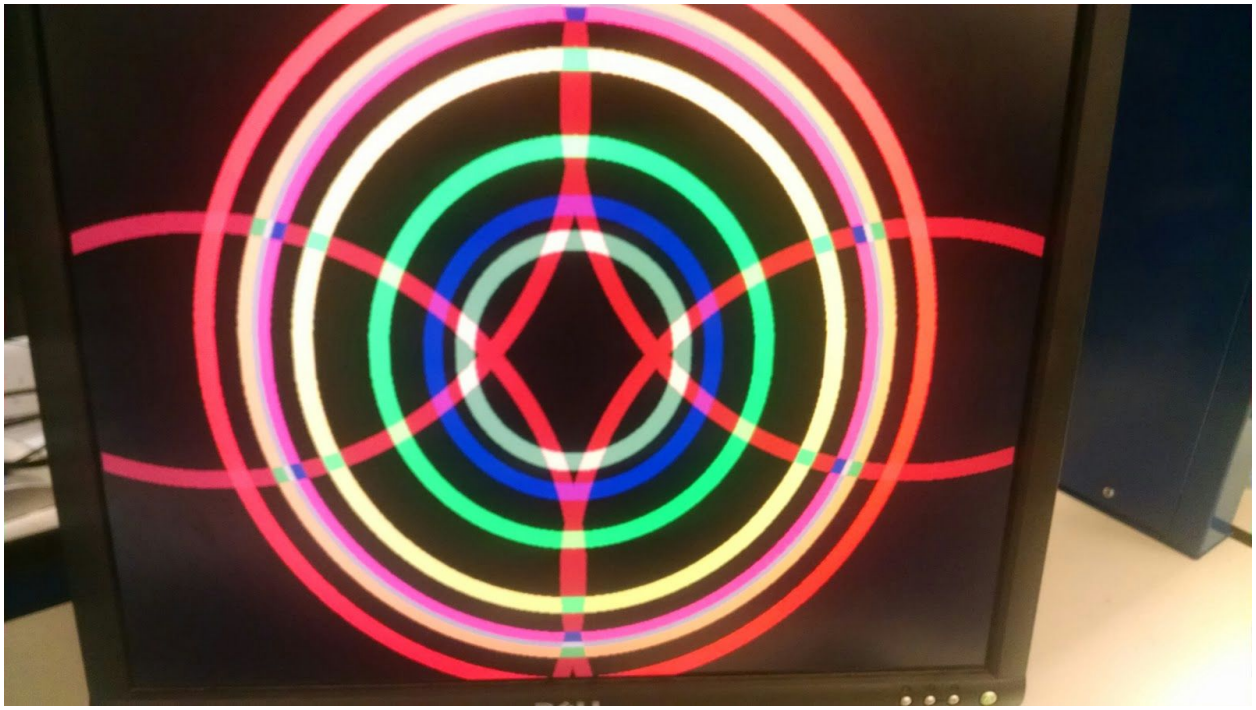

Figure 14: Final image, expanded slightly

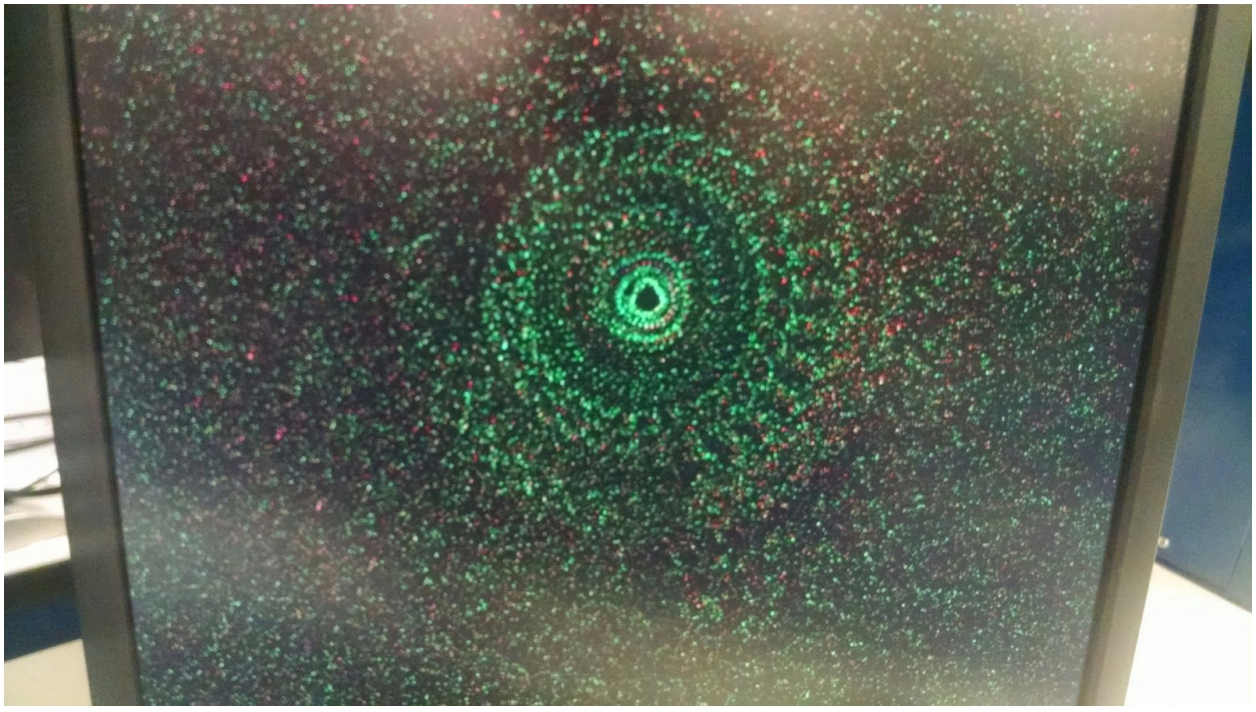

Figure 15: “Rotation mode”

5.4 color_shft

The color shift is implemented on the pixel values coming from the current ZBT rather than the pixel values being saved in the rendering ZBT. The main input values for this module are the

pixel value and vsync, and the output is the shifted pixel value. In order to achieve a shifting effect that would have multiple shifts-- so that the color would not alternate between two but rather have multiple modes-- I implemented a counter that increments on a pulse signifying a beat given by the top-level module and multiplies each 8-bit reg, green, and blue pixel value by some value relating to this increment, such as the counter squared times three. In doing so, the red, green, and blue pixel values are changed by different amounts, leading to a more interesting color shift.

5.5 vram_display

I used the provided vram_display module in order to display images from the ZBT, with some modifications. The original module assumed four pixels were being stored in each address; in my implementation I only stored one, so needed to modify the code slightly in order to output just the one pixel every clock cycle. In addition, I forecasted the hcount and vcount values slightly differently due to the differences in my use and the original intended use of the module in relation to timing of other modules.

5.6 Testing and Debugging Methods

While I ran what simulations i was able to on my modules, such as with the rotation module, often I found that testing using the display itself was more informative about the issues I was having; and, since I could not use an actual ZBT in simulation, it made more sense to debug the switching between ZBT rams first and then use the images I was generating for testing. I created the translation module originally as a test module, and it was very helpful for me to use for several issues I was having. For almost all of the visualization debugging, I used the display to try and figure out the problem as opposed to a simulation in ISE.

6. Final Product

The final project we produced was able to listen to a song and detect significant beats in the song, in real time. Using this beat, the image was able to change color in time with the music, creating an interesting and colorful image that could match up with the music. While the tempo was unable to be determined in the audio aspect of the project, the image can change speeds from a series of switches on the FPGA. In addition, although the rotation was not able to be implemented correctly, one can still go into “rotation mode” by the first switch on the labkit. Though the rotation does not work as originally intended, it still does rotate a very distorted image, leading to a color-changing whirlpool-like image that is actually very interesting to look at.

Appendix A: Verilog source code

```
`default_nettype none

//////////////////////////////////////////////////////////////////////////////
//
// Switch Debounce Module
//
//////////////////////////////////////////////////////////////////////////////

module debounce (
  input wire reset, clock, noisy,
  output reg clean
);
  reg [18:0] count;
  reg new;

  always @(posedge clock)
    if (reset) begin
      count <= 0;
      new <= noisy;
      clean <= noisy;
    end
    else if (noisy != new) begin
      // noisy input changed, restart the .01 sec clock
      new <= noisy;
      count <= 0;
    end
    else if (count == 270000)
      // noisy input stable for .01 secs, pass it along!
      clean <= new;
    else
      // waiting for .01 sec to pass
      count <= count+1;

endmodule

//////////////////////////////////////////////////////////////////////////////
//
// bi-directional monaural interface to AC97
//
//////////////////////////////////////////////////////////////////////////////

module lab5audio (
  input wire clock_27mhz,
  input wire reset,
  input wire [4:0] volume,
  output wire [7:0] audio_in_data,
  input wire [7:0] audio_out_data,
  output wire ready,
```

```
  output reg audio_reset_b,   // ac97 interface signals
  output wire ac97_sdata_out,
  input wire ac97_sdata_in,
  output wire ac97_synch,
  input wire ac97_bit_clock
);

  wire [7:0] command_address;
  wire [15:0] command_data;
  wire command_valid;
  wire [19:0] left_in_data, right_in_data;
  wire [19:0] left_out_data, right_out_data;

  // wait a little before enabling the AC97 codec
  reg [9:0] reset_count;
  always @(posedge clock_27mhz) begin
    if (reset) begin
      audio_reset_b = 1'b0;
      reset_count = 0;
    end else if (reset_count == 1023)
      audio_reset_b = 1'b1;
    else
      reset_count = reset_count+1;
  end

  wire ac97_ready;
  ac97 ac97(.ready(ac97_ready),
            .command_address(command_address),
            .command_data(command_data),
            .command_valid(command_valid),
            .left_data(left_out_data), .left_valid(1'b1),
            .right_data(right_out_data), .right_valid(1'b1),
            .left_in_data(left_in_data), .right_in_data(right_in_data),
            .ac97_sdata_out(ac97_sdata_out),
            .ac97_sdata_in(ac97_sdata_in),
            .ac97_synch(ac97_synch),
            .ac97_bit_clock(ac97_bit_clock));

  // ready: one cycle pulse synchronous with clock_27mhz
  reg [2:0] ready_sync;
  always @ (posedge clock_27mhz) ready_sync <= {ready_sync[1:0], ac97_ready};
  assign ready = ready_sync[1] & ~ready_sync[2];

  reg [7:0] out_data;
  always @ (posedge clock_27mhz)
    if (ready) out_data <= audio_out_data;
  assign audio_in_data = left_in_data[19:12];
  assign left_out_data = {out_data, 12'b000000000000};
```

```
  assign right_out_data = left_out_data;

  // generate repeating sequence of read/writes to AC97 registers
  ac97commands cmds(.clock(clock_27mhz), .ready(ready),
                    .command_address(command_address),
                    .command_data(command_data),
                    .command_valid(command_valid),
                    .volume(volume),
                    .source(3'b000));     // mic
endmodule

// assemble/disassemble AC97 serial frames
module ac97 (
  output reg ready,
  input wire [7:0] command_address,
  input wire [15:0] command_data,
  input wire command_valid,
  input wire [19:0] left_data,
  input wire left_valid,
  input wire [19:0] right_data,
  input wire right_valid,
  output reg [19:0] left_in_data, right_in_data,
  output reg ac97_sdata_out,
  input wire ac97_sdata_in,
  output reg ac97_synch,
  input wire ac97_bit_clock
);
  reg [7:0] bit_count;

  reg [19:0] l_cmd_addr;
  reg [19:0] l_cmd_data;
  reg [19:0] l_left_data, l_right_data;
  reg l_cmd_v, l_left_v, l_right_v;

  initial begin
    ready <= 1'b0;
    // synthesis attribute init of ready is "0";
    ac97_sdata_out <= 1'b0;
    // synthesis attribute init of ac97_sdata_out is "0";
    ac97_synch <= 1'b0;
    // synthesis attribute init of ac97_synch is "0";

    bit_count <= 8'h00;
    // synthesis attribute init of bit_count is "0000";
    l_cmd_v <= 1'b0;
    // synthesis attribute init of l_cmd_v is "0";
    l_left_v <= 1'b0;
    // synthesis attribute init of l_left_v is "0";
```

```
  l_right_v <= 1'b0;
  // synthesis attribute init of l_right_v is "0";

  left_in_data <= 20'h00000;
  // synthesis attribute init of left_in_data is "00000";
  right_in_data <= 20'h00000;
  // synthesis attribute init of right_in_data is "00000";
end

always @(posedge ac97_bit_clock) begin
  // Generate the sync signal
  if (bit_count == 255)
    ac97_synch <= 1'b1;
  if (bit_count == 15)
    ac97_synch <= 1'b0;

  // Generate the ready signal
  if (bit_count == 128)
    ready <= 1'b1;
  if (bit_count == 2)
    ready <= 1'b0;

  // Latch user data at the end of each frame. This ensures that the
  // first frame after reset will be empty.
  if (bit_count == 255) begin
    l_cmd_addr <= {command_address, 12'h000};
    l_cmd_data <= {command_data, 4'h0};
    l_cmd_v <= command_valid;
    l_left_data <= left_data;
    l_left_v <= left_valid;
    l_right_data <= right_data;
    l_right_v <= right_valid;
  end

  if ((bit_count >= 0) && (bit_count <= 15))
    // Slot 0: Tags
    case (bit_count[3:0])
      4'h0: ac97_sdata_out <= 1'b1;      // Frame valid
      4'h1: ac97_sdata_out <= l_cmd_v;   // Command address valid
      4'h2: ac97_sdata_out <= l_cmd_v;   // Command data valid
      4'h3: ac97_sdata_out <= l_left_v;  // Left data valid
      4'h4: ac97_sdata_out <= l_right_v; // Right data valid
      default: ac97_sdata_out <= 1'b0;
    endcase
  else if ((bit_count >= 16) && (bit_count <= 35))
    // Slot 1: Command address (8-bits, left justified)
    ac97_sdata_out <= l_cmd_v ? l_cmd_addr[35-bit_count] : 1'b0;
  else if ((bit_count >= 36) && (bit_count <= 55))
```

```
      // Slot 2: Command data (16-bits, left justified)
      ac97_sdata_out <= l_cmd_v ? l_cmd_data[55-bit_count] : 1'b0;
    else if ((bit_count >= 56) && (bit_count <= 75)) begin
      // Slot 3: Left channel
      ac97_sdata_out <= l_left_v ? l_left_data[19] : 1'b0;
      l_left_data <= { l_left_data[18:0], l_left_data[19] };
    end
    else if ((bit_count >= 76) && (bit_count <= 95))
      // Slot 4: Right channel
      ac97_sdata_out <= l_right_v ? l_right_data[95-bit_count] : 1'b0;
    else
      ac97_sdata_out <= 1'b0;

    bit_count <= bit_count+1;
  end // always @ (posedge ac97_bit_clock)

  always @(negedge ac97_bit_clock) begin
    if ((bit_count >= 57) && (bit_count <= 76))
      // Slot 3: Left channel
      left_in_data <= { left_in_data[18:0], ac97_sdata_in };
    else if ((bit_count >= 77) && (bit_count <= 96))
      // Slot 4: Right channel
      right_in_data <= { right_in_data[18:0], ac97_sdata_in };
  end
endmodule

// issue initialization commands to AC97
module ac97commands (
  input wire clock,
  input wire ready,
  output wire [7:0] command_address,
  output wire [15:0] command_data,
  output reg command_valid,
  input wire [4:0] volume,
  input wire [2:0] source
);
  reg [23:0] command;

  reg [3:0] state;
  initial begin
    command <= 4'h0;
    // synthesis attribute init of command is "0";
    command_valid <= 1'b0;
    // synthesis attribute init of command_valid is "0";
    state <= 16'h0000;
    // synthesis attribute init of state is "0000";
  end
```

```
  assign command_address = command[23:16];
  assign command_data = command[15:0];

  wire [4:0] vol;
  assign vol = 31-volume;  // convert to attenuation

  always @(posedge clock) begin
    if (ready) state <= state+1;

    case (state)
      4'h0: // Read ID
        begin
          command <= 24'h80_0000;
          command_valid <= 1'b1;
        end
      4'h1: // Read ID
        command <= 24'h80_0000;
      4'h3: // headphone volume
        command <= { 8'h04, 3'b000, vol, 3'b000, vol };
      4'h5: // PCM volume
        command <= 24'h18_0808;
      4'h6: // Record source select
        command <= { 8'h1A, 5'b00000, source, 5'b00000, source};
      4'h7: // Record gain = max
        command <= 24'h1C_0F0F;
      4'h9: // set +20db mic gain
        command <= 24'h0E_8048;
      4'hA: // Set beep volume
        command <= 24'h0A_0000;
      4'hB: // PCM out bypass mix1
        command <= 24'h20_8000;
      default:
        command <= 24'h80_0000;
    endcase // case(state)
  end // always @ (posedge clock)
endmodule // ac97commands



//////////////////////////////////////////////////////////////////////////////////
////
//// 6.111 FPGA Labkit -- Template Toplevel Module
////
//// For Labkit Revision 004
//// Created: October 31, 2004, from revision 003 file
//// Author: Nathan Ickes, 6.111 staff
////
//////////////////////////////////////////////////////////////////////////////////
```

```
module final_project  (beep, audio_reset_b, ac97_sdata_out, ac97_sdata_in, ac97_synch,
                       ac97_bit_clock,

                       vga_out_red, vga_out_green, vga_out_blue, vga_out_sync_b,
                       vga_out_blank_b, vga_out_pixel_clock, vga_out_hsync,
                       vga_out_vsync,

                       tv_out_ycrcb, tv_out_reset_b, tv_out_clock, tv_out_i2c_clock,
                       tv_out_i2c_data, tv_out_pal_ntsc, tv_out_hsync_b,
                       tv_out_vsync_b, tv_out_blank_b, tv_out_subcar_reset,

                       tv_in_ycrcb, tv_in_data_valid, tv_in_line_clock1,
                       tv_in_line_clock2, tv_in_aef, tv_in_hff, tv_in_aff,
                       tv_in_i2c_clock, tv_in_i2c_data, tv_in_fifo_read,
                       tv_in_fifo_clock, tv_in_iso, tv_in_reset_b, tv_in_clock,

                       ram0_data, ram0_address, ram0_adv_ld, ram0_clk, ram0_cen_b,
                       ram0_ce_b, ram0_oe_b, ram0_we_b, ram0_bwe_b,

                       ram1_data, ram1_address, ram1_adv_ld, ram1_clk, ram1_cen_b,
                       ram1_ce_b, ram1_oe_b, ram1_we_b, ram1_bwe_b,

                       clock_feedback_out, clock_feedback_in,

                       flash_data, flash_address, flash_ce_b, flash_oe_b, flash_we_b,
                       flash_reset_b, flash_sts, flash_byte_b,

                       rs232_txd, rs232_rxd, rs232_rts, rs232_cts,

                       mouse_clock, mouse_data, keyboard_clock, keyboard_data,

                       clock_27mhz, clock1, clock2,

                       disp_blank, disp_data_out, disp_clock, disp_rs, disp_ce_b,
                       disp_reset_b, disp_data_in,

                       button0, button1, button2, button3, button_enter, button_right,
                       button_left, button_down, button_up,

                       switch,

                       led,

                       user1, user2, user3, user4,

                       daughtercard,
```

```
            systemace_data, systemace_address, systemace_ce_b,
            systemace_we_b, systemace_oe_b, systemace_irq, systemace_mpbrdy,

            analyzer1_data, analyzer1_clock,
            analyzer2_data, analyzer2_clock,
            analyzer3_data, analyzer3_clock,
            analyzer4_data, analyzer4_clock);

output beep, audio_reset_b, ac97_synch, ac97_sdata_out;
input  ac97_bit_clock, ac97_sdata_in;

output [7:0] vga_out_red, vga_out_green, vga_out_blue;
output vga_out_sync_b, vga_out_blank_b, vga_out_pixel_clock,
        vga_out_hsync, vga_out_vsync;

output [9:0] tv_out_ycrcb;
output tv_out_reset_b, tv_out_clock, tv_out_i2c_clock, tv_out_i2c_data,
        tv_out_pal_ntsc, tv_out_hsync_b, tv_out_vsync_b, tv_out_blank_b,
        tv_out_subcar_reset;

input  [19:0] tv_in_ycrcb;
input  tv_in_data_valid, tv_in_line_clock1, tv_in_line_clock2, tv_in_aef,
        tv_in_hff, tv_in_aff;
output tv_in_i2c_clock, tv_in_fifo_read, tv_in_fifo_clock, tv_in_iso,
        tv_in_reset_b, tv_in_clock;
inout  tv_in_i2c_data;

inout  [35:0] ram0_data;
output [18:0] ram0_address;
output ram0_adv_ld, ram0_clk, ram0_cen_b, ram0_ce_b, ram0_oe_b, ram0_we_b;
output [3:0] ram0_bwe_b;

inout  [35:0] ram1_data;
output [18:0] ram1_address;
output ram1_adv_ld, ram1_clk, ram1_cen_b, ram1_ce_b, ram1_oe_b, ram1_we_b;
output [3:0] ram1_bwe_b;

input  clock_feedback_in;
output clock_feedback_out;

inout  [15:0] flash_data;
output [23:0] flash_address;
output flash_ce_b, flash_oe_b, flash_we_b, flash_reset_b, flash_byte_b;
input  flash_sts;

output rs232_txd, rs232_rts;
input  rs232_rxd, rs232_cts;
```

```
  input  mouse_clock, mouse_data, keyboard_clock, keyboard_data;

  input  clock_27mhz, clock1, clock2;

  output disp_blank, disp_clock, disp_rs, disp_ce_b, disp_reset_b;
  input  disp_data_in;
  output  disp_data_out;

  input  button0, button1, button2, button3, button_enter, button_right,
         button_left, button_down, button_up;
  input  [7:0] switch;
  output [7:0] led;

  inout [31:0] user1, user2, user3, user4;

  inout [43:0] daughtercard;

  inout  [15:0] systemace_data;
  output [6:0]  systemace_address;
  output systemace_ce_b, systemace_we_b, systemace_oe_b;
  input  systemace_irq, systemace_mpbrdy;

  output [15:0] analyzer1_data, analyzer2_data, analyzer3_data,
                analyzer4_data;
  output analyzer1_clock, analyzer2_clock, analyzer3_clock, analyzer4_clock;

  ////////////////////////////////////////////////////////////////////////
  //
  // I/O Assignments
  //
  ////////////////////////////////////////////////////////////////////////


  // Audio Input and Output
  assign beep= 1'b0;
  //lab5 assign audio_reset_b = 1'b0;
  //lab5 assign ac97_synch = 1'b0;
  //lab5 assign ac97_sdata_out = 1'b0;
  // ac97_sdata_in is an input

  // VGA Output
//   assign vga_out_red = 10'h0;
//   assign vga_out_green = 10'h0;
//   assign vga_out_blue = 10'h0;
//   assign vga_out_sync_b = 1'b1;
//   assign vga_out_blank_b = 1'b1;
//   assign vga_out_pixel_clock = 1'b0;
//   assign vga_out_hsync = 1'b0;
```

```
//   assign vga_out_vsync = 1'b0;

   // Video Output
   assign tv_out_ycrcb = 10'h0;
   assign tv_out_reset_b = 1'b0;
   assign tv_out_clock = 1'b0;
   assign tv_out_i2c_clock = 1'b0;
   assign tv_out_i2c_data = 1'b0;
   assign tv_out_pal_ntsc = 1'b0;
   assign tv_out_hsync_b = 1'b1;
   assign tv_out_vsync_b = 1'b1;
   assign tv_out_blank_b = 1'b1;
   assign tv_out_subcar_reset = 1'b0;

   // Video Input
   assign tv_in_i2c_clock = 1'b0;
   assign tv_in_fifo_read = 1'b0;
   assign tv_in_fifo_clock = 1'b0;
   assign tv_in_iso = 1'b0;
   assign tv_in_reset_b = 1'b0;
   assign tv_in_clock = 1'b0;
   assign tv_in_i2c_data = 1'bZ;
   // tv_in_ycrcb, tv_in_data_valid, tv_in_line_clock1, tv_in_line_clock2,
   // tv_in_aef, tv_in_hff, and tv_in_aff are inputs

   // SRAMs
         /* enable RAM pins */

   assign ram0_ce_b = 1'b0;
   assign ram0_oe_b = 1'b0;
   assign ram0_adv_ld = 1'b0;
   assign ram0_bwe_b = 4'h0;
/*
   assign ram1_data = 36'hZ;
   assign ram1_address = 19'h0;*/
   assign ram1_adv_ld = 1'b0;
   //assign ram1_clk = 1'b0;

   //These values has to be set to 0 like ram0 if ram1 is used.
   //assign ram1_cen_b = 1'b1;
   assign ram1_ce_b = 1'b0;
   assign ram1_oe_b = 1'b0;
   //assign ram1_we_b = 1'b1;
   assign ram1_bwe_b = 4'h0;

//   assign ram0_data = 36'hZ;
//   assign ram0_address = 19'h0;
//   assign ram0_adv_ld = 1'b0;
```

```
//   assign ram0_clk = 1'b0;
//   assign ram0_cen_b = 1'b1;
//   assign ram0_ce_b = 1'b1;
//   assign ram0_oe_b = 1'b1;
//   assign ram0_we_b = 1'b1;
//   assign ram0_bwe_b = 4'hF;
//   assign ram1_data = 36'hZ;
//   assign ram1_address = 19'h0;
//   assign ram1_adv_ld = 1'b0;
//   assign ram1_clk = 1'b0;
//   assign ram1_cen_b = 1'b1;
//   assign ram1_ce_b = 1'b1;
//   assign ram1_oe_b = 1'b1;
//   assign ram1_we_b = 1'b1;
//   assign ram1_bwe_b = 4'hF;
//   assign clock_feedback_out = 1'b0;
   // clock_feedback_in is an input

   // Flash ROM
   assign flash_data = 16'hZ;
   assign flash_address = 24'h0;
   assign flash_ce_b = 1'b1;
   assign flash_oe_b = 1'b1;
   assign flash_we_b = 1'b1;
   assign flash_reset_b = 1'b0;
   assign flash_byte_b = 1'b1;
   // flash_sts is an input

   // RS-232 Interface
   assign rs232_txd = 1'b1;
   assign rs232_rts = 1'b1;
   // rs232_rxd and rs232_cts are inputs

   // PS/2 Ports
   // mouse_clock, mouse_data, keyboard_clock, and keyboard_data are inputs

   // LED Displays
   assign disp_blank = 1'b1;
   assign disp_clock = 1'b0;
   assign disp_rs = 1'b0;
   assign disp_ce_b = 1'b1;
   assign disp_reset_b = 1'b0;
   assign disp_data_out = 1'b0;
   // disp_data_in is an input

   // Buttons, Switches, and Individual LEDs
   assign led = 8'hFF;
   // button0, button1, button2, button3, button_enter, button_right,
```

```
  // button_left, button_down, button_up, and switches are inputs

  // User I/Os
  //assign user1 = 32'hZ;
  assign user2 = 32'hZ;
  assign user3 = 32'hZ;
  assign user4 = 32'hZ;

  // Daughtercard Connectors
  assign daughtercard = 44'hZ;

  // SystemACE Microprocessor Port
  assign systemace_data = 16'hZ;
  assign systemace_address = 7'h0;
  assign systemace_ce_b = 1'b1;
  assign systemace_we_b = 1'b1;
  assign systemace_oe_b = 1'b1;
  // systemace_irq and systemace_mpbrdy are inputs

  // Logic Analyzer
  //lab5 assign analyzer1_data = 16'h0;
  //lab5 assign analyzer1_clock = 1'b1;
  //assign analyzer2_data = 16'h0;
  //assign analyzer2_clock = 1'b1;
  //lab5 assign analyzer3_data = 16'h0;
  //lab5 assign analyzer3_clock = 1'b1;
//   assign analyzer4_data = 16'h0;
            assign analyzer4_clock = 1'b1;

//   wire [7:0] from_ac97_data, to_ac97_data;
//   wire ready;

  //////////////////////////////////////////////////////////////////////////
  //
  // Reset Generation
  //
  // A shift register primitive is used to generate an active-high reset
  // signal that remains high for 16 clock cycles after configuration finishes
  // and the FPGA's internal clocks begin toggling.
  //
  //////////////////////////////////////////////////////////////////////////
  wire reset;
  SRL16 #(.INIT(16'hFFFF)) reset_sr(.D(1'b0), .CLK(clock_27mhz), .Q(reset),
                              .A0(1'b1), .A1(1'b1), .A2(1'b1), .A3(1'b1));

  wire hard_reset;
         debounce
db_reset(.reset(reset),.clock(clock_27mhz),.noisy(button_enter),.clean(hard_reset));
```

```
        wire [7:0] from_ac97_data, to_ac97_data;
  wire ready;
        wire [7:0]volume = 0;

  // AC97 driver
  lab5audio a(clock_27mhz, reset, volume, from_ac97_data, to_ac97_data, ready,
              audio_reset_b, ac97_sdata_out, ac97_sdata_in,
              ac97_synch, ac97_bit_clock);

        // ZBT clock
        wire clk;

        wire clock_40mhz_unbuf,clock_40mhz;
  DCM vclk1(.CLKIN(clock_27mhz),.CLKFX(clock_40mhz_unbuf));
  // synthesis attribute CLKFX_DIVIDE of vclk1 is 2
  // synthesis attribute CLKFX_MULTIPLY of vclk1 is 3
  // synthesis attribute CLK_FEEDBACK of vclk1 is NONE
  // synthesis attribute CLKIN_PERIOD of vclk1 is 37
  BUFG vclk2(.O(clock_40mhz),.I(clock_40mhz_unbuf));

  wire locked;
        //assign clock_feedback_out = 0; // gph 2011-Nov-10

  ramclock1 rc(.ref_clock(clock_40mhz), .fpga_clock(clk),
                                        .ram0_clock(ram0_clk),
                                        .ram1_clock(ram1_clk),   //uncomment if ram1 is used
                                        .clock_feedback_in(clock_feedback_in),
                                        .clock_feedback_out(clock_feedback_out),
.locked(locked));
        // Enter button reset
        wire reset1,user_reset1;
// power-on reset generation
  wire power_on_reset1;    // remain high for first 16 clocks
  SRL16 reset_sr1 (.D(1'b0), .CLK(clk), .Q(power_on_reset1),
                   .A0(1'b1), .A1(1'b1), .A2(1'b1), .A3(1'b1));
  defparam reset_sr.INIT = 16'hFFFF;

  // ENTER button is user reset

  debounce db1(power_on_reset1, clk, ~button_enter, user_reset1);
  assign reset1 = user_reset1 | power_on_reset1;

        ////my code/////

        //declare connections
        wire signed [17:0]lowpassed;
        wire signed [7:0]filtband0;
```

```
wire signed [7:0]filtband1;
wire signed [7:0]filtband2;
wire signed [7:0]filtband3;
wire signed [7:0]filtband4;
wire signed [7:0]filtband5;
wire signed [7:0]comb00;
wire signed [7:0]comb01;
wire signed [7:0]comb02;
wire signed [7:0]comb03;
wire signed [7:0]comb04;
wire signed [7:0]comb05;
wire signed [7:0]comb10;
wire signed [7:0]comb11;
wire signed [7:0]comb12;
wire signed [7:0]comb13;
wire signed [7:0]comb14;
wire signed [7:0]comb15;
wire signed [7:0]comb20;
wire signed [7:0]comb21;
wire signed [7:0]comb22;
wire signed [7:0]comb23;
wire signed [7:0]comb24;
wire signed [7:0]comb25;
wire signed [7:0]comb30;
wire signed [7:0]comb31;
wire signed [7:0]comb32;
wire signed [7:0]comb33;
wire signed [7:0]comb34;
wire signed [7:0]comb35;
wire signed [7:0]comb40;
wire signed [7:0]comb41;
wire signed [7:0]comb42;
wire signed [7:0]comb43;
wire signed [7:0]comb44;
wire signed [7:0]comb45;
wire [7:0]tempo;
wire beat;

wire sixk_ready;
wire signed [7:0]fw_rect_band0;
wire signed [7:0]hann_out0;
wire signed [7:0]diff_out0;
wire signed [7:0]hw_rect_band0;
wire signed [7:0]fw_rect_band1;
wire signed [7:0]hann_out1;
wire signed [7:0]diff_out1;
wire signed [7:0]hw_rect_band1;
wire signed [7:0]fw_rect_band2;
```

```
        wire signed [7:0]hann_out2;
        wire signed [7:0]diff_out2;
        wire signed [7:0]hw_rect_band2;
        wire signed [7:0]fw_rect_band3;
        wire signed [7:0]hann_out3;
        wire signed [7:0]diff_out3;
        wire signed [7:0]hw_rect_band3;
        wire signed [7:0]fw_rect_band;
        wire signed [7:0]hann_out;
        wire signed [7:0]diff_out;
        wire signed [7:0]hw_rect_band;
        wire [15:0]eng60;
        wire [15:0]eng90;
        wire [15:0]eng120;
        wire [15:0]eng180;
        wire [15:0]eng210;
        wire [15:0]eng240;

        // output useful things to the logic analyzer connectors
  assign analyzer3_clock = clock_27mhz;
  assign analyzer3_data = {filtband0,hann_out};
        assign analyzer1_clock = ready;
        assign analyzer1_data = {diff_out,7'd0,beat};
        //assign analyzer4_clock = sixk_ready;
        assign analyzer4_data = {eng60};
        assign analyzer2_clock = sixk_ready;
  assign analyzer2_data = {eng120};

        //lowpass audio by half of sampling frequency as it comes in
        lowpass3k
LP(.clock(clock_27mhz),.reset(hard_reset),.ready(ready),.x(from_ac97_data),.y(lowpassed));

        //sample audio input and split into frequency bands
        Filterbank
fbank1(.clk(clock_27mhz),.reset(hard_reset),.ready(ready),.sixk_ready(sixk_ready),.x(lowpassed[17:10]),
  .band0(filtband0),.band1(filtband1),.band2(filtband2),.band3(filtband3),.band4(filtband4));

        //process audio in freq bands
        AudioProcessingUnit
APU0(.clk(clock_27mhz),.reset(hard_reset),.ready(sixk_ready),.bandx(filtband0),

.comb60(comb00),.comb90(comb01),.comb120(comb02),.comb180(comb03),.comb210(comb04),.comb240(comb05),

.hann_clip(hann_out0),.diff_out(diff_out0),.fw_rect_band(fw_rect_band0),.hw_rect_band(hw_rect_band0));
```

```
        AudioProcessingUnit
APU1(.clk(clock_27mhz),.reset(hard_reset),.ready(sixk_ready),.bandx(filtband1),

.comb60(comb10),.comb90(comb11),.comb120(comb12),.comb180(comb13),.comb210(comb14),.c
omb240(comb15),

.hann_clip(hann_out1),.diff_out(diff_out1),.fw_rect_band(fw_rect_band1),.hw_rect_band(hw_rect
_band1));
        AudioProcessingUnit
APU2(.clk(clock_27mhz),.reset(hard_reset),.ready(sixk_ready),.bandx(filtband2),

.comb60(comb20),.comb90(comb21),.comb120(comb22),.comb180(comb23),.comb210(comb24
),.comb240(comb25),

.hann_clip(hann_out2),.diff_out(diff_out2),.fw_rect_band(fw_rect_band2),.hw_rect_band(hw_re
ct_band2));
        AudioProcessingUnit
APU3(.clk(clock_27mhz),.reset(hard_reset),.ready(sixk_ready),.bandx(filtband3),

.comb60(comb30),.comb90(comb31),.comb120(comb32),.comb180(comb33),.comb210(comb34)
,.comb240(comb35),

.hann_clip(hann_out3),.diff_out(diff_out3),.fw_rect_band(fw_rect_band3),.hw_rect_band(hw_re
ct_band3));
        AudioProcessingUnit
APU4(.clk(clock_27mhz),.reset(hard_reset),.ready(sixk_ready),.bandx(filtband4),

.comb60(comb40),.comb90(comb41),.comb120(comb42),.comb180(comb43),.comb210(comb44)
,.comb240(comb45),

.hann_clip(hann_out),.diff_out(diff_out),.fw_rect_band(fw_rect_band),.hw_rect_band(hw_rect_b
and));

//      //find which tempo has highest energy
        Peakfinder peakfinder1(.clk(clock_27mhz),.ready(sixk_ready),.reset(hard_reset),

.comb00(comb00),.comb01(comb01),.comb02(comb02),.comb03(comb03),.comb04(comb04),.c
omb05(comb05),

.comb10(comb10),.comb11(comb11),.comb12(comb12),.comb13(comb13),.comb14(comb14),.comb15
(comb15),

.comb20(comb20),.comb21(comb21),.comb22(comb22),.comb23(comb23),.comb24(comb24),.c
omb25(comb25),

.comb30(comb30),.comb31(comb31),.comb32(comb32),.comb33(comb33),.comb34(comb34),.co
mb35(comb35),
```

```
.comb40(comb40),.comb41(comb41),.comb42(comb42),.comb43(comb43),.comb44(comb44),.co
mb45(comb45),
        .energy60(eng60),.energy90(eng90),.energy120(eng120),.energy180(eng180),
        .energy210(eng210),.energy240(eng240),.tempo(tempo),.beat(beat));

          // generate basic SVGA video signals
  wire [10:0] hcount;
  wire [9:0]  vcount;
  wire hsync,vsync,blank;
  svga1 svga1(clk,hcount,vcount,hsync,vsync,blank);

  // wire up to ZBT ram
        wire [35:0] vram_write_data, vram_write_data_init, vram_write_data1;
  wire [35:0] vram0_read_data, vram1_read_data, vram_read_data;
  wire [18:0] vram_addr, vram0_addr, vram1_addr, vram_addr0, vram_addr3, vram_addr2;
        wire        vram_we, vram0_we, vram1_we;
  reg we_render;
  reg currentram;

        reg init;
        reg [2:0] addr_count;
  wire ram0_clk_not_used;

        wire [31:0] angle;
        wire [11:0] x_rot;
        wire [10:0] y_rot;
        wire [11:0] x_trans;
        wire [10:0] y_trans;
        wire [11:0] x_in;
        wire [10:0] y_in;

        // generate pixel value from reading ZBT memory
  wire [23:0]    vr_pixel;
        wire [23:0]       circle_pixel;

  wire [18:0]     vram_addr1;
        wire [23:0] pixel_out;

        reg circle = 1;
        wire [7:0] temp = switch[7:0];

        wire [18:0] vram_addr_init = {hcount[10:0] + vcount[9:0]*800};

        assign vram_addr0 = currentram ? vram_addr2 : vram_addr3;

  wire [18:0] write_addr = circle ? vram_addr_init : vram_addr0;
```

```
        assign vram0_addr = ~init ? vram_addr_init : (currentram ? write_addr: vram_addr1);
        assign vram0_we = currentram ? 1 : we_render;

        assign vram1_addr = ~currentram ? write_addr : vram_addr1;
        assign vram1_we = currentram ? we_render : 1;

        assign vram_read_data = currentram ? vram1_read_data : vram0_read_data;

        assign vram_write_data1 = vram_read_data;

   zbt_6111 zbt0(clk, 1'b1, vram0_we, vram0_addr,
                 vram_write_data, vram0_read_data,
                 ram0_clk_not_used,   //to get good timing, don't connect ram_clk to zbt_6111
                 ram0_we_b, ram0_address, ram0_data, ram0_cen_b);

        wire ram1_clk_not_used;

   zbt_6111 zbt1(clk, 1'b1, vram1_we, vram1_addr,
                 vram_write_data, vram1_read_data,
                 ram1_clk_not_used,   //to get good timing, don't connect ram_clk to zbt_6111
                 ram1_we_b, ram1_address, ram1_data, ram1_cen_b);

        graphics_rotation g1(.clk(clk),.reset(reset1), .circle(circle),.addr_count(addr_count),
.hcount(hcount), .vcount(vcount),
                        .tempo(temp), .vram_addr2(vram_addr2), .vram_addr3(vram_addr3));

        image_init im0(.clk(clk), .tempo(temp), .circle(circle),.hcount(hcount), .vcount(vcount),
.vsync(vsync),.pixel(circle_pixel));

        assign vram_write_data = (circle ? circle_pixel : vram_write_data1);

        vram_display1 vd1(reset1,clk, ~init, hcount,vcount,vr_pixel,
                  vram_addr1,vram_read_data);

        always@(posedge clk) begin
                we_render <= ~init ? 1 : 0;

        end

        reg shift = 0;

        color_shft color(.reset(reset1), .shft(shift), .vr(vr_pixel), .pixel(pixel_out));


        reg beat_state;
        parameter RESET = 0;
        parameter BEAT = 1;
        reg old_vsync;
```

```
	reg vs_pulse;

	always@(posedge clk) begin
		old_vsync <= vsync;
		vs_pulse <= !old_vsync && vsync;
		case(beat_state)
			RESET: if (beat) beat_state <= BEAT;
			BEAT: if (vs_pulse) beat_state <= RESET;
		endcase
	end



	reg [9:0] count= 0;
	reg reset_circ = 0;
	reg [8:0] max_tempo = 300;
	reg [9:0] col_count = 0;

	always@(posedge vsync) begin
		if (reset1 | (reset_circ != circle))  begin
			init <= 0;
			shift <= 0;
			currentram <= 0;
			addr_count <= 0;
			reset_circ <= 0;
			col_count <= 0; end
		else begin
			init <= 1;
			currentram <= ~currentram;
			if (count< temp) begin
			  count <= count + 1;
			end
			if (count == temp) begin
				count <= 0;
			end
			if (beat_state==BEAT) shift <= 1;
			if (beat_state==RESET) shift <= 0;
			addr_count <= addr_count+1;
		end

		circle <= switch[0] ? 1 : 0;
		reset_circ <= circle;
	end

reg	b,hs,vs;

always @(posedge clk)
  begin
```

```
                b <= blank;
                hs <= hsync;
                vs <= vsync;
  end

  // VGA Output.  In order to meet the setup and hold times of the
  // AD7125, we send it ~clk.
        assign vga_out_red = pixel_out[23:16];
  assign vga_out_green = pixel_out[15:8];
  assign vga_out_blue = pixel_out[7:0];
  assign vga_out_sync_b = 1'b1;    // not used
  assign vga_out_pixel_clock = ~clk;
  assign vga_out_blank_b = ~b;
  assign vga_out_hsync = hs;
  assign vga_out_vsync = vs;

endmodule


module svga1(vclock,hcount,vcount,hsync,vsync,blank);
  input vclock;
  output [10:0] hcount;
  output [9:0] vcount;
  output        vsync;
  output        hsync;
  output        blank;

  reg     hsync,vsync,hblank,vblank,blank;
  reg [10:0]      hcount;    // pixel number on current line
  reg [9:0] vcount;       // line number

  // horizontal: 1056 pixels total
  // display 800 pixels per line
  wire     hsyncon,hsyncoff,hreset,hblankon;
  assign    hblankon = (hcount == 799);
  assign    hsyncon = (hcount == 839);
  assign    hsyncoff = (hcount == 967);
  assign    hreset = (hcount == 1055);

  // vertical: 628 lines total
  // display 600 lines
  wire     vsyncon,vsyncoff,vreset,vblankon;
  assign    vblankon = hreset & (vcount == 599);
  assign    vsyncon = hreset & (vcount == 600);
  assign    vsyncoff = hreset & (vcount == 604);
  assign    vreset = hreset & (vcount == 627);

  // sync and blanking
```

```
  wire      next_hblank,next_vblank;
  assign next_hblank = hreset ? 0 : hblankon ? 1 : hblank;
  assign next_vblank = vreset ? 0 : vblankon ? 1 : vblank;
  always @(posedge vclock) begin
    hcount <= hreset ? 0 : hcount + 1;
    hblank <= next_hblank;
    hsync <= hsyncon ? 0 : hsyncoff ? 1 : hsync;  // active low

    vcount <= hreset ? (vreset ? 0 : vcount + 1) : vcount;
    vblank <= next_vblank;
    vsync <= vsyncon ? 0 : vsyncoff ? 1 : vsync;  // active low

    blank <= next_vblank | (next_hblank & ~hreset);
  end
endmodule

module vram_display1(reset,clk, init, hcount,vcount,vr_pixel,
                     vram_addr,vram_read_data);

  input reset, clk;
  input [10:0] hcount;
  input [9:0]    vcount;
        input init;
  output [23:0] vr_pixel;
  output [18:0] vram_addr;
  input [35:0]  vram_read_data;

        //forecast hcount & vcount 2 clock cycle ahead to get data from ZBT
  wire [10:0] hcount_f = (hcount >= 1054) ? (hcount - 1054) : (hcount + 2);
  wire [9:0] vcount_f = (hcount >= 1054) ? ((vcount == 627) ? 0 : vcount+1) : vcount;

  wire [18:0]     vram_addr =  {hcount_f[10:0] + vcount_f[9:0]*800};

  wire [1:0]      hc4 = hcount[1:0];
  reg [23:0]      vr_pixel;
  reg [35:0]      vr_data_latched;
  reg [35:0]      last_vr_data;

  always @(posedge clk)
    vr_pixel <= vram_read_data[23:0];


endmodule // vram_display

module ramclock1(ref_clock, fpga_clock, ram0_clock, ram1_clock,
                 clock_feedback_in, clock_feedback_out, locked);

  input ref_clock;              // Reference clock input
```

```
output fpga_clock;              // Output clock to drive FPGA logic
output ram0_clock, ram1_clock;  // Output clocks for each RAM chip
input  clock_feedback_in;       // Output to feedback trace
output clock_feedback_out;      // Input from feedback trace
output locked;                  // Indicates that clock outputs are stable

wire  ref_clk, fpga_clk, ram_clk, fb_clk, lock1, lock2, dcm_reset, ram_clock;

//////////////////////////////////////////////////////////////////////////

//To force ISE to compile the ramclock, this line has to be removed.
//IBUFG ref_buf (.O(ref_clk), .I(ref_clock));

        assign ref_clk = ref_clock;

BUFG int_buf (.O(fpga_clock), .I(fpga_clk));

DCM int_dcm (.CLKFB(fpga_clock),
                .CLKIN(ref_clk),
                .RST(dcm_reset),
                .CLK0(fpga_clk),
                .LOCKED(lock1));
// synthesis attribute DLL_FREQUENCY_MODE of int_dcm is "LOW"
// synthesis attribute DUTY_CYCLE_CORRECTION of int_dcm is "TRUE"
// synthesis attribute STARTUP_WAIT of int_dcm is "FALSE"
// synthesis attribute DFS_FREQUENCY_MODE of int_dcm is "LOW"
// synthesis attribute CLK_FEEDBACK of int_dcm  is "1X"
// synthesis attribute CLKOUT_PHASE_SHIFT of int_dcm is "NONE"
// synthesis attribute PHASE_SHIFT of int_dcm is 0

BUFG ext_buf (.O(ram_clock), .I(ram_clk));

IBUFG fb_buf (.O(fb_clk), .I(clock_feedback_in));

DCM ext_dcm (.CLKFB(fb_clk),
                .CLKIN(ref_clk),
                .RST(dcm_reset),
                .CLK0(ram_clk),
                .LOCKED(lock2));
// synthesis attribute DLL_FREQUENCY_MODE of ext_dcm is "LOW"
// synthesis attribute DUTY_CYCLE_CORRECTION of ext_dcm is "TRUE"
// synthesis attribute STARTUP_WAIT of ext_dcm is "FALSE"
// synthesis attribute DFS_FREQUENCY_MODE of ext_dcm is "LOW"
// synthesis attribute CLK_FEEDBACK of ext_dcm  is "1X"
// synthesis attribute CLKOUT_PHASE_SHIFT of ext_dcm is "NONE"
// synthesis attribute PHASE_SHIFT of ext_dcm is 0

SRL16 dcm_rst_sr (.D(1'b0), .CLK(ref_clk), .Q(dcm_reset),
```

```
                        .A0(1'b1), .A1(1'b1), .A2(1'b1), .A3(1'b1));
  // synthesis attribute init of dcm_rst_sr is "000F";


  OFDDRRSE ddr_reg0 (.Q(ram0_clock), .C0(ram_clock), .C1(~ram_clock),
                        .CE (1'b1), .D0(1'b1), .D1(1'b0), .R(1'b0), .S(1'b0));
  OFDDRRSE ddr_reg1 (.Q(ram1_clock), .C0(ram_clock), .C1(~ram_clock),
                        .CE (1'b1), .D0(1'b1), .D1(1'b0), .R(1'b0), .S(1'b0));
  OFDDRRSE ddr_reg2 (.Q(clock_feedback_out), .C0(ram_clock), .C1(~ram_clock),
                        .CE (1'b1), .D0(1'b1), .D1(1'b0), .R(1'b0), .S(1'b0));

  assign locked = lock1 && lock2;

endmodule

//Audio Processing unit

module AudioProcessingUnit
   (input clk, reset, ready,
          input signed [7:0]bandx,
          output signed [7:0]comb60,comb90,comb120,comb180,comb210,comb240,
          output signed [7:0] hann_clip,diff_out,fw_rect_band,hw_rect_band);

          //define signals
          wire signed [27:0]hann_out;
//        wire signed [7:0]diff_out;
//        wire signed [7:0]fw_rect_band;
//        wire signed [7:0]hw_rect_band;

          assign hann_clip = hann_out[21:14];

          //full wave rectify to make positive
          assign fw_rect_band = (bandx>0)?bandx:(-bandx);
          //low pass filter to get envelope
          hannfilter
HannWindow1(.clk(clk),.reset(reset),.ready(ready),.x(fw_rect_band),.y(hann_out));
          //differentiate to find sudden jumps
          Differentiator diff1(.clk(clk),.reset(reset),.ready(ready),.x(hann_clip),.y(diff_out));
          //half wave rectify to only get positive jumps
          assign hw_rect_band = (diff_out>0)?diff_out:(8'd0);
          //comb signal for different tempos
          CombFilter comb1(.clk(clk),.reset(reset),.ready(ready),.x(hw_rect_band),

.comb60(comb60),.comb90(comb90),.comb120(comb120),.comb180(comb180),.comb210(comb
210),.comb240(comb240));

endmodule
```

```
module band0coeffs(
  input wire [5:0] index,
  output reg signed [9:0] coeff
);
  // lowpass 200 Hz
  // tools will turn this into a 61x10 ROM
  always @(index)
    case (index)
      5'd0:  coeff = 10'sd84;
      5'd1:  coeff = -10'sd9;
      5'd2:  coeff = -10'sd10;
      5'd3:  coeff = -10'sd11;
      5'd4:  coeff = -10'sd12;
      5'd5:  coeff = -10'sd14;
      5'd6:  coeff = -10'sd16;
      5'd7:  coeff = -10'sd18;
      5'd8:  coeff = -10'sd19;
      5'd9:  coeff = -10'sd21;
      5'd10: coeff = -10'sd21;
      5'd11: coeff = -10'sd21;
      5'd12: coeff = -10'sd20;
      5'd13: coeff = -10'sd18;
      5'd14: coeff = -10'sd15;
      5'd15: coeff = -10'sd11;
      5'd16: coeff = -10'sd6;
      5'd17: coeff = 10'sd0;
      5'd18: coeff = 10'sd6;
      5'd19: coeff = 10'sd13;
      5'd20: coeff = 10'sd21;
      5'd21: coeff = 10'sd29;
      5'd22: coeff = 10'sd37;
      5'd23: coeff = 10'sd45;
      5'd24: coeff = 10'sd52;
      5'd25: coeff = 10'sd59;
      5'd26: coeff = 10'sd65;
      5'd27: coeff = 10'sd70;
      5'd28: coeff = 10'sd73;
      5'd29: coeff = 10'sd75;
      5'd30: coeff = 10'sd76;
                    5'd31: coeff = 10'sd75;
      5'd32: coeff = 10'sd73;
      5'd33: coeff = 10'sd70;
      5'd34: coeff = 10'sd65;
      5'd35: coeff = 10'sd59;
      5'd36: coeff = 10'sd52;
      5'd37: coeff = 10'sd45;
      5'd38: coeff = 10'sd37;
      5'd39: coeff = 10'sd29;
```

```
      5'd40: coeff = 10'sd21;
      5'd41: coeff = 10'sd13;
      5'd42: coeff = 10'sd6;
      5'd43: coeff = 10'sd0;
      5'd44: coeff = -10'sd6;
      5'd45: coeff = -10'sd11;
      5'd46: coeff = -10'sd15;
      5'd47: coeff = -10'sd18;
      5'd48: coeff = -10'sd20;
      5'd49: coeff = -10'sd21;
      5'd50: coeff = -10'sd21;
      5'd51: coeff = -10'sd21;
      5'd52: coeff = -10'sd19;
      5'd53: coeff = -10'sd18;
      5'd54: coeff = -10'sd16;
      5'd55: coeff = -10'sd14;
      5'd56: coeff = -10'sd12;
      5'd57: coeff = -10'sd11;
      5'd58: coeff = -10'sd10;
      5'd59: coeff = -10'sd9;
      5'd60: coeff = 10'sd84;
      default: coeff = 10'hXXX;
    endcase
endmodule


module band1coeffs(
  input wire [5:0] index,
  output reg signed [9:0] coeff
);
  // bandpass 200 to 400 Hz
  // tools will turn this into a 61x10 ROM
  always @(index)
    case (index)
      5'd0: coeff = 10'sd0;
      5'd1: coeff = -10'sd1;
      5'd2: coeff = -10'sd1;
      5'd3: coeff = -10'sd2;
      5'd4: coeff = -10'sd3;
      5'd5: coeff = -10'sd5;
      5'd6: coeff = -10'sd7;
      5'd7: coeff = -10'sd9;
      5'd8: coeff = -10'sd12;
      5'd9: coeff = -10'sd15;
      5'd10: coeff = -10'sd17;
      5'd11: coeff = -10'sd20;
      5'd12: coeff = -10'sd22;
      5'd13: coeff = -10'sd23;
```

```
5'd14: coeff = -10'sd24;
5'd15: coeff = -10'sd23;
5'd16: coeff = -10'sd21;
5'd17: coeff = -10'sd18;
5'd18: coeff = -10'sd13;
5'd19: coeff = -10'sd7;
5'd20: coeff = 10'sd0;
5'd21: coeff = 10'sd8;
5'd22: coeff = 10'sd17;
5'd23: coeff = 10'sd26;
5'd24: coeff = 10'sd36;
5'd25: coeff = 10'sd44;
5'd26: coeff = 10'sd52;
5'd27: coeff = 10'sd59;
5'd28: coeff = 10'sd63;
5'd29: coeff = 10'sd67;
5'd30: coeff = 10'sd68;
5'd31: coeff = 10'sd67;
5'd32: coeff = 10'sd63;
5'd33: coeff = 10'sd59;
5'd34: coeff = 10'sd52;
5'd35: coeff = 10'sd44;
5'd36: coeff = 10'sd36;
5'd37: coeff = 10'sd26;
5'd38: coeff = 10'sd17;
5'd39: coeff = 10'sd8;
5'd40: coeff = 10'sd0;
5'd41: coeff = -10'sd7;
5'd42: coeff = -10'sd13;
5'd43: coeff = -10'sd18;
5'd44: coeff = -10'sd21;
5'd45: coeff = -10'sd23;
5'd46: coeff = -10'sd24;
5'd47: coeff = -10'sd23;
5'd48: coeff = -10'sd22;
5'd49: coeff = -10'sd20;
5'd50: coeff = -10'sd17;
5'd51: coeff = -10'sd15;
5'd52: coeff = -10'sd12;
5'd53: coeff = -10'sd9;
5'd54: coeff = -10'sd7;
5'd55: coeff = -10'sd5;
5'd56: coeff = -10'sd3;
5'd57: coeff = -10'sd2;
5'd58: coeff = -10'sd1;
5'd59: coeff = -10'sd1;
5'd60: coeff = 10'sd0;
default: coeff = 10'hXXX;
```

```
            endcase
endmodule

module band2coeffs(
  input wire [5:0] index,
  output reg signed [9:0] coeff
);
  // bandpass 400 to 800 Hz
  // tools will turn this into a 61x10 ROM
  always @(index)
    case (index)
      5'd0: coeff = 10'sd0;
      5'd1: coeff = 10'sd0;
      5'd2: coeff = 10'sd0;
      5'd3: coeff = -10'sd1;
      5'd4: coeff = 10'sd0;
      5'd5: coeff = 10'sd0;
      5'd6: coeff = 10'sd1;
      5'd7: coeff = 10'sd3;
      5'd8: coeff = 10'sd5;
      5'd9: coeff = 10'sd8;
      5'd10: coeff = 10'sd10;
      5'd11: coeff = 10'sd12;
      5'd12: coeff = 10'sd13;
      5'd13: coeff = 10'sd12;
      5'd14: coeff = 10'sd7;
      5'd15: coeff = 10'sd0;
      5'd16: coeff = -10'sd10;
      5'd17: coeff = -10'sd22;
      5'd18: coeff = -10'sd34;
      5'd19: coeff = -10'sd44;
      5'd20: coeff = -10'sd52;
      5'd21: coeff = -10'sd54;
      5'd22: coeff = -10'sd49;
      5'd23: coeff = -10'sd38;
      5'd24: coeff = -10'sd21;
      5'd25: coeff = 10'sd0;
      5'd26: coeff = 10'sd23;
      5'd27: coeff = 10'sd46;
      5'd28: coeff = 10'sd64;
      5'd29: coeff = 10'sd77;
      5'd30: coeff = 10'sd81;
      5'd31: coeff = 10'sd77;
      5'd32: coeff = 10'sd64;
      5'd33: coeff = 10'sd46;
      5'd34: coeff = 10'sd23;
      5'd35: coeff = 10'sd0;
      5'd36: coeff = -10'sd21;
```

```
      5'd37: coeff = -10'sd38;
      5'd38: coeff = -10'sd49;
      5'd39: coeff = -10'sd54;
      5'd40: coeff = -10'sd52;
      5'd41: coeff = -10'sd44;
      5'd42: coeff = -10'sd34;
      5'd43: coeff = -10'sd22;
      5'd44: coeff = -10'sd10;
      5'd45: coeff = 10'sd0;
      5'd46: coeff = 10'sd7;
      5'd47: coeff = 10'sd12;
      5'd48: coeff = 10'sd13;
      5'd49: coeff = 10'sd12;
      5'd50: coeff = 10'sd10;
      5'd51: coeff = 10'sd8;
      5'd52: coeff = 10'sd5;
      5'd53: coeff = 10'sd3;
      5'd54: coeff = 10'sd1;
      5'd55: coeff = 10'sd0;
      5'd56: coeff = 10'sd0;
      5'd57: coeff = -10'sd1;
      5'd58: coeff = 10'sd0;
      5'd59: coeff = 10'sd0;
      5'd60: coeff = 10'sd0;
      default: coeff = 10'hXXX;
          endcase
endmodule

module band3coeffs(
  input wire [5:0] index,
  output reg signed [9:0] coeff
);
  // bandpass 800 to 1600 Hz
  // tools will turn this into a 61x10 ROM
  always @(index)
    case (index)
      5'd0: coeff = 10'sd0;
      5'd1: coeff = 10'sd0;
      5'd2: coeff = 10'sd0;
      5'd3: coeff = 10'sd0;
      5'd4: coeff = 10'sd2;
      5'd5: coeff = 10'sd3;
      5'd6: coeff = 10'sd3;
      5'd7: coeff = 10'sd2;
      5'd8: coeff = -10'sd2;
      5'd9: coeff = -10'sd6;
      5'd10: coeff = -10'sd9;
      5'd11: coeff = -10'sd7;
```

```
5'd12: coeff = -10'sd3;
5'd13: coeff = 10'sd2;
5'd14: coeff = 10'sd3;
5'd15: coeff = 10'sd0;
5'd16: coeff = -10'sd5;
5'd17: coeff = -10'sd4;
5'd18: coeff = 10'sd7;
5'd19: coeff = 10'sd26;
5'd20: coeff = 10'sd43;
5'd21: coeff = 10'sd45;
5'd22: coeff = 10'sd21;
5'd23: coeff = -10'sd25;
5'd24: coeff = -10'sd76;
5'd25: coeff = -10'sd106;
5'd26: coeff = -10'sd94;
5'd27: coeff = -10'sd38;
5'd28: coeff = 10'sd40;
5'd29: coeff = 10'sd109;
5'd30: coeff = 10'sd136;
5'd31: coeff = 10'sd109;
5'd32: coeff = 10'sd40;
5'd33: coeff = -10'sd38;
5'd34: coeff = -10'sd94;
5'd35: coeff = -10'sd106;
5'd36: coeff = -10'sd76;
5'd37: coeff = -10'sd25;
5'd38: coeff = 10'sd21;
5'd39: coeff = 10'sd45;
5'd40: coeff = 10'sd43;
5'd41: coeff = 10'sd26;
5'd42: coeff = 10'sd7;
5'd43: coeff = -10'sd4;
5'd44: coeff = -10'sd5;
5'd45: coeff = 10'sd0;
5'd46: coeff = 10'sd3;
5'd47: coeff = 10'sd2;
5'd48: coeff = -10'sd3;
5'd49: coeff = -10'sd7;
5'd50: coeff = -10'sd9;
5'd51: coeff = -10'sd6;
5'd52: coeff = -10'sd2;
5'd53: coeff = 10'sd2;
5'd54: coeff = 10'sd3;
5'd55: coeff = 10'sd3;
5'd56: coeff = 10'sd2;
5'd57: coeff = 10'sd0;
5'd58: coeff = 10'sd0;
5'd59: coeff = 10'sd0;
```

```
      5'd60: coeff = 10'sd0;
      default: coeff = 10'hXXX;
           endcase
endmodule

module band4coeffs(
  input wire [5:0] index,
  output reg signed [9:0] coeff
);
  // bandpass 1600 to 3200 Hz
  // tools will turn this into a 61x10 ROM
  always @(index)
    case (index)
      5'd0: coeff = 10'sd0;
      5'd1: coeff = 10'sd0;
      5'd2: coeff = 10'sd1;
      5'd3: coeff = 10'sd2;
      5'd4: coeff = -10'sd1;
      5'd5: coeff = -10'sd3;
      5'd6: coeff = -10'sd1;
      5'd7: coeff = 10'sd1;
      5'd8: coeff = -10'sd1;
      5'd9: coeff = 10'sd2;
      5'd10: coeff = 10'sd9;
      5'd11: coeff = 10'sd4;
      5'd12: coeff = -10'sd11;
      5'd13: coeff = -10'sd10;
      5'd14: coeff = 10'sd3;
      5'd15: coeff = 10'sd0;
      5'd16: coeff = -10'sd3;
      5'd17: coeff = 10'sd19;
      5'd18: coeff = 10'sd29;
      5'd19: coeff = -10'sd13;
      5'd20: coeff = -10'sd44;
      5'd21: coeff = -10'sd11;
      5'd22: coeff = 10'sd12;
      5'd23: coeff = -10'sd14;
      5'd24: coeff = 10'sd18;
      5'd25: coeff = 10'sd106;
      5'd26: coeff = 10'sd48;
      5'd27: coeff = -10'sd164;
      5'd28: coeff = -10'sd194;
      5'd29: coeff = 10'sd82;
      5'd30: coeff = 10'sd274;
      5'd31: coeff = 10'sd82;
      5'd32: coeff = -10'sd194;
      5'd33: coeff = -10'sd164;
      5'd34: coeff = 10'sd48;
```

```
    5'd35: coeff = 10'sd106;
    5'd36: coeff = 10'sd18;
    5'd37: coeff = -10'sd14;
    5'd38: coeff = 10'sd12;
    5'd39: coeff = -10'sd11;
    5'd40: coeff = -10'sd44;
    5'd41: coeff = -10'sd13;
    5'd42: coeff = 10'sd29;
    5'd43: coeff = 10'sd19;
    5'd44: coeff = -10'sd3;
    5'd45: coeff = 10'sd0;
    5'd46: coeff = 10'sd3;
    5'd47: coeff = -10'sd10;
    5'd48: coeff = -10'sd11;
    5'd49: coeff = 10'sd4;
    5'd50: coeff = 10'sd9;
    5'd51: coeff = 10'sd2;
    5'd52: coeff = -10'sd1;
    5'd53: coeff = 10'sd1;
    5'd54: coeff = -10'sd1;
    5'd55: coeff = -10'sd3;
    5'd56: coeff = -10'sd1;
    5'd57: coeff = 10'sd2;
    5'd58: coeff = 10'sd1;
    5'd59: coeff = 10'sd0;
    5'd60: coeff = 10'sd0;
    default: coeff = 10'hXXX;
        endcase
endmodule


module BandpassFilter0(
  input wire clock,reset,ready,
  input wire signed [7:0] x,
  output reg signed [17:0] y
);

        reg signed[17:0]accumulator = 0;
        reg [5:0]offset = 0;
        reg signed [7:0] sample [61:0];
        reg [5:0]index = 0;
        reg [6:0]counter = 0;
        wire signed [9:0]coeff;

        band0coeffs coeffs(.index(index),.coeff(coeff));

        always @(posedge clock)
                begin
```

```
		if (ready)
			begin
			offset <= offset + 1;
			sample[offset] <= x;
			accumulator <= 0;
			index <= 0;
			counter <= 0;
			end
		else if (counter < 61)
			begin
			accumulator <= accumulator + coeff*sample[offset-index];
			index <= index + 1;
			counter <= counter + 1;
			end
		else if (counter == 61) y <= accumulator;
		end

endmodule

module BandpassFilter1(
  input wire clock,reset,ready,
  input wire signed [7:0] x,
  output reg signed [17:0] y
);

	reg signed[17:0]accumulator = 0;
	reg [5:0]offset = 0;
	reg signed [7:0] sample [61:0];
	reg [5:0]index = 0;
	reg [6:0]counter = 0;
	wire signed [9:0]coeff;

	band1coeffs coeffs(.index(index),.coeff(coeff));

	always @(posedge clock)
		begin
		if (ready)
			begin
			offset <= offset + 1;
			sample[offset] <= x;
			accumulator <= 0;
			index <= 0;
			counter <= 0;
			end
		else if (counter < 61)
			begin
			accumulator <= accumulator + coeff*sample[offset-index];
			index <= index + 1;
```

```
                counter <= counter + 1;
                end
        else if (counter == 61) y <= accumulator;
        end

endmodule

module BandpassFilter2(
  input wire clock,reset,ready,
  input wire signed [7:0] x,
  output reg signed [17:0] y
);

        reg signed[17:0]accumulator = 0;
        reg [5:0]offset = 0;
        reg signed [7:0] sample [61:0];
        reg [5:0]index = 0;
        reg [6:0]counter = 0;
        wire signed [9:0]coeff;

        band2coeffs coeffs(.index(index),.coeff(coeff));

        always @(posedge clock)
                begin
                if (ready)
                        begin
                        offset <= offset + 1;
                        sample[offset] <= x;
                        accumulator <= 0;
                        index <= 0;
                        counter <= 0;
                        end
                else if (counter < 61)
                        begin
                        accumulator <= accumulator + coeff*sample[offset-index];
                        index <= index + 1;
                        counter <= counter + 1;
                        end
                else if (counter == 61) y <= accumulator;
                end

endmodule

module BandpassFilter3(
  input wire clock,reset,ready,
  input wire signed [7:0] x,
  output reg signed [17:0] y
);
```

```
	reg signed[17:0]accumulator = 0;
	reg [5:0]offset = 0;
	reg signed [7:0] sample [61:0];
	reg [5:0]index = 0;
	reg [6:0]counter = 0;
	wire signed [9:0]coeff;

	band3coeffs coeffs(.index(index),.coeff(coeff));

	always @(posedge clock)
		begin
		if (ready)
			begin
			offset <= offset + 1;
			sample[offset] <= x;
			accumulator <= 0;
			index <= 0;
			counter <= 0;
			end
		else if (counter < 61)
			begin
			accumulator <= accumulator + coeff*sample[offset-index];
			index <= index + 1;
			counter <= counter + 1;
			end
		else if (counter == 61) y <= accumulator;
		end

endmodule

module BandpassFilter4(
  input wire clock,reset,ready,
  input wire signed [7:0] x,
  output reg signed [17:0] y
);

	reg signed[17:0]accumulator = 0;
	reg [5:0]offset = 0;
	reg signed [7:0] sample [61:0];
	reg [5:0]index = 0;
	reg [6:0]counter = 0;
	wire signed [9:0]coeff;

	band4coeffs coeffs(.index(index),.coeff(coeff));

	always @(posedge clock)
		begin
```

```
                if (ready)
                        begin
                        offset <= offset + 1;
                        sample[offset] <= x;
                        accumulator <= 0;
                        index <= 0;
                        counter <= 0;
                        end
                else if (counter < 61)
                        begin
                        accumulator <= accumulator + coeff*sample[offset-index];
                        index <= index + 1;
                        counter <= counter + 1;
                        end
                else if (counter == 61) y <= accumulator;
                end

endmodule

module mybram #(parameter LOGSIZE=14, WIDTH=1)
            (input wire [LOGSIZE-1:0] addr,
             input wire clk,
             input wire [WIDTH-1:0] din,
             output reg [WIDTH-1:0] dout,
             input wire we);
  // let the tools infer the right number of BRAMs
  (* ram_style = "block" *)
  reg [WIDTH-1:0] mem[LOGSIZE-1:0];
  always @(posedge clk) begin
    if (we) mem[addr] <= din;
    dout <= mem[addr];
  end
endmodule

module CombFilter
        (input clk, reset, ready,
        input signed [7:0]x,
        output reg signed [7:0]comb60,comb90,comb120,comb180,comb210,comb240);

        parameter SHIFTSIZE = 12000;
        reg [13:0]addr = 0;
        reg signed [7:0]memin;
        wire signed [7:0]memout;
        reg we = 0;
        reg [13:0]offset = 0;
        reg [4:0]tap = 0;

        reg signed [7:0]buffer_0 = 0;
```

```
reg signed [7:0]buffer_1499 = 0;
reg signed [7:0]buffer_1713 = 0;
reg signed [7:0]buffer_1999 = 0;
reg signed [7:0]buffer_2999 = 0;
reg signed [7:0]buffer_3427 = 0;
reg signed [7:0]buffer_3999 = 0;
reg signed [7:0]buffer_5999 = 0;
reg signed [7:0]buffer_7999 = 0;
reg signed [7:0]buffer_11999 = 0;

mybram #(.LOGSIZE(14),.WIDTH(8))
	mybram1(.addr(addr),.clk(clk),.din(memin),.dout(memout),.we(we));


always @(posedge clk)
	begin
	if (ready && tap==0)
		begin
		we <= 1;
		memin <= x;
		addr <= offset;
		buffer_0 <= x;
		tap <= 1;
		end
	if (tap==1)
		begin
		we <= 0;
		addr <= (offset>1500)?(offset-1499):(SHIFTSIZE+offset-1499);
		tap <= 2;
		end
	if (tap==2)
		begin
		buffer_1499 <= memout;
		addr <= (offset>1714)?(offset-1713):(SHIFTSIZE+offset-1713);
		tap <= 3;
		end
	if (tap==3)
		begin
		buffer_1713 <= memout;
		addr <= (offset>2000)?(offset-1999):(SHIFTSIZE+offset-1999);
		tap <= 4;
		end
	if (tap==4)
		begin
		buffer_1999 <= memout;
		addr <= (offset>3000)?(offset-2999):(SHIFTSIZE+offset-2999);
		tap <= 5;
		end
```

```
if (tap==5)
        begin
        buffer_2999 <= memout;
        addr <= (offset>3428)?(offset-3427):(SHIFTSIZE+offset-3427);
        tap <= 6;
        end
if (tap==6)
        begin
        buffer_3427 <= memout;
        addr <= (offset>4000)?(offset-3999):(SHIFTSIZE+offset-3999);
        tap <= 7;
        end
if (tap==7)
        begin
        buffer_3999 <= memout;
        addr <= (offset>6000)?(offset-5999):(SHIFTSIZE+offset-5999);
        tap <= 8;
        end
if (tap==8)
        begin
        buffer_5999 <= memout;
        addr <= (offset>8000)?(offset-7999):(SHIFTSIZE+offset-7999);
        tap <= 9;
        end
if (tap==9)
        begin
        buffer_7999 <= memout;
        addr <= (offset>12000)?(offset-11999):(SHIFTSIZE+offset-11999);
        tap <= 10;
        end
if (tap==10)
        begin
        tap <= 0;
        comb60 <= buffer_0 + buffer_5999 + memout;
        comb90 <= buffer_0 + buffer_3999 + buffer_7999;
        comb120 <= buffer_0 + buffer_2999 + buffer_5999;
        comb180 <= buffer_0 + buffer_1999 + buffer_3999;
        comb210 <= buffer_0 + buffer_1713 + buffer_3427;
        comb240 <= buffer_0 + buffer_1499 + buffer_2999;
        offset <= offset<SHIFTSIZE-2?offset+1:0;
        end
end


endmodule
```

module Differentiator

```
    (input clk, reset, ready,
            input signed [7:0]x,
    output reg signed [7:0]y);

            reg signed [7:0]buffer;

            always @(posedge clk)
                    begin
                    if (ready)
                            buffer <= x;
                            y <= (x-buffer)*16;
                    end

endmodule
```

module Filterbank

```
    (input clk, reset, ready,
            input signed [7:0]x,
            output reg sixk_ready,
    output signed [7:0]band0,band1,band2,band3,band4);

            //initialize things
            reg [2:0]counter = 0;
            reg [7:0]data_in;
            wire signed [17:0]out0;
            wire signed [17:0]out1;
            wire signed [17:0]out2;
            wire signed [17:0]out3;
            wire signed [17:0]out4;

            //connect and scale outputs
            assign band0[7:0] = out0[17:10];
            assign band1[7:0] = out1[17:10];
            assign band2[7:0] = out2[17:10];
            assign band3[7:0] = out3[17:10];
            assign band4[7:0] = out4[17:10];

            BandpassFilter0 bpf0(.clock(clk),.reset(reset),.ready(sixk_ready),.x(data_in),.y(out0));
            BandpassFilter1 bpf1(.clock(clk),.reset(reset),.ready(sixk_ready),.x(data_in),.y(out1));
            BandpassFilter2 bpf2(.clock(clk),.reset(reset),.ready(sixk_ready),.x(data_in),.y(out2));
            BandpassFilter3 bpf3(.clock(clk),.reset(reset),.ready(sixk_ready),.x(data_in),.y(out3));
            BandpassFilter4 bpf4(.clock(clk),.reset(reset),.ready(sixk_ready),.x(data_in),.y(out4));

            always @(posedge clk)
                    begin
                    sixk_ready <= 0;
                    if (ready)
                            begin
```

```
                        if (counter == 7)
                                begin
                                counter <= 0;
                                data_in <= x;
                                sixk_ready <= 1;
                                end
                        else
                                begin
                                counter <= counter + 1;
                                end
                        end
                end
endmodule

module hanncoeffs4000(
        input wire [11:0]index,
        output reg signed [9:0]coeff
        );
        always @(index)
                case(index)
                12'd0: coeff = 10'sd0;
                12'd1: coeff = 10'sd0;
                12'd2: coeff = 10'sd0;
                12'd3: coeff = 10'sd0;
                12'd4: coeff = 10'sd0;
                12'd5: coeff = 10'sd0;
                12'd6: coeff = 10'sd0;
                12'd7: coeff = 10'sd0;
                12'd8: coeff = 10'sd0;
                12'd9: coeff = 10'sd0;
                12'd10: coeff = 10'sd0;
                12'd11: coeff = 10'sd0;
                12'd12: coeff = 10'sd0;
                12'd13: coeff = 10'sd0;
                12'd14: coeff = 10'sd0;
                12'd15: coeff = 10'sd0;
                12'd16: coeff = 10'sd0;
                12'd17: coeff = 10'sd0;
                12'd18: coeff = 10'sd0;
                12'd19: coeff = 10'sd0;
                12'd20: coeff = 10'sd0;
                12'd21: coeff = 10'sd0;
                12'd22: coeff = 10'sd0;
                12'd23: coeff = 10'sd0;
                12'd24: coeff = 10'sd0;
                12'd25: coeff = 10'sd0;
                12'd26: coeff = 10'sd0;
                12'd27: coeff = 10'sd0;
```

12'd28: coeff = 10'sd0;
12'd29: coeff = 10'sd1;
12'd30: coeff = 10'sd1;
12'd31: coeff = 10'sd1;
12'd32: coeff = 10'sd1;
12'd33: coeff = 10'sd1;
12'd34: coeff = 10'sd1;
12'd35: coeff = 10'sd1;
12'd36: coeff = 10'sd1;
12'd37: coeff = 10'sd1;
12'd38: coeff = 10'sd1;
12'd39: coeff = 10'sd1;
12'd40: coeff = 10'sd1;
12'd41: coeff = 10'sd1;
12'd42: coeff = 10'sd1;
12'd43: coeff = 10'sd1;
12'd44: coeff = 10'sd1;
12'd45: coeff = 10'sd1;
12'd46: coeff = 10'sd1;
12'd47: coeff = 10'sd1;
12'd48: coeff = 10'sd1;
12'd49: coeff = 10'sd2;
12'd50: coeff = 10'sd2;
12'd51: coeff = 10'sd2;
12'd52: coeff = 10'sd2;
12'd53: coeff = 10'sd2;
12'd54: coeff = 10'sd2;
12'd55: coeff = 10'sd2;
12'd56: coeff = 10'sd2;
12'd57: coeff = 10'sd2;
12'd58: coeff = 10'sd2;
12'd59: coeff = 10'sd2;
12'd60: coeff = 10'sd2;
12'd61: coeff = 10'sd2;
12'd62: coeff = 10'sd2;
12'd63: coeff = 10'sd3;
12'd64: coeff = 10'sd3;
12'd65: coeff = 10'sd3;
12'd66: coeff = 10'sd3;
12'd67: coeff = 10'sd3;
12'd68: coeff = 10'sd3;
12'd69: coeff = 10'sd3;
12'd70: coeff = 10'sd3;
12'd71: coeff = 10'sd3;
12'd72: coeff = 10'sd3;
12'd73: coeff = 10'sd3;
12'd74: coeff = 10'sd3;
12'd75: coeff = 10'sd4;

```
12'd76: coeff = 10'sd4;
12'd77: coeff = 10'sd4;
12'd78: coeff = 10'sd4;
12'd79: coeff = 10'sd4;
12'd80: coeff = 10'sd4;
12'd81: coeff = 10'sd4;
12'd82: coeff = 10'sd4;
12'd83: coeff = 10'sd4;
12'd84: coeff = 10'sd4;
12'd85: coeff = 10'sd5;
12'd86: coeff = 10'sd5;
12'd87: coeff = 10'sd5;
12'd88: coeff = 10'sd5;
12'd89: coeff = 10'sd5;
12'd90: coeff = 10'sd5;
12'd91: coeff = 10'sd5;
12'd92: coeff = 10'sd5;
12'd93: coeff = 10'sd5;
12'd94: coeff = 10'sd6;
12'd95: coeff = 10'sd6;
12'd96: coeff = 10'sd6;
12'd97: coeff = 10'sd6;
12'd98: coeff = 10'sd6;
12'd99: coeff = 10'sd6;
12'd100: coeff = 10'sd6;
12'd101: coeff = 10'sd6;
12'd102: coeff = 10'sd7;
12'd103: coeff = 10'sd7;
12'd104: coeff = 10'sd7;
12'd105: coeff = 10'sd7;
12'd106: coeff = 10'sd7;
12'd107: coeff = 10'sd7;
12'd108: coeff = 10'sd7;
12'd109: coeff = 10'sd7;
12'd110: coeff = 10'sd8;
12'd111: coeff = 10'sd8;
12'd112: coeff = 10'sd8;
12'd113: coeff = 10'sd8;
12'd114: coeff = 10'sd8;
12'd115: coeff = 10'sd8;
12'd116: coeff = 10'sd8;
12'd117: coeff = 10'sd9;
12'd118: coeff = 10'sd9;
12'd119: coeff = 10'sd9;
12'd120: coeff = 10'sd9;
12'd121: coeff = 10'sd9;
12'd122: coeff = 10'sd9;
12'd123: coeff = 10'sd10;
```

```
12'd124: coeff = 10'sd10;
12'd125: coeff = 10'sd10;
12'd126: coeff = 10'sd10;
12'd127: coeff = 10'sd10;
12'd128: coeff = 10'sd10;
12'd129: coeff = 10'sd10;
12'd130: coeff = 10'sd11;
12'd131: coeff = 10'sd11;
12'd132: coeff = 10'sd11;
12'd133: coeff = 10'sd11;
12'd134: coeff = 10'sd11;
12'd135: coeff = 10'sd11;
12'd136: coeff = 10'sd12;
12'd137: coeff = 10'sd12;
12'd138: coeff = 10'sd12;
12'd139: coeff = 10'sd12;
12'd140: coeff = 10'sd12;
12'd141: coeff = 10'sd13;
12'd142: coeff = 10'sd13;
12'd143: coeff = 10'sd13;
12'd144: coeff = 10'sd13;
12'd145: coeff = 10'sd13;
12'd146: coeff = 10'sd13;
12'd147: coeff = 10'sd14;
12'd148: coeff = 10'sd14;
12'd149: coeff = 10'sd14;
12'd150: coeff = 10'sd14;
12'd151: coeff = 10'sd14;
12'd152: coeff = 10'sd15;
12'd153: coeff = 10'sd15;
12'd154: coeff = 10'sd15;
12'd155: coeff = 10'sd15;
12'd156: coeff = 10'sd15;
12'd157: coeff = 10'sd15;
12'd158: coeff = 10'sd16;
12'd159: coeff = 10'sd16;
12'd160: coeff = 10'sd16;
12'd161: coeff = 10'sd16;
12'd162: coeff = 10'sd16;
12'd163: coeff = 10'sd17;
12'd164: coeff = 10'sd17;
12'd165: coeff = 10'sd17;
12'd166: coeff = 10'sd17;
12'd167: coeff = 10'sd18;
12'd168: coeff = 10'sd18;
12'd169: coeff = 10'sd18;
12'd170: coeff = 10'sd18;
12'd171: coeff = 10'sd18;
```

```
12'd172: coeff = 10'sd19;
12'd173: coeff = 10'sd19;
12'd174: coeff = 10'sd19;
12'd175: coeff = 10'sd19;
12'd176: coeff = 10'sd19;
12'd177: coeff = 10'sd20;
12'd178: coeff = 10'sd20;
12'd179: coeff = 10'sd20;
12'd180: coeff = 10'sd20;
12'd181: coeff = 10'sd21;
12'd182: coeff = 10'sd21;
12'd183: coeff = 10'sd21;
12'd184: coeff = 10'sd21;
12'd185: coeff = 10'sd21;
12'd186: coeff = 10'sd22;
12'd187: coeff = 10'sd22;
12'd188: coeff = 10'sd22;
12'd189: coeff = 10'sd22;
12'd190: coeff = 10'sd23;
12'd191: coeff = 10'sd23;
12'd192: coeff = 10'sd23;
12'd193: coeff = 10'sd23;
12'd194: coeff = 10'sd24;
12'd195: coeff = 10'sd24;
12'd196: coeff = 10'sd24;
12'd197: coeff = 10'sd24;
12'd198: coeff = 10'sd25;
12'd199: coeff = 10'sd25;
12'd200: coeff = 10'sd25;
12'd201: coeff = 10'sd25;
12'd202: coeff = 10'sd26;
12'd203: coeff = 10'sd26;
12'd204: coeff = 10'sd26;
12'd205: coeff = 10'sd26;
12'd206: coeff = 10'sd27;
12'd207: coeff = 10'sd27;
12'd208: coeff = 10'sd27;
12'd209: coeff = 10'sd27;
12'd210: coeff = 10'sd28;
12'd211: coeff = 10'sd28;
12'd212: coeff = 10'sd28;
12'd213: coeff = 10'sd28;
12'd214: coeff = 10'sd29;
12'd215: coeff = 10'sd29;
12'd216: coeff = 10'sd29;
12'd217: coeff = 10'sd29;
12'd218: coeff = 10'sd30;
12'd219: coeff = 10'sd30;
```

```
12'd220: coeff = 10'sd30;
12'd221: coeff = 10'sd31;
12'd222: coeff = 10'sd31;
12'd223: coeff = 10'sd31;
12'd224: coeff = 10'sd31;
12'd225: coeff = 10'sd32;
12'd226: coeff = 10'sd32;
12'd227: coeff = 10'sd32;
12'd228: coeff = 10'sd33;
12'd229: coeff = 10'sd33;
12'd230: coeff = 10'sd33;
12'd231: coeff = 10'sd33;
12'd232: coeff = 10'sd34;
12'd233: coeff = 10'sd34;
12'd234: coeff = 10'sd34;
12'd235: coeff = 10'sd35;
12'd236: coeff = 10'sd35;
12'd237: coeff = 10'sd35;
12'd238: coeff = 10'sd35;
12'd239: coeff = 10'sd36;
12'd240: coeff = 10'sd36;
12'd241: coeff = 10'sd36;
12'd242: coeff = 10'sd37;
12'd243: coeff = 10'sd37;
12'd244: coeff = 10'sd37;
12'd245: coeff = 10'sd37;
12'd246: coeff = 10'sd38;
12'd247: coeff = 10'sd38;
12'd248: coeff = 10'sd38;
12'd249: coeff = 10'sd39;
12'd250: coeff = 10'sd39;
12'd251: coeff = 10'sd39;
12'd252: coeff = 10'sd40;
12'd253: coeff = 10'sd40;
12'd254: coeff = 10'sd40;
12'd255: coeff = 10'sd41;
12'd256: coeff = 10'sd41;
12'd257: coeff = 10'sd41;
12'd258: coeff = 10'sd41;
12'd259: coeff = 10'sd42;
12'd260: coeff = 10'sd42;
12'd261: coeff = 10'sd42;
12'd262: coeff = 10'sd43;
12'd263: coeff = 10'sd43;
12'd264: coeff = 10'sd43;
12'd265: coeff = 10'sd44;
12'd266: coeff = 10'sd44;
12'd267: coeff = 10'sd44;
```

```
12'd268: coeff = 10'sd45;
12'd269: coeff = 10'sd45;
12'd270: coeff = 10'sd45;
12'd271: coeff = 10'sd46;
12'd272: coeff = 10'sd46;
12'd273: coeff = 10'sd46;
12'd274: coeff = 10'sd47;
12'd275: coeff = 10'sd47;
12'd276: coeff = 10'sd47;
12'd277: coeff = 10'sd48;
12'd278: coeff = 10'sd48;
12'd279: coeff = 10'sd48;
12'd280: coeff = 10'sd49;
12'd281: coeff = 10'sd49;
12'd282: coeff = 10'sd49;
12'd283: coeff = 10'sd50;
12'd284: coeff = 10'sd50;
12'd285: coeff = 10'sd50;
12'd286: coeff = 10'sd51;
12'd287: coeff = 10'sd51;
12'd288: coeff = 10'sd52;
12'd289: coeff = 10'sd52;
12'd290: coeff = 10'sd52;
12'd291: coeff = 10'sd53;
12'd292: coeff = 10'sd53;
12'd293: coeff = 10'sd53;
12'd294: coeff = 10'sd54;
12'd295: coeff = 10'sd54;
12'd296: coeff = 10'sd54;
12'd297: coeff = 10'sd55;
12'd298: coeff = 10'sd55;
12'd299: coeff = 10'sd55;
12'd300: coeff = 10'sd56;
12'd301: coeff = 10'sd56;
12'd302: coeff = 10'sd57;
12'd303: coeff = 10'sd57;
12'd304: coeff = 10'sd57;
12'd305: coeff = 10'sd58;
12'd306: coeff = 10'sd58;
12'd307: coeff = 10'sd58;
12'd308: coeff = 10'sd59;
12'd309: coeff = 10'sd59;
12'd310: coeff = 10'sd60;
12'd311: coeff = 10'sd60;
12'd312: coeff = 10'sd60;
12'd313: coeff = 10'sd61;
12'd314: coeff = 10'sd61;
12'd315: coeff = 10'sd61;
```

```
12'd316: coeff = 10'sd62;
12'd317: coeff = 10'sd62;
12'd318: coeff = 10'sd63;
12'd319: coeff = 10'sd63;
12'd320: coeff = 10'sd63;
12'd321: coeff = 10'sd64;
12'd322: coeff = 10'sd64;
12'd323: coeff = 10'sd65;
12'd324: coeff = 10'sd65;
12'd325: coeff = 10'sd65;
12'd326: coeff = 10'sd66;
12'd327: coeff = 10'sd66;
12'd328: coeff = 10'sd66;
12'd329: coeff = 10'sd67;
12'd330: coeff = 10'sd67;
12'd331: coeff = 10'sd68;
12'd332: coeff = 10'sd68;
12'd333: coeff = 10'sd68;
12'd334: coeff = 10'sd69;
12'd335: coeff = 10'sd69;
12'd336: coeff = 10'sd70;
12'd337: coeff = 10'sd70;
12'd338: coeff = 10'sd71;
12'd339: coeff = 10'sd71;
12'd340: coeff = 10'sd71;
12'd341: coeff = 10'sd72;
12'd342: coeff = 10'sd72;
12'd343: coeff = 10'sd73;
12'd344: coeff = 10'sd73;
12'd345: coeff = 10'sd73;
12'd346: coeff = 10'sd74;
12'd347: coeff = 10'sd74;
12'd348: coeff = 10'sd75;
12'd349: coeff = 10'sd75;
12'd350: coeff = 10'sd75;
12'd351: coeff = 10'sd76;
12'd352: coeff = 10'sd76;
12'd353: coeff = 10'sd77;
12'd354: coeff = 10'sd77;
12'd355: coeff = 10'sd78;
12'd356: coeff = 10'sd78;
12'd357: coeff = 10'sd78;
12'd358: coeff = 10'sd79;
12'd359: coeff = 10'sd79;
12'd360: coeff = 10'sd80;
12'd361: coeff = 10'sd80;
12'd362: coeff = 10'sd81;
12'd363: coeff = 10'sd81;
```

```
12'd364: coeff = 10'sd81;
12'd365: coeff = 10'sd82;
12'd366: coeff = 10'sd82;
12'd367: coeff = 10'sd83;
12'd368: coeff = 10'sd83;
12'd369: coeff = 10'sd84;
12'd370: coeff = 10'sd84;
12'd371: coeff = 10'sd85;
12'd372: coeff = 10'sd85;
12'd373: coeff = 10'sd85;
12'd374: coeff = 10'sd86;
12'd375: coeff = 10'sd86;
12'd376: coeff = 10'sd87;
12'd377: coeff = 10'sd87;
12'd378: coeff = 10'sd88;
12'd379: coeff = 10'sd88;
12'd380: coeff = 10'sd89;
12'd381: coeff = 10'sd89;
12'd382: coeff = 10'sd89;
12'd383: coeff = 10'sd90;
12'd384: coeff = 10'sd90;
12'd385: coeff = 10'sd91;
12'd386: coeff = 10'sd91;
12'd387: coeff = 10'sd92;
12'd388: coeff = 10'sd92;
12'd389: coeff = 10'sd93;
12'd390: coeff = 10'sd93;
12'd391: coeff = 10'sd94;
12'd392: coeff = 10'sd94;
12'd393: coeff = 10'sd95;
12'd394: coeff = 10'sd95;
12'd395: coeff = 10'sd95;
12'd396: coeff = 10'sd96;
12'd397: coeff = 10'sd96;
12'd398: coeff = 10'sd97;
12'd399: coeff = 10'sd97;
12'd400: coeff = 10'sd98;
12'd401: coeff = 10'sd98;
12'd402: coeff = 10'sd99;
12'd403: coeff = 10'sd99;
12'd404: coeff = 10'sd100;
12'd405: coeff = 10'sd100;
12'd406: coeff = 10'sd101;
12'd407: coeff = 10'sd101;
12'd408: coeff = 10'sd102;
12'd409: coeff = 10'sd102;
12'd410: coeff = 10'sd103;
12'd411: coeff = 10'sd103;
```

```
12'd412: coeff = 10'sd104;
12'd413: coeff = 10'sd104;
12'd414: coeff = 10'sd105;
12'd415: coeff = 10'sd105;
12'd416: coeff = 10'sd106;
12'd417: coeff = 10'sd106;
12'd418: coeff = 10'sd107;
12'd419: coeff = 10'sd107;
12'd420: coeff = 10'sd107;
12'd421: coeff = 10'sd108;
12'd422: coeff = 10'sd108;
12'd423: coeff = 10'sd109;
12'd424: coeff = 10'sd109;
12'd425: coeff = 10'sd110;
12'd426: coeff = 10'sd110;
12'd427: coeff = 10'sd111;
12'd428: coeff = 10'sd111;
12'd429: coeff = 10'sd112;
12'd430: coeff = 10'sd112;
12'd431: coeff = 10'sd113;
12'd432: coeff = 10'sd113;
12'd433: coeff = 10'sd114;
12'd434: coeff = 10'sd114;
12'd435: coeff = 10'sd115;
12'd436: coeff = 10'sd116;
12'd437: coeff = 10'sd116;
12'd438: coeff = 10'sd117;
12'd439: coeff = 10'sd117;
12'd440: coeff = 10'sd118;
12'd441: coeff = 10'sd118;
12'd442: coeff = 10'sd119;
12'd443: coeff = 10'sd119;
12'd444: coeff = 10'sd120;
12'd445: coeff = 10'sd120;
12'd446: coeff = 10'sd121;
12'd447: coeff = 10'sd121;
12'd448: coeff = 10'sd122;
12'd449: coeff = 10'sd122;
12'd450: coeff = 10'sd123;
12'd451: coeff = 10'sd123;
12'd452: coeff = 10'sd124;
12'd453: coeff = 10'sd124;
12'd454: coeff = 10'sd125;
12'd455: coeff = 10'sd125;
12'd456: coeff = 10'sd126;
12'd457: coeff = 10'sd126;
12'd458: coeff = 10'sd127;
12'd459: coeff = 10'sd127;
```

12'd460: coeff = 10'sd128;
12'd461: coeff = 10'sd129;
12'd462: coeff = 10'sd129;
12'd463: coeff = 10'sd130;
12'd464: coeff = 10'sd130;
12'd465: coeff = 10'sd131;
12'd466: coeff = 10'sd131;
12'd467: coeff = 10'sd132;
12'd468: coeff = 10'sd132;
12'd469: coeff = 10'sd133;
12'd470: coeff = 10'sd133;
12'd471: coeff = 10'sd134;
12'd472: coeff = 10'sd134;
12'd473: coeff = 10'sd135;
12'd474: coeff = 10'sd136;
12'd475: coeff = 10'sd136;
12'd476: coeff = 10'sd137;
12'd477: coeff = 10'sd137;
12'd478: coeff = 10'sd138;
12'd479: coeff = 10'sd138;
12'd480: coeff = 10'sd139;
12'd481: coeff = 10'sd139;
12'd482: coeff = 10'sd140;
12'd483: coeff = 10'sd140;
12'd484: coeff = 10'sd141;
12'd485: coeff = 10'sd142;
12'd486: coeff = 10'sd142;
12'd487: coeff = 10'sd143;
12'd488: coeff = 10'sd143;
12'd489: coeff = 10'sd144;
12'd490: coeff = 10'sd144;
12'd491: coeff = 10'sd145;
12'd492: coeff = 10'sd146;
12'd493: coeff = 10'sd146;
12'd494: coeff = 10'sd147;
12'd495: coeff = 10'sd147;
12'd496: coeff = 10'sd148;
12'd497: coeff = 10'sd148;
12'd498: coeff = 10'sd149;
12'd499: coeff = 10'sd149;
12'd500: coeff = 10'sd150;
12'd501: coeff = 10'sd151;
12'd502: coeff = 10'sd151;
12'd503: coeff = 10'sd152;
12'd504: coeff = 10'sd152;
12'd505: coeff = 10'sd153;
12'd506: coeff = 10'sd153;
12'd507: coeff = 10'sd154;

12'd508: coeff = 10'sd155;
12'd509: coeff = 10'sd155;
12'd510: coeff = 10'sd156;
12'd511: coeff = 10'sd156;
12'd512: coeff = 10'sd157;
12'd513: coeff = 10'sd158;
12'd514: coeff = 10'sd158;
12'd515: coeff = 10'sd159;
12'd516: coeff = 10'sd159;
12'd517: coeff = 10'sd160;
12'd518: coeff = 10'sd160;
12'd519: coeff = 10'sd161;
12'd520: coeff = 10'sd162;
12'd521: coeff = 10'sd162;
12'd522: coeff = 10'sd163;
12'd523: coeff = 10'sd163;
12'd524: coeff = 10'sd164;
12'd525: coeff = 10'sd165;
12'd526: coeff = 10'sd165;
12'd527: coeff = 10'sd166;
12'd528: coeff = 10'sd166;
12'd529: coeff = 10'sd167;
12'd530: coeff = 10'sd167;
12'd531: coeff = 10'sd168;
12'd532: coeff = 10'sd169;
12'd533: coeff = 10'sd169;
12'd534: coeff = 10'sd170;
12'd535: coeff = 10'sd170;
12'd536: coeff = 10'sd171;
12'd537: coeff = 10'sd172;
12'd538: coeff = 10'sd172;
12'd539: coeff = 10'sd173;
12'd540: coeff = 10'sd173;
12'd541: coeff = 10'sd174;
12'd542: coeff = 10'sd175;
12'd543: coeff = 10'sd175;
12'd544: coeff = 10'sd176;
12'd545: coeff = 10'sd177;
12'd546: coeff = 10'sd177;
12'd547: coeff = 10'sd178;
12'd548: coeff = 10'sd178;
12'd549: coeff = 10'sd179;
12'd550: coeff = 10'sd180;
12'd551: coeff = 10'sd180;
12'd552: coeff = 10'sd181;
12'd553: coeff = 10'sd181;
12'd554: coeff = 10'sd182;
12'd555: coeff = 10'sd183;

```
12'd556: coeff = 10'sd183;
12'd557: coeff = 10'sd184;
12'd558: coeff = 10'sd184;
12'd559: coeff = 10'sd185;
12'd560: coeff = 10'sd186;
12'd561: coeff = 10'sd186;
12'd562: coeff = 10'sd187;
12'd563: coeff = 10'sd188;
12'd564: coeff = 10'sd188;
12'd565: coeff = 10'sd189;
12'd566: coeff = 10'sd189;
12'd567: coeff = 10'sd190;
12'd568: coeff = 10'sd191;
12'd569: coeff = 10'sd191;
12'd570: coeff = 10'sd192;
12'd571: coeff = 10'sd193;
12'd572: coeff = 10'sd193;
12'd573: coeff = 10'sd194;
12'd574: coeff = 10'sd194;
12'd575: coeff = 10'sd195;
12'd576: coeff = 10'sd196;
12'd577: coeff = 10'sd196;
12'd578: coeff = 10'sd197;
12'd579: coeff = 10'sd198;
12'd580: coeff = 10'sd198;
12'd581: coeff = 10'sd199;
12'd582: coeff = 10'sd200;
12'd583: coeff = 10'sd200;
12'd584: coeff = 10'sd201;
12'd585: coeff = 10'sd201;
12'd586: coeff = 10'sd202;
12'd587: coeff = 10'sd203;
12'd588: coeff = 10'sd203;
12'd589: coeff = 10'sd204;
12'd590: coeff = 10'sd205;
12'd591: coeff = 10'sd205;
12'd592: coeff = 10'sd206;
12'd593: coeff = 10'sd207;
12'd594: coeff = 10'sd207;
12'd595: coeff = 10'sd208;
12'd596: coeff = 10'sd209;
12'd597: coeff = 10'sd209;
12'd598: coeff = 10'sd210;
12'd599: coeff = 10'sd211;
12'd600: coeff = 10'sd211;
12'd601: coeff = 10'sd212;
12'd602: coeff = 10'sd212;
12'd603: coeff = 10'sd213;
```

12'd604: coeff = 10'sd214;
12'd605: coeff = 10'sd214;
12'd606: coeff = 10'sd215;
12'd607: coeff = 10'sd216;
12'd608: coeff = 10'sd216;
12'd609: coeff = 10'sd217;
12'd610: coeff = 10'sd218;
12'd611: coeff = 10'sd218;
12'd612: coeff = 10'sd219;
12'd613: coeff = 10'sd220;
12'd614: coeff = 10'sd220;
12'd615: coeff = 10'sd221;
12'd616: coeff = 10'sd222;
12'd617: coeff = 10'sd222;
12'd618: coeff = 10'sd223;
12'd619: coeff = 10'sd224;
12'd620: coeff = 10'sd224;
12'd621: coeff = 10'sd225;
12'd622: coeff = 10'sd226;
12'd623: coeff = 10'sd226;
12'd624: coeff = 10'sd227;
12'd625: coeff = 10'sd228;
12'd626: coeff = 10'sd228;
12'd627: coeff = 10'sd229;
12'd628: coeff = 10'sd230;
12'd629: coeff = 10'sd230;
12'd630: coeff = 10'sd231;
12'd631: coeff = 10'sd232;
12'd632: coeff = 10'sd232;
12'd633: coeff = 10'sd233;
12'd634: coeff = 10'sd234;
12'd635: coeff = 10'sd234;
12'd636: coeff = 10'sd235;
12'd637: coeff = 10'sd236;
12'd638: coeff = 10'sd236;
12'd639: coeff = 10'sd237;
12'd640: coeff = 10'sd238;
12'd641: coeff = 10'sd238;
12'd642: coeff = 10'sd239;
12'd643: coeff = 10'sd240;
12'd644: coeff = 10'sd240;
12'd645: coeff = 10'sd241;
12'd646: coeff = 10'sd242;
12'd647: coeff = 10'sd243;
12'd648: coeff = 10'sd243;
12'd649: coeff = 10'sd244;
12'd650: coeff = 10'sd245;
12'd651: coeff = 10'sd245;

12'd652: coeff = 10'sd246;
12'd653: coeff = 10'sd247;
12'd654: coeff = 10'sd247;
12'd655: coeff = 10'sd248;
12'd656: coeff = 10'sd249;
12'd657: coeff = 10'sd249;
12'd658: coeff = 10'sd250;
12'd659: coeff = 10'sd251;
12'd660: coeff = 10'sd251;
12'd661: coeff = 10'sd252;
12'd662: coeff = 10'sd253;
12'd663: coeff = 10'sd254;
12'd664: coeff = 10'sd254;
12'd665: coeff = 10'sd255;
12'd666: coeff = 10'sd256;
12'd667: coeff = 10'sd256;
12'd668: coeff = 10'sd257;
12'd669: coeff = 10'sd258;
12'd670: coeff = 10'sd258;
12'd671: coeff = 10'sd259;
12'd672: coeff = 10'sd260;
12'd673: coeff = 10'sd261;
12'd674: coeff = 10'sd261;
12'd675: coeff = 10'sd262;
12'd676: coeff = 10'sd263;
12'd677: coeff = 10'sd263;
12'd678: coeff = 10'sd264;
12'd679: coeff = 10'sd265;
12'd680: coeff = 10'sd265;
12'd681: coeff = 10'sd266;
12'd682: coeff = 10'sd267;
12'd683: coeff = 10'sd268;
12'd684: coeff = 10'sd268;
12'd685: coeff = 10'sd269;
12'd686: coeff = 10'sd270;
12'd687: coeff = 10'sd270;
12'd688: coeff = 10'sd271;
12'd689: coeff = 10'sd272;
12'd690: coeff = 10'sd273;
12'd691: coeff = 10'sd273;
12'd692: coeff = 10'sd274;
12'd693: coeff = 10'sd275;
12'd694: coeff = 10'sd275;
12'd695: coeff = 10'sd276;
12'd696: coeff = 10'sd277;
12'd697: coeff = 10'sd278;
12'd698: coeff = 10'sd278;
12'd699: coeff = 10'sd279;

```
12'd700: coeff = 10'sd280;
12'd701: coeff = 10'sd280;
12'd702: coeff = 10'sd281;
12'd703: coeff = 10'sd282;
12'd704: coeff = 10'sd283;
12'd705: coeff = 10'sd283;
12'd706: coeff = 10'sd284;
12'd707: coeff = 10'sd285;
12'd708: coeff = 10'sd285;
12'd709: coeff = 10'sd286;
12'd710: coeff = 10'sd287;
12'd711: coeff = 10'sd288;
12'd712: coeff = 10'sd288;
12'd713: coeff = 10'sd289;
12'd714: coeff = 10'sd290;
12'd715: coeff = 10'sd290;
12'd716: coeff = 10'sd291;
12'd717: coeff = 10'sd292;
12'd718: coeff = 10'sd293;
12'd719: coeff = 10'sd293;
12'd720: coeff = 10'sd294;
12'd721: coeff = 10'sd295;
12'd722: coeff = 10'sd296;
12'd723: coeff = 10'sd296;
12'd724: coeff = 10'sd297;
12'd725: coeff = 10'sd298;
12'd726: coeff = 10'sd299;
12'd727: coeff = 10'sd299;
12'd728: coeff = 10'sd300;
12'd729: coeff = 10'sd301;
12'd730: coeff = 10'sd301;
12'd731: coeff = 10'sd302;
12'd732: coeff = 10'sd303;
12'd733: coeff = 10'sd304;
12'd734: coeff = 10'sd304;
12'd735: coeff = 10'sd305;
12'd736: coeff = 10'sd306;
12'd737: coeff = 10'sd307;
12'd738: coeff = 10'sd307;
12'd739: coeff = 10'sd308;
12'd740: coeff = 10'sd309;
12'd741: coeff = 10'sd310;
12'd742: coeff = 10'sd310;
12'd743: coeff = 10'sd311;
12'd744: coeff = 10'sd312;
12'd745: coeff = 10'sd312;
12'd746: coeff = 10'sd313;
12'd747: coeff = 10'sd314;
```

12'd748: coeff = 10'sd315;
12'd749: coeff = 10'sd315;
12'd750: coeff = 10'sd316;
12'd751: coeff = 10'sd317;
12'd752: coeff = 10'sd318;
12'd753: coeff = 10'sd318;
12'd754: coeff = 10'sd319;
12'd755: coeff = 10'sd320;
12'd756: coeff = 10'sd321;
12'd757: coeff = 10'sd321;
12'd758: coeff = 10'sd322;
12'd759: coeff = 10'sd323;
12'd760: coeff = 10'sd324;
12'd761: coeff = 10'sd324;
12'd762: coeff = 10'sd325;
12'd763: coeff = 10'sd326;
12'd764: coeff = 10'sd327;
12'd765: coeff = 10'sd327;
12'd766: coeff = 10'sd328;
12'd767: coeff = 10'sd329;
12'd768: coeff = 10'sd330;
12'd769: coeff = 10'sd330;
12'd770: coeff = 10'sd331;
12'd771: coeff = 10'sd332;
12'd772: coeff = 10'sd333;
12'd773: coeff = 10'sd333;
12'd774: coeff = 10'sd334;
12'd775: coeff = 10'sd335;
12'd776: coeff = 10'sd336;
12'd777: coeff = 10'sd336;
12'd778: coeff = 10'sd337;
12'd779: coeff = 10'sd338;
12'd780: coeff = 10'sd339;
12'd781: coeff = 10'sd339;
12'd782: coeff = 10'sd340;
12'd783: coeff = 10'sd341;
12'd784: coeff = 10'sd342;
12'd785: coeff = 10'sd343;
12'd786: coeff = 10'sd343;
12'd787: coeff = 10'sd344;
12'd788: coeff = 10'sd345;
12'd789: coeff = 10'sd346;
12'd790: coeff = 10'sd346;
12'd791: coeff = 10'sd347;
12'd792: coeff = 10'sd348;
12'd793: coeff = 10'sd349;
12'd794: coeff = 10'sd349;
12'd795: coeff = 10'sd350;

```
12'd796: coeff = 10'sd351;
12'd797: coeff = 10'sd352;
12'd798: coeff = 10'sd352;
12'd799: coeff = 10'sd353;
12'd800: coeff = 10'sd354;
12'd801: coeff = 10'sd355;
12'd802: coeff = 10'sd355;
12'd803: coeff = 10'sd356;
12'd804: coeff = 10'sd357;
12'd805: coeff = 10'sd358;
12'd806: coeff = 10'sd359;
12'd807: coeff = 10'sd359;
12'd808: coeff = 10'sd360;
12'd809: coeff = 10'sd361;
12'd810: coeff = 10'sd362;
12'd811: coeff = 10'sd362;
12'd812: coeff = 10'sd363;
12'd813: coeff = 10'sd364;
12'd814: coeff = 10'sd365;
12'd815: coeff = 10'sd365;
12'd816: coeff = 10'sd366;
12'd817: coeff = 10'sd367;
12'd818: coeff = 10'sd368;
12'd819: coeff = 10'sd369;
12'd820: coeff = 10'sd369;
12'd821: coeff = 10'sd370;
12'd822: coeff = 10'sd371;
12'd823: coeff = 10'sd372;
12'd824: coeff = 10'sd372;
12'd825: coeff = 10'sd373;
12'd826: coeff = 10'sd374;
12'd827: coeff = 10'sd375;
12'd828: coeff = 10'sd376;
12'd829: coeff = 10'sd376;
12'd830: coeff = 10'sd377;
12'd831: coeff = 10'sd378;
12'd832: coeff = 10'sd379;
12'd833: coeff = 10'sd379;
12'd834: coeff = 10'sd380;
12'd835: coeff = 10'sd381;
12'd836: coeff = 10'sd382;
12'd837: coeff = 10'sd382;
12'd838: coeff = 10'sd383;
12'd839: coeff = 10'sd384;
12'd840: coeff = 10'sd385;
12'd841: coeff = 10'sd386;
12'd842: coeff = 10'sd386;
12'd843: coeff = 10'sd387;
```

```
12'd844: coeff = 10'sd388;
12'd845: coeff = 10'sd389;
12'd846: coeff = 10'sd390;
12'd847: coeff = 10'sd390;
12'd848: coeff = 10'sd391;
12'd849: coeff = 10'sd392;
12'd850: coeff = 10'sd393;
12'd851: coeff = 10'sd393;
12'd852: coeff = 10'sd394;
12'd853: coeff = 10'sd395;
12'd854: coeff = 10'sd396;
12'd855: coeff = 10'sd397;
12'd856: coeff = 10'sd397;
12'd857: coeff = 10'sd398;
12'd858: coeff = 10'sd399;
12'd859: coeff = 10'sd400;
12'd860: coeff = 10'sd400;
12'd861: coeff = 10'sd401;
12'd862: coeff = 10'sd402;
12'd863: coeff = 10'sd403;
12'd864: coeff = 10'sd404;
12'd865: coeff = 10'sd404;
12'd866: coeff = 10'sd405;
12'd867: coeff = 10'sd406;
12'd868: coeff = 10'sd407;
12'd869: coeff = 10'sd408;
12'd870: coeff = 10'sd408;
12'd871: coeff = 10'sd409;
12'd872: coeff = 10'sd410;
12'd873: coeff = 10'sd411;
12'd874: coeff = 10'sd411;
12'd875: coeff = 10'sd412;
12'd876: coeff = 10'sd413;
12'd877: coeff = 10'sd414;
12'd878: coeff = 10'sd415;
12'd879: coeff = 10'sd415;
12'd880: coeff = 10'sd416;
12'd881: coeff = 10'sd417;
12'd882: coeff = 10'sd418;
12'd883: coeff = 10'sd419;
12'd884: coeff = 10'sd419;
12'd885: coeff = 10'sd420;
12'd886: coeff = 10'sd421;
12'd887: coeff = 10'sd422;
12'd888: coeff = 10'sd423;
12'd889: coeff = 10'sd423;
12'd890: coeff = 10'sd424;
12'd891: coeff = 10'sd425;
```

```
12'd892: coeff = 10'sd426;
12'd893: coeff = 10'sd427;
12'd894: coeff = 10'sd427;
12'd895: coeff = 10'sd428;
12'd896: coeff = 10'sd429;
12'd897: coeff = 10'sd430;
12'd898: coeff = 10'sd430;
12'd899: coeff = 10'sd431;
12'd900: coeff = 10'sd432;
12'd901: coeff = 10'sd433;
12'd902: coeff = 10'sd434;
12'd903: coeff = 10'sd434;
12'd904: coeff = 10'sd435;
12'd905: coeff = 10'sd436;
12'd906: coeff = 10'sd437;
12'd907: coeff = 10'sd438;
12'd908: coeff = 10'sd438;
12'd909: coeff = 10'sd439;
12'd910: coeff = 10'sd440;
12'd911: coeff = 10'sd441;
12'd912: coeff = 10'sd442;
12'd913: coeff = 10'sd442;
12'd914: coeff = 10'sd443;
12'd915: coeff = 10'sd444;
12'd916: coeff = 10'sd445;
12'd917: coeff = 10'sd446;
12'd918: coeff = 10'sd446;
12'd919: coeff = 10'sd447;
12'd920: coeff = 10'sd448;
12'd921: coeff = 10'sd449;
12'd922: coeff = 10'sd450;
12'd923: coeff = 10'sd450;
12'd924: coeff = 10'sd451;
12'd925: coeff = 10'sd452;
12'd926: coeff = 10'sd453;
12'd927: coeff = 10'sd454;
12'd928: coeff = 10'sd454;
12'd929: coeff = 10'sd455;
12'd930: coeff = 10'sd456;
12'd931: coeff = 10'sd457;
12'd932: coeff = 10'sd458;
12'd933: coeff = 10'sd458;
12'd934: coeff = 10'sd459;
12'd935: coeff = 10'sd460;
12'd936: coeff = 10'sd461;
12'd937: coeff = 10'sd462;
12'd938: coeff = 10'sd462;
12'd939: coeff = 10'sd463;
```

```
12'd940: coeff = 10'sd464;
12'd941: coeff = 10'sd465;
12'd942: coeff = 10'sd466;
12'd943: coeff = 10'sd466;
12'd944: coeff = 10'sd467;
12'd945: coeff = 10'sd468;
12'd946: coeff = 10'sd469;
12'd947: coeff = 10'sd470;
12'd948: coeff = 10'sd470;
12'd949: coeff = 10'sd471;
12'd950: coeff = 10'sd472;
12'd951: coeff = 10'sd473;
12'd952: coeff = 10'sd474;
12'd953: coeff = 10'sd474;
12'd954: coeff = 10'sd475;
12'd955: coeff = 10'sd476;
12'd956: coeff = 10'sd477;
12'd957: coeff = 10'sd478;
12'd958: coeff = 10'sd478;
12'd959: coeff = 10'sd479;
12'd960: coeff = 10'sd480;
12'd961: coeff = 10'sd481;
12'd962: coeff = 10'sd482;
12'd963: coeff = 10'sd482;
12'd964: coeff = 10'sd483;
12'd965: coeff = 10'sd484;
12'd966: coeff = 10'sd485;
12'd967: coeff = 10'sd486;
12'd968: coeff = 10'sd486;
12'd969: coeff = 10'sd487;
12'd970: coeff = 10'sd488;
12'd971: coeff = 10'sd489;
12'd972: coeff = 10'sd490;
12'd973: coeff = 10'sd490;
12'd974: coeff = 10'sd491;
12'd975: coeff = 10'sd492;
12'd976: coeff = 10'sd493;
12'd977: coeff = 10'sd494;
12'd978: coeff = 10'sd495;
12'd979: coeff = 10'sd495;
12'd980: coeff = 10'sd496;
12'd981: coeff = 10'sd497;
12'd982: coeff = 10'sd498;
12'd983: coeff = 10'sd499;
12'd984: coeff = 10'sd499;
12'd985: coeff = 10'sd500;
12'd986: coeff = 10'sd501;
12'd987: coeff = 10'sd502;
```

```
12'd988: coeff = 10'sd503;
12'd989: coeff = 10'sd503;
12'd990: coeff = 10'sd504;
12'd991: coeff = 10'sd505;
12'd992: coeff = 10'sd506;
12'd993: coeff = 10'sd507;
12'd994: coeff = 10'sd507;
12'd995: coeff = 10'sd508;
12'd996: coeff = 10'sd509;
12'd997: coeff = 10'sd510;
12'd998: coeff = 10'sd511;
12'd999: coeff = 10'sd511;
12'd1000: coeff = 10'sd512;
12'd1001: coeff = 10'sd513;
12'd1002: coeff = 10'sd514;
12'd1003: coeff = 10'sd515;
12'd1004: coeff = 10'sd515;
12'd1005: coeff = 10'sd516;
12'd1006: coeff = 10'sd517;
12'd1007: coeff = 10'sd518;
12'd1008: coeff = 10'sd519;
12'd1009: coeff = 10'sd519;
12'd1010: coeff = 10'sd520;
12'd1011: coeff = 10'sd521;
12'd1012: coeff = 10'sd522;
12'd1013: coeff = 10'sd523;
12'd1014: coeff = 10'sd523;
12'd1015: coeff = 10'sd524;
12'd1016: coeff = 10'sd525;
12'd1017: coeff = 10'sd526;
12'd1018: coeff = 10'sd527;
12'd1019: coeff = 10'sd527;
12'd1020: coeff = 10'sd528;
12'd1021: coeff = 10'sd529;
12'd1022: coeff = 10'sd530;
12'd1023: coeff = 10'sd531;
12'd1024: coeff = 10'sd532;
12'd1025: coeff = 10'sd532;
12'd1026: coeff = 10'sd533;
12'd1027: coeff = 10'sd534;
12'd1028: coeff = 10'sd535;
12'd1029: coeff = 10'sd536;
12'd1030: coeff = 10'sd536;
12'd1031: coeff = 10'sd537;
12'd1032: coeff = 10'sd538;
12'd1033: coeff = 10'sd539;
12'd1034: coeff = 10'sd540;
12'd1035: coeff = 10'sd540;
```

```
12'd1036: coeff = 10'sd541;
12'd1037: coeff = 10'sd542;
12'd1038: coeff = 10'sd543;
12'd1039: coeff = 10'sd544;
12'd1040: coeff = 10'sd544;
12'd1041: coeff = 10'sd545;
12'd1042: coeff = 10'sd546;
12'd1043: coeff = 10'sd547;
12'd1044: coeff = 10'sd548;
12'd1045: coeff = 10'sd548;
12'd1046: coeff = 10'sd549;
12'd1047: coeff = 10'sd550;
12'd1048: coeff = 10'sd551;
12'd1049: coeff = 10'sd552;
12'd1050: coeff = 10'sd552;
12'd1051: coeff = 10'sd553;
12'd1052: coeff = 10'sd554;
12'd1053: coeff = 10'sd555;
12'd1054: coeff = 10'sd556;
12'd1055: coeff = 10'sd556;
12'd1056: coeff = 10'sd557;
12'd1057: coeff = 10'sd558;
12'd1058: coeff = 10'sd559;
12'd1059: coeff = 10'sd560;
12'd1060: coeff = 10'sd560;
12'd1061: coeff = 10'sd561;
12'd1062: coeff = 10'sd562;
12'd1063: coeff = 10'sd563;
12'd1064: coeff = 10'sd564;
12'd1065: coeff = 10'sd564;
12'd1066: coeff = 10'sd565;
12'd1067: coeff = 10'sd566;
12'd1068: coeff = 10'sd567;
12'd1069: coeff = 10'sd568;
12'd1070: coeff = 10'sd568;
12'd1071: coeff = 10'sd569;
12'd1072: coeff = 10'sd570;
12'd1073: coeff = 10'sd571;
12'd1074: coeff = 10'sd572;
12'd1075: coeff = 10'sd572;
12'd1076: coeff = 10'sd573;
12'd1077: coeff = 10'sd574;
12'd1078: coeff = 10'sd575;
12'd1079: coeff = 10'sd576;
12'd1080: coeff = 10'sd576;
12'd1081: coeff = 10'sd577;
12'd1082: coeff = 10'sd578;
12'd1083: coeff = 10'sd579;
```

```
12'd1084: coeff = 10'sd580;
12'd1085: coeff = 10'sd580;
12'd1086: coeff = 10'sd581;
12'd1087: coeff = 10'sd582;
12'd1088: coeff = 10'sd583;
12'd1089: coeff = 10'sd584;
12'd1090: coeff = 10'sd584;
12'd1091: coeff = 10'sd585;
12'd1092: coeff = 10'sd586;
12'd1093: coeff = 10'sd587;
12'd1094: coeff = 10'sd588;
12'd1095: coeff = 10'sd588;
12'd1096: coeff = 10'sd589;
12'd1097: coeff = 10'sd590;
12'd1098: coeff = 10'sd591;
12'd1099: coeff = 10'sd592;
12'd1100: coeff = 10'sd592;
12'd1101: coeff = 10'sd593;
12'd1102: coeff = 10'sd594;
12'd1103: coeff = 10'sd595;
12'd1104: coeff = 10'sd595;
12'd1105: coeff = 10'sd596;
12'd1106: coeff = 10'sd597;
12'd1107: coeff = 10'sd598;
12'd1108: coeff = 10'sd599;
12'd1109: coeff = 10'sd599;
12'd1110: coeff = 10'sd600;
12'd1111: coeff = 10'sd601;
12'd1112: coeff = 10'sd602;
12'd1113: coeff = 10'sd603;
12'd1114: coeff = 10'sd603;
12'd1115: coeff = 10'sd604;
12'd1116: coeff = 10'sd605;
12'd1117: coeff = 10'sd606;
12'd1118: coeff = 10'sd607;
12'd1119: coeff = 10'sd607;
12'd1120: coeff = 10'sd608;
12'd1121: coeff = 10'sd609;
12'd1122: coeff = 10'sd610;
12'd1123: coeff = 10'sd611;
12'd1124: coeff = 10'sd611;
12'd1125: coeff = 10'sd612;
12'd1126: coeff = 10'sd613;
12'd1127: coeff = 10'sd614;
12'd1128: coeff = 10'sd614;
12'd1129: coeff = 10'sd615;
12'd1130: coeff = 10'sd616;
12'd1131: coeff = 10'sd617;
```

```
12'd1132: coeff = 10'sd618;
12'd1133: coeff = 10'sd618;
12'd1134: coeff = 10'sd619;
12'd1135: coeff = 10'sd620;
12'd1136: coeff = 10'sd621;
12'd1137: coeff = 10'sd622;
12'd1138: coeff = 10'sd622;
12'd1139: coeff = 10'sd623;
12'd1140: coeff = 10'sd624;
12'd1141: coeff = 10'sd625;
12'd1142: coeff = 10'sd625;
12'd1143: coeff = 10'sd626;
12'd1144: coeff = 10'sd627;
12'd1145: coeff = 10'sd628;
12'd1146: coeff = 10'sd629;
12'd1147: coeff = 10'sd629;
12'd1148: coeff = 10'sd630;
12'd1149: coeff = 10'sd631;
12'd1150: coeff = 10'sd632;
12'd1151: coeff = 10'sd633;
12'd1152: coeff = 10'sd633;
12'd1153: coeff = 10'sd634;
12'd1154: coeff = 10'sd635;
12'd1155: coeff = 10'sd636;
12'd1156: coeff = 10'sd636;
12'd1157: coeff = 10'sd637;
12'd1158: coeff = 10'sd638;
12'd1159: coeff = 10'sd639;
12'd1160: coeff = 10'sd640;
12'd1161: coeff = 10'sd640;
12'd1162: coeff = 10'sd641;
12'd1163: coeff = 10'sd642;
12'd1164: coeff = 10'sd643;
12'd1165: coeff = 10'sd643;
12'd1166: coeff = 10'sd644;
12'd1167: coeff = 10'sd645;
12'd1168: coeff = 10'sd646;
12'd1169: coeff = 10'sd647;
12'd1170: coeff = 10'sd647;
12'd1171: coeff = 10'sd648;
12'd1172: coeff = 10'sd649;
12'd1173: coeff = 10'sd650;
12'd1174: coeff = 10'sd650;
12'd1175: coeff = 10'sd651;
12'd1176: coeff = 10'sd652;
12'd1177: coeff = 10'sd653;
12'd1178: coeff = 10'sd654;
12'd1179: coeff = 10'sd654;
```

```
12'd1180: coeff = 10'sd655;
12'd1181: coeff = 10'sd656;
12'd1182: coeff = 10'sd657;
12'd1183: coeff = 10'sd657;
12'd1184: coeff = 10'sd658;
12'd1185: coeff = 10'sd659;
12'd1186: coeff = 10'sd660;
12'd1187: coeff = 10'sd660;
12'd1188: coeff = 10'sd661;
12'd1189: coeff = 10'sd662;
12'd1190: coeff = 10'sd663;
12'd1191: coeff = 10'sd664;
12'd1192: coeff = 10'sd664;
12'd1193: coeff = 10'sd665;
12'd1194: coeff = 10'sd666;
12'd1195: coeff = 10'sd667;
12'd1196: coeff = 10'sd667;
12'd1197: coeff = 10'sd668;
12'd1198: coeff = 10'sd669;
12'd1199: coeff = 10'sd670;
12'd1200: coeff = 10'sd670;
12'd1201: coeff = 10'sd671;
12'd1202: coeff = 10'sd672;
12'd1203: coeff = 10'sd673;
12'd1204: coeff = 10'sd674;
12'd1205: coeff = 10'sd674;
12'd1206: coeff = 10'sd675;
12'd1207: coeff = 10'sd676;
12'd1208: coeff = 10'sd677;
12'd1209: coeff = 10'sd677;
12'd1210: coeff = 10'sd678;
12'd1211: coeff = 10'sd679;
12'd1212: coeff = 10'sd680;
12'd1213: coeff = 10'sd680;
12'd1214: coeff = 10'sd681;
12'd1215: coeff = 10'sd682;
12'd1216: coeff = 10'sd683;
12'd1217: coeff = 10'sd683;
12'd1218: coeff = 10'sd684;
12'd1219: coeff = 10'sd685;
12'd1220: coeff = 10'sd686;
12'd1221: coeff = 10'sd686;
12'd1222: coeff = 10'sd687;
12'd1223: coeff = 10'sd688;
12'd1224: coeff = 10'sd689;
12'd1225: coeff = 10'sd689;
12'd1226: coeff = 10'sd690;
12'd1227: coeff = 10'sd691;
```

12'd1228: coeff = 10'sd692;
12'd1229: coeff = 10'sd692;
12'd1230: coeff = 10'sd693;
12'd1231: coeff = 10'sd694;
12'd1232: coeff = 10'sd695;
12'd1233: coeff = 10'sd695;
12'd1234: coeff = 10'sd696;
12'd1235: coeff = 10'sd697;
12'd1236: coeff = 10'sd698;
12'd1237: coeff = 10'sd698;
12'd1238: coeff = 10'sd699;
12'd1239: coeff = 10'sd700;
12'd1240: coeff = 10'sd701;
12'd1241: coeff = 10'sd701;
12'd1242: coeff = 10'sd702;
12'd1243: coeff = 10'sd703;
12'd1244: coeff = 10'sd704;
12'd1245: coeff = 10'sd704;
12'd1246: coeff = 10'sd705;
12'd1247: coeff = 10'sd706;
12'd1248: coeff = 10'sd707;
12'd1249: coeff = 10'sd707;
12'd1250: coeff = 10'sd708;
12'd1251: coeff = 10'sd709;
12'd1252: coeff = 10'sd710;
12'd1253: coeff = 10'sd710;
12'd1254: coeff = 10'sd711;
12'd1255: coeff = 10'sd712;
12'd1256: coeff = 10'sd713;
12'd1257: coeff = 10'sd713;
12'd1258: coeff = 10'sd714;
12'd1259: coeff = 10'sd715;
12'd1260: coeff = 10'sd716;
12'd1261: coeff = 10'sd716;
12'd1262: coeff = 10'sd717;
12'd1263: coeff = 10'sd718;
12'd1264: coeff = 10'sd719;
12'd1265: coeff = 10'sd719;
12'd1266: coeff = 10'sd720;
12'd1267: coeff = 10'sd721;
12'd1268: coeff = 10'sd721;
12'd1269: coeff = 10'sd722;
12'd1270: coeff = 10'sd723;
12'd1271: coeff = 10'sd724;
12'd1272: coeff = 10'sd724;
12'd1273: coeff = 10'sd725;
12'd1274: coeff = 10'sd726;
12'd1275: coeff = 10'sd727;

```
12'd1276: coeff = 10'sd727;
12'd1277: coeff = 10'sd728;
12'd1278: coeff = 10'sd729;
12'd1279: coeff = 10'sd730;
12'd1280: coeff = 10'sd730;
12'd1281: coeff = 10'sd731;
12'd1282: coeff = 10'sd732;
12'd1283: coeff = 10'sd732;
12'd1284: coeff = 10'sd733;
12'd1285: coeff = 10'sd734;
12'd1286: coeff = 10'sd735;
12'd1287: coeff = 10'sd735;
12'd1288: coeff = 10'sd736;
12'd1289: coeff = 10'sd737;
12'd1290: coeff = 10'sd737;
12'd1291: coeff = 10'sd738;
12'd1292: coeff = 10'sd739;
12'd1293: coeff = 10'sd740;
12'd1294: coeff = 10'sd740;
12'd1295: coeff = 10'sd741;
12'd1296: coeff = 10'sd742;
12'd1297: coeff = 10'sd743;
12'd1298: coeff = 10'sd743;
12'd1299: coeff = 10'sd744;
12'd1300: coeff = 10'sd745;
12'd1301: coeff = 10'sd745;
12'd1302: coeff = 10'sd746;
12'd1303: coeff = 10'sd747;
12'd1304: coeff = 10'sd748;
12'd1305: coeff = 10'sd748;
12'd1306: coeff = 10'sd749;
12'd1307: coeff = 10'sd750;
12'd1308: coeff = 10'sd750;
12'd1309: coeff = 10'sd751;
12'd1310: coeff = 10'sd752;
12'd1311: coeff = 10'sd753;
12'd1312: coeff = 10'sd753;
12'd1313: coeff = 10'sd754;
12'd1314: coeff = 10'sd755;
12'd1315: coeff = 10'sd755;
12'd1316: coeff = 10'sd756;
12'd1317: coeff = 10'sd757;
12'd1318: coeff = 10'sd757;
12'd1319: coeff = 10'sd758;
12'd1320: coeff = 10'sd759;
12'd1321: coeff = 10'sd760;
12'd1322: coeff = 10'sd760;
12'd1323: coeff = 10'sd761;
```

12'd1324: coeff = 10'sd762;
12'd1325: coeff = 10'sd762;
12'd1326: coeff = 10'sd763;
12'd1327: coeff = 10'sd764;
12'd1328: coeff = 10'sd765;
12'd1329: coeff = 10'sd765;
12'd1330: coeff = 10'sd766;
12'd1331: coeff = 10'sd767;
12'd1332: coeff = 10'sd767;
12'd1333: coeff = 10'sd768;
12'd1334: coeff = 10'sd769;
12'd1335: coeff = 10'sd769;
12'd1336: coeff = 10'sd770;
12'd1337: coeff = 10'sd771;
12'd1338: coeff = 10'sd771;
12'd1339: coeff = 10'sd772;
12'd1340: coeff = 10'sd773;
12'd1341: coeff = 10'sd774;
12'd1342: coeff = 10'sd774;
12'd1343: coeff = 10'sd775;
12'd1344: coeff = 10'sd776;
12'd1345: coeff = 10'sd776;
12'd1346: coeff = 10'sd777;
12'd1347: coeff = 10'sd778;
12'd1348: coeff = 10'sd778;
12'd1349: coeff = 10'sd779;
12'd1350: coeff = 10'sd780;
12'd1351: coeff = 10'sd780;
12'd1352: coeff = 10'sd781;
12'd1353: coeff = 10'sd782;
12'd1354: coeff = 10'sd782;
12'd1355: coeff = 10'sd783;
12'd1356: coeff = 10'sd784;
12'd1357: coeff = 10'sd785;
12'd1358: coeff = 10'sd785;
12'd1359: coeff = 10'sd786;
12'd1360: coeff = 10'sd787;
12'd1361: coeff = 10'sd787;
12'd1362: coeff = 10'sd788;
12'd1363: coeff = 10'sd789;
12'd1364: coeff = 10'sd789;
12'd1365: coeff = 10'sd790;
12'd1366: coeff = 10'sd791;
12'd1367: coeff = 10'sd791;
12'd1368: coeff = 10'sd792;
12'd1369: coeff = 10'sd793;
12'd1370: coeff = 10'sd793;
12'd1371: coeff = 10'sd794;

```
12'd1372: coeff = 10'sd795;
12'd1373: coeff = 10'sd795;
12'd1374: coeff = 10'sd796;
12'd1375: coeff = 10'sd797;
12'd1376: coeff = 10'sd797;
12'd1377: coeff = 10'sd798;
12'd1378: coeff = 10'sd799;
12'd1379: coeff = 10'sd799;
12'd1380: coeff = 10'sd800;
12'd1381: coeff = 10'sd801;
12'd1382: coeff = 10'sd801;
12'd1383: coeff = 10'sd802;
12'd1384: coeff = 10'sd803;
12'd1385: coeff = 10'sd803;
12'd1386: coeff = 10'sd804;
12'd1387: coeff = 10'sd805;
12'd1388: coeff = 10'sd805;
12'd1389: coeff = 10'sd806;
12'd1390: coeff = 10'sd807;
12'd1391: coeff = 10'sd807;
12'd1392: coeff = 10'sd808;
12'd1393: coeff = 10'sd809;
12'd1394: coeff = 10'sd809;
12'd1395: coeff = 10'sd810;
12'd1396: coeff = 10'sd811;
12'd1397: coeff = 10'sd811;
12'd1398: coeff = 10'sd812;
12'd1399: coeff = 10'sd813;
12'd1400: coeff = 10'sd813;
12'd1401: coeff = 10'sd814;
12'd1402: coeff = 10'sd814;
12'd1403: coeff = 10'sd815;
12'd1404: coeff = 10'sd816;
12'd1405: coeff = 10'sd816;
12'd1406: coeff = 10'sd817;
12'd1407: coeff = 10'sd818;
12'd1408: coeff = 10'sd818;
12'd1409: coeff = 10'sd819;
12'd1410: coeff = 10'sd820;
12'd1411: coeff = 10'sd820;
12'd1412: coeff = 10'sd821;
12'd1413: coeff = 10'sd822;
12'd1414: coeff = 10'sd822;
12'd1415: coeff = 10'sd823;
12'd1416: coeff = 10'sd823;
12'd1417: coeff = 10'sd824;
12'd1418: coeff = 10'sd825;
12'd1419: coeff = 10'sd825;
```

```
12'd1420: coeff = 10'sd826;
12'd1421: coeff = 10'sd827;
12'd1422: coeff = 10'sd827;
12'd1423: coeff = 10'sd828;
12'd1424: coeff = 10'sd829;
12'd1425: coeff = 10'sd829;
12'd1426: coeff = 10'sd830;
12'd1427: coeff = 10'sd830;
12'd1428: coeff = 10'sd831;
12'd1429: coeff = 10'sd832;
12'd1430: coeff = 10'sd832;
12'd1431: coeff = 10'sd833;
12'd1432: coeff = 10'sd834;
12'd1433: coeff = 10'sd834;
12'd1434: coeff = 10'sd835;
12'd1435: coeff = 10'sd835;
12'd1436: coeff = 10'sd836;
12'd1437: coeff = 10'sd837;
12'd1438: coeff = 10'sd837;
12'd1439: coeff = 10'sd838;
12'd1440: coeff = 10'sd839;
12'd1441: coeff = 10'sd839;
12'd1442: coeff = 10'sd840;
12'd1443: coeff = 10'sd840;
12'd1444: coeff = 10'sd841;
12'd1445: coeff = 10'sd842;
12'd1446: coeff = 10'sd842;
12'd1447: coeff = 10'sd843;
12'd1448: coeff = 10'sd844;
12'd1449: coeff = 10'sd844;
12'd1450: coeff = 10'sd845;
12'd1451: coeff = 10'sd845;
12'd1452: coeff = 10'sd846;
12'd1453: coeff = 10'sd847;
12'd1454: coeff = 10'sd847;
12'd1455: coeff = 10'sd848;
12'd1456: coeff = 10'sd848;
12'd1457: coeff = 10'sd849;
12'd1458: coeff = 10'sd850;
12'd1459: coeff = 10'sd850;
12'd1460: coeff = 10'sd851;
12'd1461: coeff = 10'sd851;
12'd1462: coeff = 10'sd852;
12'd1463: coeff = 10'sd853;
12'd1464: coeff = 10'sd853;
12'd1465: coeff = 10'sd854;
12'd1466: coeff = 10'sd854;
12'd1467: coeff = 10'sd855;
```

```
12'd1468: coeff = 10'sd856;
12'd1469: coeff = 10'sd856;
12'd1470: coeff = 10'sd857;
12'd1471: coeff = 10'sd857;
12'd1472: coeff = 10'sd858;
12'd1473: coeff = 10'sd859;
12'd1474: coeff = 10'sd859;
12'd1475: coeff = 10'sd860;
12'd1476: coeff = 10'sd860;
12'd1477: coeff = 10'sd861;
12'd1478: coeff = 10'sd862;
12'd1479: coeff = 10'sd862;
12'd1480: coeff = 10'sd863;
12'd1481: coeff = 10'sd863;
12'd1482: coeff = 10'sd864;
12'd1483: coeff = 10'sd864;
12'd1484: coeff = 10'sd865;
12'd1485: coeff = 10'sd866;
12'd1486: coeff = 10'sd866;
12'd1487: coeff = 10'sd867;
12'd1488: coeff = 10'sd867;
12'd1489: coeff = 10'sd868;
12'd1490: coeff = 10'sd869;
12'd1491: coeff = 10'sd869;
12'd1492: coeff = 10'sd870;
12'd1493: coeff = 10'sd870;
12'd1494: coeff = 10'sd871;
12'd1495: coeff = 10'sd871;
12'd1496: coeff = 10'sd872;
12'd1497: coeff = 10'sd873;
12'd1498: coeff = 10'sd873;
12'd1499: coeff = 10'sd874;
12'd1500: coeff = 10'sd874;
12'd1501: coeff = 10'sd875;
12'd1502: coeff = 10'sd875;
12'd1503: coeff = 10'sd876;
12'd1504: coeff = 10'sd877;
12'd1505: coeff = 10'sd877;
12'd1506: coeff = 10'sd878;
12'd1507: coeff = 10'sd878;
12'd1508: coeff = 10'sd879;
12'd1509: coeff = 10'sd879;
12'd1510: coeff = 10'sd880;
12'd1511: coeff = 10'sd880;
12'd1512: coeff = 10'sd881;
12'd1513: coeff = 10'sd882;
12'd1514: coeff = 10'sd882;
12'd1515: coeff = 10'sd883;
```

12'd1516: coeff = 10'sd883;
12'd1517: coeff = 10'sd884;
12'd1518: coeff = 10'sd884;
12'd1519: coeff = 10'sd885;
12'd1520: coeff = 10'sd885;
12'd1521: coeff = 10'sd886;
12'd1522: coeff = 10'sd887;
12'd1523: coeff = 10'sd887;
12'd1524: coeff = 10'sd888;
12'd1525: coeff = 10'sd888;
12'd1526: coeff = 10'sd889;
12'd1527: coeff = 10'sd889;
12'd1528: coeff = 10'sd890;
12'd1529: coeff = 10'sd890;
12'd1530: coeff = 10'sd891;
12'd1531: coeff = 10'sd891;
12'd1532: coeff = 10'sd892;
12'd1533: coeff = 10'sd893;
12'd1534: coeff = 10'sd893;
12'd1535: coeff = 10'sd894;
12'd1536: coeff = 10'sd894;
12'd1537: coeff = 10'sd895;
12'd1538: coeff = 10'sd895;
12'd1539: coeff = 10'sd896;
12'd1540: coeff = 10'sd896;
12'd1541: coeff = 10'sd897;
12'd1542: coeff = 10'sd897;
12'd1543: coeff = 10'sd898;
12'd1544: coeff = 10'sd898;
12'd1545: coeff = 10'sd899;
12'd1546: coeff = 10'sd899;
12'd1547: coeff = 10'sd900;
12'd1548: coeff = 10'sd900;
12'd1549: coeff = 10'sd901;
12'd1550: coeff = 10'sd902;
12'd1551: coeff = 10'sd902;
12'd1552: coeff = 10'sd903;
12'd1553: coeff = 10'sd903;
12'd1554: coeff = 10'sd904;
12'd1555: coeff = 10'sd904;
12'd1556: coeff = 10'sd905;
12'd1557: coeff = 10'sd905;
12'd1558: coeff = 10'sd906;
12'd1559: coeff = 10'sd906;
12'd1560: coeff = 10'sd907;
12'd1561: coeff = 10'sd907;
12'd1562: coeff = 10'sd908;
12'd1563: coeff = 10'sd908;

12'd1564: coeff = 10'sd909;
12'd1565: coeff = 10'sd909;
12'd1566: coeff = 10'sd910;
12'd1567: coeff = 10'sd910;
12'd1568: coeff = 10'sd911;
12'd1569: coeff = 10'sd911;
12'd1570: coeff = 10'sd912;
12'd1571: coeff = 10'sd912;
12'd1572: coeff = 10'sd913;
12'd1573: coeff = 10'sd913;
12'd1574: coeff = 10'sd914;
12'd1575: coeff = 10'sd914;
12'd1576: coeff = 10'sd915;
12'd1577: coeff = 10'sd915;
12'd1578: coeff = 10'sd916;
12'd1579: coeff = 10'sd916;
12'd1580: coeff = 10'sd917;
12'd1581: coeff = 10'sd917;
12'd1582: coeff = 10'sd918;
12'd1583: coeff = 10'sd918;
12'd1584: coeff = 10'sd919;
12'd1585: coeff = 10'sd919;
12'd1586: coeff = 10'sd920;
12'd1587: coeff = 10'sd920;
12'd1588: coeff = 10'sd921;
12'd1589: coeff = 10'sd921;
12'd1590: coeff = 10'sd922;
12'd1591: coeff = 10'sd922;
12'd1592: coeff = 10'sd923;
12'd1593: coeff = 10'sd923;
12'd1594: coeff = 10'sd924;
12'd1595: coeff = 10'sd924;
12'd1596: coeff = 10'sd925;
12'd1597: coeff = 10'sd925;
12'd1598: coeff = 10'sd925;
12'd1599: coeff = 10'sd926;
12'd1600: coeff = 10'sd926;
12'd1601: coeff = 10'sd927;
12'd1602: coeff = 10'sd927;
12'd1603: coeff = 10'sd928;
12'd1604: coeff = 10'sd928;
12'd1605: coeff = 10'sd929;
12'd1606: coeff = 10'sd929;
12'd1607: coeff = 10'sd930;
12'd1608: coeff = 10'sd930;
12'd1609: coeff = 10'sd931;
12'd1610: coeff = 10'sd931;
12'd1611: coeff = 10'sd932;

```
12'd1612: coeff = 10'sd932;
12'd1613: coeff = 10'sd932;
12'd1614: coeff = 10'sd933;
12'd1615: coeff = 10'sd933;
12'd1616: coeff = 10'sd934;
12'd1617: coeff = 10'sd934;
12'd1618: coeff = 10'sd935;
12'd1619: coeff = 10'sd935;
12'd1620: coeff = 10'sd936;
12'd1621: coeff = 10'sd936;
12'd1622: coeff = 10'sd937;
12'd1623: coeff = 10'sd937;
12'd1624: coeff = 10'sd937;
12'd1625: coeff = 10'sd938;
12'd1626: coeff = 10'sd938;
12'd1627: coeff = 10'sd939;
12'd1628: coeff = 10'sd939;
12'd1629: coeff = 10'sd940;
12'd1630: coeff = 10'sd940;
12'd1631: coeff = 10'sd941;
12'd1632: coeff = 10'sd941;
12'd1633: coeff = 10'sd941;
12'd1634: coeff = 10'sd942;
12'd1635: coeff = 10'sd942;
12'd1636: coeff = 10'sd943;
12'd1637: coeff = 10'sd943;
12'd1638: coeff = 10'sd944;
12'd1639: coeff = 10'sd944;
12'd1640: coeff = 10'sd944;
12'd1641: coeff = 10'sd945;
12'd1642: coeff = 10'sd945;
12'd1643: coeff = 10'sd946;
12'd1644: coeff = 10'sd946;
12'd1645: coeff = 10'sd947;
12'd1646: coeff = 10'sd947;
12'd1647: coeff = 10'sd947;
12'd1648: coeff = 10'sd948;
12'd1649: coeff = 10'sd948;
12'd1650: coeff = 10'sd949;
12'd1651: coeff = 10'sd949;
12'd1652: coeff = 10'sd950;
12'd1653: coeff = 10'sd950;
12'd1654: coeff = 10'sd950;
12'd1655: coeff = 10'sd951;
12'd1656: coeff = 10'sd951;
12'd1657: coeff = 10'sd952;
12'd1658: coeff = 10'sd952;
12'd1659: coeff = 10'sd952;
```

```
12'd1660: coeff = 10'sd953;
12'd1661: coeff = 10'sd953;
12'd1662: coeff = 10'sd954;
12'd1663: coeff = 10'sd954;
12'd1664: coeff = 10'sd954;
12'd1665: coeff = 10'sd955;
12'd1666: coeff = 10'sd955;
12'd1667: coeff = 10'sd956;
12'd1668: coeff = 10'sd956;
12'd1669: coeff = 10'sd957;
12'd1670: coeff = 10'sd957;
12'd1671: coeff = 10'sd957;
12'd1672: coeff = 10'sd958;
12'd1673: coeff = 10'sd958;
12'd1674: coeff = 10'sd958;
12'd1675: coeff = 10'sd959;
12'd1676: coeff = 10'sd959;
12'd1677: coeff = 10'sd960;
12'd1678: coeff = 10'sd960;
12'd1679: coeff = 10'sd960;
12'd1680: coeff = 10'sd961;
12'd1681: coeff = 10'sd961;
12'd1682: coeff = 10'sd962;
12'd1683: coeff = 10'sd962;
12'd1684: coeff = 10'sd962;
12'd1685: coeff = 10'sd963;
12'd1686: coeff = 10'sd963;
12'd1687: coeff = 10'sd964;
12'd1688: coeff = 10'sd964;
12'd1689: coeff = 10'sd964;
12'd1690: coeff = 10'sd965;
12'd1691: coeff = 10'sd965;
12'd1692: coeff = 10'sd965;
12'd1693: coeff = 10'sd966;
12'd1694: coeff = 10'sd966;
12'd1695: coeff = 10'sd967;
12'd1696: coeff = 10'sd967;
12'd1697: coeff = 10'sd967;
12'd1698: coeff = 10'sd968;
12'd1699: coeff = 10'sd968;
12'd1700: coeff = 10'sd968;
12'd1701: coeff = 10'sd969;
12'd1702: coeff = 10'sd969;
12'd1703: coeff = 10'sd969;
12'd1704: coeff = 10'sd970;
12'd1705: coeff = 10'sd970;
12'd1706: coeff = 10'sd971;
12'd1707: coeff = 10'sd971;
```

12'd1708: coeff = 10'sd971;
12'd1709: coeff = 10'sd972;
12'd1710: coeff = 10'sd972;
12'd1711: coeff = 10'sd972;
12'd1712: coeff = 10'sd973;
12'd1713: coeff = 10'sd973;
12'd1714: coeff = 10'sd973;
12'd1715: coeff = 10'sd974;
12'd1716: coeff = 10'sd974;
12'd1717: coeff = 10'sd974;
12'd1718: coeff = 10'sd975;
12'd1719: coeff = 10'sd975;
12'd1720: coeff = 10'sd975;
12'd1721: coeff = 10'sd976;
12'd1722: coeff = 10'sd976;
12'd1723: coeff = 10'sd976;
12'd1724: coeff = 10'sd977;
12'd1725: coeff = 10'sd977;
12'd1726: coeff = 10'sd977;
12'd1727: coeff = 10'sd978;
12'd1728: coeff = 10'sd978;
12'd1729: coeff = 10'sd978;
12'd1730: coeff = 10'sd979;
12'd1731: coeff = 10'sd979;
12'd1732: coeff = 10'sd979;
12'd1733: coeff = 10'sd980;
12'd1734: coeff = 10'sd980;
12'd1735: coeff = 10'sd980;
12'd1736: coeff = 10'sd981;
12'd1737: coeff = 10'sd981;
12'd1738: coeff = 10'sd981;
12'd1739: coeff = 10'sd982;
12'd1740: coeff = 10'sd982;
12'd1741: coeff = 10'sd982;
12'd1742: coeff = 10'sd983;
12'd1743: coeff = 10'sd983;
12'd1744: coeff = 10'sd983;
12'd1745: coeff = 10'sd984;
12'd1746: coeff = 10'sd984;
12'd1747: coeff = 10'sd984;
12'd1748: coeff = 10'sd985;
12'd1749: coeff = 10'sd985;
12'd1750: coeff = 10'sd985;
12'd1751: coeff = 10'sd985;
12'd1752: coeff = 10'sd986;
12'd1753: coeff = 10'sd986;
12'd1754: coeff = 10'sd986;
12'd1755: coeff = 10'sd987;

```
12'd1756: coeff = 10'sd987;
12'd1757: coeff = 10'sd987;
12'd1758: coeff = 10'sd988;
12'd1759: coeff = 10'sd988;
12'd1760: coeff = 10'sd988;
12'd1761: coeff = 10'sd988;
12'd1762: coeff = 10'sd989;
12'd1763: coeff = 10'sd989;
12'd1764: coeff = 10'sd989;
12'd1765: coeff = 10'sd990;
12'd1766: coeff = 10'sd990;
12'd1767: coeff = 10'sd990;
12'd1768: coeff = 10'sd991;
12'd1769: coeff = 10'sd991;
12'd1770: coeff = 10'sd991;
12'd1771: coeff = 10'sd991;
12'd1772: coeff = 10'sd992;
12'd1773: coeff = 10'sd992;
12'd1774: coeff = 10'sd992;
12'd1775: coeff = 10'sd992;
12'd1776: coeff = 10'sd993;
12'd1777: coeff = 10'sd993;
12'd1778: coeff = 10'sd993;
12'd1779: coeff = 10'sd994;
12'd1780: coeff = 10'sd994;
12'd1781: coeff = 10'sd994;
12'd1782: coeff = 10'sd994;
12'd1783: coeff = 10'sd995;
12'd1784: coeff = 10'sd995;
12'd1785: coeff = 10'sd995;
12'd1786: coeff = 10'sd995;
12'd1787: coeff = 10'sd996;
12'd1788: coeff = 10'sd996;
12'd1789: coeff = 10'sd996;
12'd1790: coeff = 10'sd997;
12'd1791: coeff = 10'sd997;
12'd1792: coeff = 10'sd997;
12'd1793: coeff = 10'sd997;
12'd1794: coeff = 10'sd998;
12'd1795: coeff = 10'sd998;
12'd1796: coeff = 10'sd998;
12'd1797: coeff = 10'sd998;
12'd1798: coeff = 10'sd999;
12'd1799: coeff = 10'sd999;
12'd1800: coeff = 10'sd999;
12'd1801: coeff = 10'sd999;
12'd1802: coeff = 10'sd1000;
12'd1803: coeff = 10'sd1000;
```

12'd1804: coeff = 10'sd1000;
12'd1805: coeff = 10'sd1000;
12'd1806: coeff = 10'sd1001;
12'd1807: coeff = 10'sd1001;
12'd1808: coeff = 10'sd1001;
12'd1809: coeff = 10'sd1001;
12'd1810: coeff = 10'sd1001;
12'd1811: coeff = 10'sd1002;
12'd1812: coeff = 10'sd1002;
12'd1813: coeff = 10'sd1002;
12'd1814: coeff = 10'sd1002;
12'd1815: coeff = 10'sd1003;
12'd1816: coeff = 10'sd1003;
12'd1817: coeff = 10'sd1003;
12'd1818: coeff = 10'sd1003;
12'd1819: coeff = 10'sd1004;
12'd1820: coeff = 10'sd1004;
12'd1821: coeff = 10'sd1004;
12'd1822: coeff = 10'sd1004;
12'd1823: coeff = 10'sd1004;
12'd1824: coeff = 10'sd1005;
12'd1825: coeff = 10'sd1005;
12'd1826: coeff = 10'sd1005;
12'd1827: coeff = 10'sd1005;
12'd1828: coeff = 10'sd1006;
12'd1829: coeff = 10'sd1006;
12'd1830: coeff = 10'sd1006;
12'd1831: coeff = 10'sd1006;
12'd1832: coeff = 10'sd1006;
12'd1833: coeff = 10'sd1007;
12'd1834: coeff = 10'sd1007;
12'd1835: coeff = 10'sd1007;
12'd1836: coeff = 10'sd1007;
12'd1837: coeff = 10'sd1007;
12'd1838: coeff = 10'sd1008;
12'd1839: coeff = 10'sd1008;
12'd1840: coeff = 10'sd1008;
12'd1841: coeff = 10'sd1008;
12'd1842: coeff = 10'sd1008;
12'd1843: coeff = 10'sd1009;
12'd1844: coeff = 10'sd1009;
12'd1845: coeff = 10'sd1009;
12'd1846: coeff = 10'sd1009;
12'd1847: coeff = 10'sd1009;
12'd1848: coeff = 10'sd1010;
12'd1849: coeff = 10'sd1010;
12'd1850: coeff = 10'sd1010;
12'd1851: coeff = 10'sd1010;

12'd1852: coeff = 10'sd1010;
12'd1853: coeff = 10'sd1010;
12'd1854: coeff = 10'sd1011;
12'd1855: coeff = 10'sd1011;
12'd1856: coeff = 10'sd1011;
12'd1857: coeff = 10'sd1011;
12'd1858: coeff = 10'sd1011;
12'd1859: coeff = 10'sd1012;
12'd1860: coeff = 10'sd1012;
12'd1861: coeff = 10'sd1012;
12'd1862: coeff = 10'sd1012;
12'd1863: coeff = 10'sd1012;
12'd1864: coeff = 10'sd1012;
12'd1865: coeff = 10'sd1013;
12'd1866: coeff = 10'sd1013;
12'd1867: coeff = 10'sd1013;
12'd1868: coeff = 10'sd1013;
12'd1869: coeff = 10'sd1013;
12'd1870: coeff = 10'sd1013;
12'd1871: coeff = 10'sd1014;
12'd1872: coeff = 10'sd1014;
12'd1873: coeff = 10'sd1014;
12'd1874: coeff = 10'sd1014;
12'd1875: coeff = 10'sd1014;
12'd1876: coeff = 10'sd1014;
12'd1877: coeff = 10'sd1015;
12'd1878: coeff = 10'sd1015;
12'd1879: coeff = 10'sd1015;
12'd1880: coeff = 10'sd1015;
12'd1881: coeff = 10'sd1015;
12'd1882: coeff = 10'sd1015;
12'd1883: coeff = 10'sd1015;
12'd1884: coeff = 10'sd1016;
12'd1885: coeff = 10'sd1016;
12'd1886: coeff = 10'sd1016;
12'd1887: coeff = 10'sd1016;
12'd1888: coeff = 10'sd1016;
12'd1889: coeff = 10'sd1016;
12'd1890: coeff = 10'sd1016;
12'd1891: coeff = 10'sd1017;
12'd1892: coeff = 10'sd1017;
12'd1893: coeff = 10'sd1017;
12'd1894: coeff = 10'sd1017;
12'd1895: coeff = 10'sd1017;
12'd1896: coeff = 10'sd1017;
12'd1897: coeff = 10'sd1017;
12'd1898: coeff = 10'sd1018;
12'd1899: coeff = 10'sd1018;

```
12'd1900: coeff = 10'sd1018;
12'd1901: coeff = 10'sd1018;
12'd1902: coeff = 10'sd1018;
12'd1903: coeff = 10'sd1018;
12'd1904: coeff = 10'sd1018;
12'd1905: coeff = 10'sd1018;
12'd1906: coeff = 10'sd1018;
12'd1907: coeff = 10'sd1019;
12'd1908: coeff = 10'sd1019;
12'd1909: coeff = 10'sd1019;
12'd1910: coeff = 10'sd1019;
12'd1911: coeff = 10'sd1019;
12'd1912: coeff = 10'sd1019;
12'd1913: coeff = 10'sd1019;
12'd1914: coeff = 10'sd1019;
12'd1915: coeff = 10'sd1019;
12'd1916: coeff = 10'sd1020;
12'd1917: coeff = 10'sd1020;
12'd1918: coeff = 10'sd1020;
12'd1919: coeff = 10'sd1020;
12'd1920: coeff = 10'sd1020;
12'd1921: coeff = 10'sd1020;
12'd1922: coeff = 10'sd1020;
12'd1923: coeff = 10'sd1020;
12'd1924: coeff = 10'sd1020;
12'd1925: coeff = 10'sd1020;
12'd1926: coeff = 10'sd1021;
12'd1927: coeff = 10'sd1021;
12'd1928: coeff = 10'sd1021;
12'd1929: coeff = 10'sd1021;
12'd1930: coeff = 10'sd1021;
12'd1931: coeff = 10'sd1021;
12'd1932: coeff = 10'sd1021;
12'd1933: coeff = 10'sd1021;
12'd1934: coeff = 10'sd1021;
12'd1935: coeff = 10'sd1021;
12'd1936: coeff = 10'sd1021;
12'd1937: coeff = 10'sd1022;
12'd1938: coeff = 10'sd1022;
12'd1939: coeff = 10'sd1022;
12'd1940: coeff = 10'sd1022;
12'd1941: coeff = 10'sd1022;
12'd1942: coeff = 10'sd1022;
12'd1943: coeff = 10'sd1022;
12'd1944: coeff = 10'sd1022;
12'd1945: coeff = 10'sd1022;
12'd1946: coeff = 10'sd1022;
12'd1947: coeff = 10'sd1022;
```

```
12'd1948: coeff = 10'sd1022;
12'd1949: coeff = 10'sd1022;
12'd1950: coeff = 10'sd1022;
12'd1951: coeff = 10'sd1023;
12'd1952: coeff = 10'sd1023;
12'd1953: coeff = 10'sd1023;
12'd1954: coeff = 10'sd1023;
12'd1955: coeff = 10'sd1023;
12'd1956: coeff = 10'sd1023;
12'd1957: coeff = 10'sd1023;
12'd1958: coeff = 10'sd1023;
12'd1959: coeff = 10'sd1023;
12'd1960: coeff = 10'sd1023;
12'd1961: coeff = 10'sd1023;
12'd1962: coeff = 10'sd1023;
12'd1963: coeff = 10'sd1023;
12'd1964: coeff = 10'sd1023;
12'd1965: coeff = 10'sd1023;
12'd1966: coeff = 10'sd1023;
12'd1967: coeff = 10'sd1023;
12'd1968: coeff = 10'sd1023;
12'd1969: coeff = 10'sd1023;
12'd1970: coeff = 10'sd1023;
12'd1971: coeff = 10'sd1023;
12'd1972: coeff = 10'sd1024;
12'd1973: coeff = 10'sd1024;
12'd1974: coeff = 10'sd1024;
12'd1975: coeff = 10'sd1024;
12'd1976: coeff = 10'sd1024;
12'd1977: coeff = 10'sd1024;
12'd1978: coeff = 10'sd1024;
12'd1979: coeff = 10'sd1024;
12'd1980: coeff = 10'sd1024;
12'd1981: coeff = 10'sd1024;
12'd1982: coeff = 10'sd1024;
12'd1983: coeff = 10'sd1024;
12'd1984: coeff = 10'sd1024;
12'd1985: coeff = 10'sd1024;
12'd1986: coeff = 10'sd1024;
12'd1987: coeff = 10'sd1024;
12'd1988: coeff = 10'sd1024;
12'd1989: coeff = 10'sd1024;
12'd1990: coeff = 10'sd1024;
12'd1991: coeff = 10'sd1024;
12'd1992: coeff = 10'sd1024;
12'd1993: coeff = 10'sd1024;
12'd1994: coeff = 10'sd1024;
12'd1995: coeff = 10'sd1024;
```

12'd1996: coeff = 10'sd1024;
12'd1997: coeff = 10'sd1024;
12'd1998: coeff = 10'sd1024;
12'd1999: coeff = 10'sd1024;
12'd2000: coeff = 10'sd1024;
12'd2001: coeff = 10'sd1024;
12'd2002: coeff = 10'sd1024;
12'd2003: coeff = 10'sd1024;
12'd2004: coeff = 10'sd1024;
12'd2005: coeff = 10'sd1024;
12'd2006: coeff = 10'sd1024;
12'd2007: coeff = 10'sd1024;
12'd2008: coeff = 10'sd1024;
12'd2009: coeff = 10'sd1024;
12'd2010: coeff = 10'sd1024;
12'd2011: coeff = 10'sd1024;
12'd2012: coeff = 10'sd1024;
12'd2013: coeff = 10'sd1024;
12'd2014: coeff = 10'sd1024;
12'd2015: coeff = 10'sd1024;
12'd2016: coeff = 10'sd1024;
12'd2017: coeff = 10'sd1024;
12'd2018: coeff = 10'sd1024;
12'd2019: coeff = 10'sd1024;
12'd2020: coeff = 10'sd1024;
12'd2021: coeff = 10'sd1024;
12'd2022: coeff = 10'sd1024;
12'd2023: coeff = 10'sd1024;
12'd2024: coeff = 10'sd1024;
12'd2025: coeff = 10'sd1024;
12'd2026: coeff = 10'sd1024;
12'd2027: coeff = 10'sd1024;
12'd2028: coeff = 10'sd1023;
12'd2029: coeff = 10'sd1023;
12'd2030: coeff = 10'sd1023;
12'd2031: coeff = 10'sd1023;
12'd2032: coeff = 10'sd1023;
12'd2033: coeff = 10'sd1023;
12'd2034: coeff = 10'sd1023;
12'd2035: coeff = 10'sd1023;
12'd2036: coeff = 10'sd1023;
12'd2037: coeff = 10'sd1023;
12'd2038: coeff = 10'sd1023;
12'd2039: coeff = 10'sd1023;
12'd2040: coeff = 10'sd1023;
12'd2041: coeff = 10'sd1023;
12'd2042: coeff = 10'sd1023;
12'd2043: coeff = 10'sd1023;

12'd2044: coeff = 10'sd1023;
12'd2045: coeff = 10'sd1023;
12'd2046: coeff = 10'sd1023;
12'd2047: coeff = 10'sd1023;
12'd2048: coeff = 10'sd1023;
12'd2049: coeff = 10'sd1022;
12'd2050: coeff = 10'sd1022;
12'd2051: coeff = 10'sd1022;
12'd2052: coeff = 10'sd1022;
12'd2053: coeff = 10'sd1022;
12'd2054: coeff = 10'sd1022;
12'd2055: coeff = 10'sd1022;
12'd2056: coeff = 10'sd1022;
12'd2057: coeff = 10'sd1022;
12'd2058: coeff = 10'sd1022;
12'd2059: coeff = 10'sd1022;
12'd2060: coeff = 10'sd1022;
12'd2061: coeff = 10'sd1022;
12'd2062: coeff = 10'sd1022;
12'd2063: coeff = 10'sd1021;
12'd2064: coeff = 10'sd1021;
12'd2065: coeff = 10'sd1021;
12'd2066: coeff = 10'sd1021;
12'd2067: coeff = 10'sd1021;
12'd2068: coeff = 10'sd1021;
12'd2069: coeff = 10'sd1021;
12'd2070: coeff = 10'sd1021;
12'd2071: coeff = 10'sd1021;
12'd2072: coeff = 10'sd1021;
12'd2073: coeff = 10'sd1021;
12'd2074: coeff = 10'sd1020;
12'd2075: coeff = 10'sd1020;
12'd2076: coeff = 10'sd1020;
12'd2077: coeff = 10'sd1020;
12'd2078: coeff = 10'sd1020;
12'd2079: coeff = 10'sd1020;
12'd2080: coeff = 10'sd1020;
12'd2081: coeff = 10'sd1020;
12'd2082: coeff = 10'sd1020;
12'd2083: coeff = 10'sd1020;
12'd2084: coeff = 10'sd1019;
12'd2085: coeff = 10'sd1019;
12'd2086: coeff = 10'sd1019;
12'd2087: coeff = 10'sd1019;
12'd2088: coeff = 10'sd1019;
12'd2089: coeff = 10'sd1019;
12'd2090: coeff = 10'sd1019;
12'd2091: coeff = 10'sd1019;

12'd2092: coeff = 10'sd1019;
12'd2093: coeff = 10'sd1018;
12'd2094: coeff = 10'sd1018;
12'd2095: coeff = 10'sd1018;
12'd2096: coeff = 10'sd1018;
12'd2097: coeff = 10'sd1018;
12'd2098: coeff = 10'sd1018;
12'd2099: coeff = 10'sd1018;
12'd2100: coeff = 10'sd1018;
12'd2101: coeff = 10'sd1018;
12'd2102: coeff = 10'sd1017;
12'd2103: coeff = 10'sd1017;
12'd2104: coeff = 10'sd1017;
12'd2105: coeff = 10'sd1017;
12'd2106: coeff = 10'sd1017;
12'd2107: coeff = 10'sd1017;
12'd2108: coeff = 10'sd1017;
12'd2109: coeff = 10'sd1016;
12'd2110: coeff = 10'sd1016;
12'd2111: coeff = 10'sd1016;
12'd2112: coeff = 10'sd1016;
12'd2113: coeff = 10'sd1016;
12'd2114: coeff = 10'sd1016;
12'd2115: coeff = 10'sd1016;
12'd2116: coeff = 10'sd1015;
12'd2117: coeff = 10'sd1015;
12'd2118: coeff = 10'sd1015;
12'd2119: coeff = 10'sd1015;
12'd2120: coeff = 10'sd1015;
12'd2121: coeff = 10'sd1015;
12'd2122: coeff = 10'sd1015;
12'd2123: coeff = 10'sd1014;
12'd2124: coeff = 10'sd1014;
12'd2125: coeff = 10'sd1014;
12'd2126: coeff = 10'sd1014;
12'd2127: coeff = 10'sd1014;
12'd2128: coeff = 10'sd1014;
12'd2129: coeff = 10'sd1013;
12'd2130: coeff = 10'sd1013;
12'd2131: coeff = 10'sd1013;
12'd2132: coeff = 10'sd1013;
12'd2133: coeff = 10'sd1013;
12'd2134: coeff = 10'sd1013;
12'd2135: coeff = 10'sd1012;
12'd2136: coeff = 10'sd1012;
12'd2137: coeff = 10'sd1012;
12'd2138: coeff = 10'sd1012;
12'd2139: coeff = 10'sd1012;

12'd2140: coeff = 10'sd1012;
12'd2141: coeff = 10'sd1011;
12'd2142: coeff = 10'sd1011;
12'd2143: coeff = 10'sd1011;
12'd2144: coeff = 10'sd1011;
12'd2145: coeff = 10'sd1011;
12'd2146: coeff = 10'sd1010;
12'd2147: coeff = 10'sd1010;
12'd2148: coeff = 10'sd1010;
12'd2149: coeff = 10'sd1010;
12'd2150: coeff = 10'sd1010;
12'd2151: coeff = 10'sd1010;
12'd2152: coeff = 10'sd1009;
12'd2153: coeff = 10'sd1009;
12'd2154: coeff = 10'sd1009;
12'd2155: coeff = 10'sd1009;
12'd2156: coeff = 10'sd1009;
12'd2157: coeff = 10'sd1008;
12'd2158: coeff = 10'sd1008;
12'd2159: coeff = 10'sd1008;
12'd2160: coeff = 10'sd1008;
12'd2161: coeff = 10'sd1008;
12'd2162: coeff = 10'sd1007;
12'd2163: coeff = 10'sd1007;
12'd2164: coeff = 10'sd1007;
12'd2165: coeff = 10'sd1007;
12'd2166: coeff = 10'sd1007;
12'd2167: coeff = 10'sd1006;
12'd2168: coeff = 10'sd1006;
12'd2169: coeff = 10'sd1006;
12'd2170: coeff = 10'sd1006;
12'd2171: coeff = 10'sd1006;
12'd2172: coeff = 10'sd1005;
12'd2173: coeff = 10'sd1005;
12'd2174: coeff = 10'sd1005;
12'd2175: coeff = 10'sd1005;
12'd2176: coeff = 10'sd1004;
12'd2177: coeff = 10'sd1004;
12'd2178: coeff = 10'sd1004;
12'd2179: coeff = 10'sd1004;
12'd2180: coeff = 10'sd1004;
12'd2181: coeff = 10'sd1003;
12'd2182: coeff = 10'sd1003;
12'd2183: coeff = 10'sd1003;
12'd2184: coeff = 10'sd1003;
12'd2185: coeff = 10'sd1002;
12'd2186: coeff = 10'sd1002;
12'd2187: coeff = 10'sd1002;

12'd2188: coeff = 10'sd1002;
12'd2189: coeff = 10'sd1001;
12'd2190: coeff = 10'sd1001;
12'd2191: coeff = 10'sd1001;
12'd2192: coeff = 10'sd1001;
12'd2193: coeff = 10'sd1001;
12'd2194: coeff = 10'sd1000;
12'd2195: coeff = 10'sd1000;
12'd2196: coeff = 10'sd1000;
12'd2197: coeff = 10'sd1000;
12'd2198: coeff = 10'sd999;
12'd2199: coeff = 10'sd999;
12'd2200: coeff = 10'sd999;
12'd2201: coeff = 10'sd999;
12'd2202: coeff = 10'sd998;
12'd2203: coeff = 10'sd998;
12'd2204: coeff = 10'sd998;
12'd2205: coeff = 10'sd998;
12'd2206: coeff = 10'sd997;
12'd2207: coeff = 10'sd997;
12'd2208: coeff = 10'sd997;
12'd2209: coeff = 10'sd997;
12'd2210: coeff = 10'sd996;
12'd2211: coeff = 10'sd996;
12'd2212: coeff = 10'sd996;
12'd2213: coeff = 10'sd995;
12'd2214: coeff = 10'sd995;
12'd2215: coeff = 10'sd995;
12'd2216: coeff = 10'sd995;
12'd2217: coeff = 10'sd994;
12'd2218: coeff = 10'sd994;
12'd2219: coeff = 10'sd994;
12'd2220: coeff = 10'sd994;
12'd2221: coeff = 10'sd993;
12'd2222: coeff = 10'sd993;
12'd2223: coeff = 10'sd993;
12'd2224: coeff = 10'sd992;
12'd2225: coeff = 10'sd992;
12'd2226: coeff = 10'sd992;
12'd2227: coeff = 10'sd992;
12'd2228: coeff = 10'sd991;
12'd2229: coeff = 10'sd991;
12'd2230: coeff = 10'sd991;
12'd2231: coeff = 10'sd991;
12'd2232: coeff = 10'sd990;
12'd2233: coeff = 10'sd990;
12'd2234: coeff = 10'sd990;
12'd2235: coeff = 10'sd989;

12'd2236: coeff = 10'sd989;
12'd2237: coeff = 10'sd989;
12'd2238: coeff = 10'sd988;
12'd2239: coeff = 10'sd988;
12'd2240: coeff = 10'sd988;
12'd2241: coeff = 10'sd988;
12'd2242: coeff = 10'sd987;
12'd2243: coeff = 10'sd987;
12'd2244: coeff = 10'sd987;
12'd2245: coeff = 10'sd986;
12'd2246: coeff = 10'sd986;
12'd2247: coeff = 10'sd986;
12'd2248: coeff = 10'sd985;
12'd2249: coeff = 10'sd985;
12'd2250: coeff = 10'sd985;
12'd2251: coeff = 10'sd985;
12'd2252: coeff = 10'sd984;
12'd2253: coeff = 10'sd984;
12'd2254: coeff = 10'sd984;
12'd2255: coeff = 10'sd983;
12'd2256: coeff = 10'sd983;
12'd2257: coeff = 10'sd983;
12'd2258: coeff = 10'sd982;
12'd2259: coeff = 10'sd982;
12'd2260: coeff = 10'sd982;
12'd2261: coeff = 10'sd981;
12'd2262: coeff = 10'sd981;
12'd2263: coeff = 10'sd981;
12'd2264: coeff = 10'sd980;
12'd2265: coeff = 10'sd980;
12'd2266: coeff = 10'sd980;
12'd2267: coeff = 10'sd979;
12'd2268: coeff = 10'sd979;
12'd2269: coeff = 10'sd979;
12'd2270: coeff = 10'sd978;
12'd2271: coeff = 10'sd978;
12'd2272: coeff = 10'sd978;
12'd2273: coeff = 10'sd977;
12'd2274: coeff = 10'sd977;
12'd2275: coeff = 10'sd977;
12'd2276: coeff = 10'sd976;
12'd2277: coeff = 10'sd976;
12'd2278: coeff = 10'sd976;
12'd2279: coeff = 10'sd975;
12'd2280: coeff = 10'sd975;
12'd2281: coeff = 10'sd975;
12'd2282: coeff = 10'sd974;
12'd2283: coeff = 10'sd974;

```
12'd2284: coeff = 10'sd974;
12'd2285: coeff = 10'sd973;
12'd2286: coeff = 10'sd973;
12'd2287: coeff = 10'sd973;
12'd2288: coeff = 10'sd972;
12'd2289: coeff = 10'sd972;
12'd2290: coeff = 10'sd972;
12'd2291: coeff = 10'sd971;
12'd2292: coeff = 10'sd971;
12'd2293: coeff = 10'sd971;
12'd2294: coeff = 10'sd970;
12'd2295: coeff = 10'sd970;
12'd2296: coeff = 10'sd969;
12'd2297: coeff = 10'sd969;
12'd2298: coeff = 10'sd969;
12'd2299: coeff = 10'sd968;
12'd2300: coeff = 10'sd968;
12'd2301: coeff = 10'sd968;
12'd2302: coeff = 10'sd967;
12'd2303: coeff = 10'sd967;
12'd2304: coeff = 10'sd967;
12'd2305: coeff = 10'sd966;
12'd2306: coeff = 10'sd966;
12'd2307: coeff = 10'sd965;
12'd2308: coeff = 10'sd965;
12'd2309: coeff = 10'sd965;
12'd2310: coeff = 10'sd964;
12'd2311: coeff = 10'sd964;
12'd2312: coeff = 10'sd964;
12'd2313: coeff = 10'sd963;
12'd2314: coeff = 10'sd963;
12'd2315: coeff = 10'sd962;
12'd2316: coeff = 10'sd962;
12'd2317: coeff = 10'sd962;
12'd2318: coeff = 10'sd961;
12'd2319: coeff = 10'sd961;
12'd2320: coeff = 10'sd960;
12'd2321: coeff = 10'sd960;
12'd2322: coeff = 10'sd960;
12'd2323: coeff = 10'sd959;
12'd2324: coeff = 10'sd959;
12'd2325: coeff = 10'sd958;
12'd2326: coeff = 10'sd958;
12'd2327: coeff = 10'sd958;
12'd2328: coeff = 10'sd957;
12'd2329: coeff = 10'sd957;
12'd2330: coeff = 10'sd957;
12'd2331: coeff = 10'sd956;
```

```
12'd2332: coeff = 10'sd956;
12'd2333: coeff = 10'sd955;
12'd2334: coeff = 10'sd955;
12'd2335: coeff = 10'sd954;
12'd2336: coeff = 10'sd954;
12'd2337: coeff = 10'sd954;
12'd2338: coeff = 10'sd953;
12'd2339: coeff = 10'sd953;
12'd2340: coeff = 10'sd952;
12'd2341: coeff = 10'sd952;
12'd2342: coeff = 10'sd952;
12'd2343: coeff = 10'sd951;
12'd2344: coeff = 10'sd951;
12'd2345: coeff = 10'sd950;
12'd2346: coeff = 10'sd950;
12'd2347: coeff = 10'sd950;
12'd2348: coeff = 10'sd949;
12'd2349: coeff = 10'sd949;
12'd2350: coeff = 10'sd948;
12'd2351: coeff = 10'sd948;
12'd2352: coeff = 10'sd947;
12'd2353: coeff = 10'sd947;
12'd2354: coeff = 10'sd947;
12'd2355: coeff = 10'sd946;
12'd2356: coeff = 10'sd946;
12'd2357: coeff = 10'sd945;
12'd2358: coeff = 10'sd945;
12'd2359: coeff = 10'sd944;
12'd2360: coeff = 10'sd944;
12'd2361: coeff = 10'sd944;
12'd2362: coeff = 10'sd943;
12'd2363: coeff = 10'sd943;
12'd2364: coeff = 10'sd942;
12'd2365: coeff = 10'sd942;
12'd2366: coeff = 10'sd941;
12'd2367: coeff = 10'sd941;
12'd2368: coeff = 10'sd941;
12'd2369: coeff = 10'sd940;
12'd2370: coeff = 10'sd940;
12'd2371: coeff = 10'sd939;
12'd2372: coeff = 10'sd939;
12'd2373: coeff = 10'sd938;
12'd2374: coeff = 10'sd938;
12'd2375: coeff = 10'sd937;
12'd2376: coeff = 10'sd937;
12'd2377: coeff = 10'sd937;
12'd2378: coeff = 10'sd936;
12'd2379: coeff = 10'sd936;
```

12'd2380: coeff = 10'sd935;
12'd2381: coeff = 10'sd935;
12'd2382: coeff = 10'sd934;
12'd2383: coeff = 10'sd934;
12'd2384: coeff = 10'sd933;
12'd2385: coeff = 10'sd933;
12'd2386: coeff = 10'sd932;
12'd2387: coeff = 10'sd932;
12'd2388: coeff = 10'sd932;
12'd2389: coeff = 10'sd931;
12'd2390: coeff = 10'sd931;
12'd2391: coeff = 10'sd930;
12'd2392: coeff = 10'sd930;
12'd2393: coeff = 10'sd929;
12'd2394: coeff = 10'sd929;
12'd2395: coeff = 10'sd928;
12'd2396: coeff = 10'sd928;
12'd2397: coeff = 10'sd927;
12'd2398: coeff = 10'sd927;
12'd2399: coeff = 10'sd926;
12'd2400: coeff = 10'sd926;
12'd2401: coeff = 10'sd925;
12'd2402: coeff = 10'sd925;
12'd2403: coeff = 10'sd925;
12'd2404: coeff = 10'sd924;
12'd2405: coeff = 10'sd924;
12'd2406: coeff = 10'sd923;
12'd2407: coeff = 10'sd923;
12'd2408: coeff = 10'sd922;
12'd2409: coeff = 10'sd922;
12'd2410: coeff = 10'sd921;
12'd2411: coeff = 10'sd921;
12'd2412: coeff = 10'sd920;
12'd2413: coeff = 10'sd920;
12'd2414: coeff = 10'sd919;
12'd2415: coeff = 10'sd919;
12'd2416: coeff = 10'sd918;
12'd2417: coeff = 10'sd918;
12'd2418: coeff = 10'sd917;
12'd2419: coeff = 10'sd917;
12'd2420: coeff = 10'sd916;
12'd2421: coeff = 10'sd916;
12'd2422: coeff = 10'sd915;
12'd2423: coeff = 10'sd915;
12'd2424: coeff = 10'sd914;
12'd2425: coeff = 10'sd914;
12'd2426: coeff = 10'sd913;
12'd2427: coeff = 10'sd913;

12'd2428: coeff = 10'sd912;
12'd2429: coeff = 10'sd912;
12'd2430: coeff = 10'sd911;
12'd2431: coeff = 10'sd911;
12'd2432: coeff = 10'sd910;
12'd2433: coeff = 10'sd910;
12'd2434: coeff = 10'sd909;
12'd2435: coeff = 10'sd909;
12'd2436: coeff = 10'sd908;
12'd2437: coeff = 10'sd908;
12'd2438: coeff = 10'sd907;
12'd2439: coeff = 10'sd907;
12'd2440: coeff = 10'sd906;
12'd2441: coeff = 10'sd906;
12'd2442: coeff = 10'sd905;
12'd2443: coeff = 10'sd905;
12'd2444: coeff = 10'sd904;
12'd2445: coeff = 10'sd904;
12'd2446: coeff = 10'sd903;
12'd2447: coeff = 10'sd903;
12'd2448: coeff = 10'sd902;
12'd2449: coeff = 10'sd902;
12'd2450: coeff = 10'sd901;
12'd2451: coeff = 10'sd900;
12'd2452: coeff = 10'sd900;
12'd2453: coeff = 10'sd899;
12'd2454: coeff = 10'sd899;
12'd2455: coeff = 10'sd898;
12'd2456: coeff = 10'sd898;
12'd2457: coeff = 10'sd897;
12'd2458: coeff = 10'sd897;
12'd2459: coeff = 10'sd896;
12'd2460: coeff = 10'sd896;
12'd2461: coeff = 10'sd895;
12'd2462: coeff = 10'sd895;
12'd2463: coeff = 10'sd894;
12'd2464: coeff = 10'sd894;
12'd2465: coeff = 10'sd893;
12'd2466: coeff = 10'sd893;
12'd2467: coeff = 10'sd892;
12'd2468: coeff = 10'sd891;
12'd2469: coeff = 10'sd891;
12'd2470: coeff = 10'sd890;
12'd2471: coeff = 10'sd890;
12'd2472: coeff = 10'sd889;
12'd2473: coeff = 10'sd889;
12'd2474: coeff = 10'sd888;
12'd2475: coeff = 10'sd888;

```
12'd2476: coeff = 10'sd887;
12'd2477: coeff = 10'sd887;
12'd2478: coeff = 10'sd886;
12'd2479: coeff = 10'sd885;
12'd2480: coeff = 10'sd885;
12'd2481: coeff = 10'sd884;
12'd2482: coeff = 10'sd884;
12'd2483: coeff = 10'sd883;
12'd2484: coeff = 10'sd883;
12'd2485: coeff = 10'sd882;
12'd2486: coeff = 10'sd882;
12'd2487: coeff = 10'sd881;
12'd2488: coeff = 10'sd880;
12'd2489: coeff = 10'sd880;
12'd2490: coeff = 10'sd879;
12'd2491: coeff = 10'sd879;
12'd2492: coeff = 10'sd878;
12'd2493: coeff = 10'sd878;
12'd2494: coeff = 10'sd877;
12'd2495: coeff = 10'sd877;
12'd2496: coeff = 10'sd876;
12'd2497: coeff = 10'sd875;
12'd2498: coeff = 10'sd875;
12'd2499: coeff = 10'sd874;
12'd2500: coeff = 10'sd874;
12'd2501: coeff = 10'sd873;
12'd2502: coeff = 10'sd873;
12'd2503: coeff = 10'sd872;
12'd2504: coeff = 10'sd871;
12'd2505: coeff = 10'sd871;
12'd2506: coeff = 10'sd870;
12'd2507: coeff = 10'sd870;
12'd2508: coeff = 10'sd869;
12'd2509: coeff = 10'sd869;
12'd2510: coeff = 10'sd868;
12'd2511: coeff = 10'sd867;
12'd2512: coeff = 10'sd867;
12'd2513: coeff = 10'sd866;
12'd2514: coeff = 10'sd866;
12'd2515: coeff = 10'sd865;
12'd2516: coeff = 10'sd864;
12'd2517: coeff = 10'sd864;
12'd2518: coeff = 10'sd863;
12'd2519: coeff = 10'sd863;
12'd2520: coeff = 10'sd862;
12'd2521: coeff = 10'sd862;
12'd2522: coeff = 10'sd861;
12'd2523: coeff = 10'sd860;
```

```
12'd2524: coeff = 10'sd860;
12'd2525: coeff = 10'sd859;
12'd2526: coeff = 10'sd859;
12'd2527: coeff = 10'sd858;
12'd2528: coeff = 10'sd857;
12'd2529: coeff = 10'sd857;
12'd2530: coeff = 10'sd856;
12'd2531: coeff = 10'sd856;
12'd2532: coeff = 10'sd855;
12'd2533: coeff = 10'sd854;
12'd2534: coeff = 10'sd854;
12'd2535: coeff = 10'sd853;
12'd2536: coeff = 10'sd853;
12'd2537: coeff = 10'sd852;
12'd2538: coeff = 10'sd851;
12'd2539: coeff = 10'sd851;
12'd2540: coeff = 10'sd850;
12'd2541: coeff = 10'sd850;
12'd2542: coeff = 10'sd849;
12'd2543: coeff = 10'sd848;
12'd2544: coeff = 10'sd848;
12'd2545: coeff = 10'sd847;
12'd2546: coeff = 10'sd847;
12'd2547: coeff = 10'sd846;
12'd2548: coeff = 10'sd845;
12'd2549: coeff = 10'sd845;
12'd2550: coeff = 10'sd844;
12'd2551: coeff = 10'sd844;
12'd2552: coeff = 10'sd843;
12'd2553: coeff = 10'sd842;
12'd2554: coeff = 10'sd842;
12'd2555: coeff = 10'sd841;
12'd2556: coeff = 10'sd840;
12'd2557: coeff = 10'sd840;
12'd2558: coeff = 10'sd839;
12'd2559: coeff = 10'sd839;
12'd2560: coeff = 10'sd838;
12'd2561: coeff = 10'sd837;
12'd2562: coeff = 10'sd837;
12'd2563: coeff = 10'sd836;
12'd2564: coeff = 10'sd835;
12'd2565: coeff = 10'sd835;
12'd2566: coeff = 10'sd834;
12'd2567: coeff = 10'sd834;
12'd2568: coeff = 10'sd833;
12'd2569: coeff = 10'sd832;
12'd2570: coeff = 10'sd832;
12'd2571: coeff = 10'sd831;
```

```
12'd2572: coeff = 10'sd830;
12'd2573: coeff = 10'sd830;
12'd2574: coeff = 10'sd829;
12'd2575: coeff = 10'sd829;
12'd2576: coeff = 10'sd828;
12'd2577: coeff = 10'sd827;
12'd2578: coeff = 10'sd827;
12'd2579: coeff = 10'sd826;
12'd2580: coeff = 10'sd825;
12'd2581: coeff = 10'sd825;
12'd2582: coeff = 10'sd824;
12'd2583: coeff = 10'sd823;
12'd2584: coeff = 10'sd823;
12'd2585: coeff = 10'sd822;
12'd2586: coeff = 10'sd822;
12'd2587: coeff = 10'sd821;
12'd2588: coeff = 10'sd820;
12'd2589: coeff = 10'sd820;
12'd2590: coeff = 10'sd819;
12'd2591: coeff = 10'sd818;
12'd2592: coeff = 10'sd818;
12'd2593: coeff = 10'sd817;
12'd2594: coeff = 10'sd816;
12'd2595: coeff = 10'sd816;
12'd2596: coeff = 10'sd815;
12'd2597: coeff = 10'sd814;
12'd2598: coeff = 10'sd814;
12'd2599: coeff = 10'sd813;
12'd2600: coeff = 10'sd813;
12'd2601: coeff = 10'sd812;
12'd2602: coeff = 10'sd811;
12'd2603: coeff = 10'sd811;
12'd2604: coeff = 10'sd810;
12'd2605: coeff = 10'sd809;
12'd2606: coeff = 10'sd809;
12'd2607: coeff = 10'sd808;
12'd2608: coeff = 10'sd807;
12'd2609: coeff = 10'sd807;
12'd2610: coeff = 10'sd806;
12'd2611: coeff = 10'sd805;
12'd2612: coeff = 10'sd805;
12'd2613: coeff = 10'sd804;
12'd2614: coeff = 10'sd803;
12'd2615: coeff = 10'sd803;
12'd2616: coeff = 10'sd802;
12'd2617: coeff = 10'sd801;
12'd2618: coeff = 10'sd801;
12'd2619: coeff = 10'sd800;
```

```
12'd2620: coeff = 10'sd799;
12'd2621: coeff = 10'sd799;
12'd2622: coeff = 10'sd798;
12'd2623: coeff = 10'sd797;
12'd2624: coeff = 10'sd797;
12'd2625: coeff = 10'sd796;
12'd2626: coeff = 10'sd795;
12'd2627: coeff = 10'sd795;
12'd2628: coeff = 10'sd794;
12'd2629: coeff = 10'sd793;
12'd2630: coeff = 10'sd793;
12'd2631: coeff = 10'sd792;
12'd2632: coeff = 10'sd791;
12'd2633: coeff = 10'sd791;
12'd2634: coeff = 10'sd790;
12'd2635: coeff = 10'sd789;
12'd2636: coeff = 10'sd789;
12'd2637: coeff = 10'sd788;
12'd2638: coeff = 10'sd787;
12'd2639: coeff = 10'sd787;
12'd2640: coeff = 10'sd786;
12'd2641: coeff = 10'sd785;
12'd2642: coeff = 10'sd785;
12'd2643: coeff = 10'sd784;
12'd2644: coeff = 10'sd783;
12'd2645: coeff = 10'sd782;
12'd2646: coeff = 10'sd782;
12'd2647: coeff = 10'sd781;
12'd2648: coeff = 10'sd780;
12'd2649: coeff = 10'sd780;
12'd2650: coeff = 10'sd779;
12'd2651: coeff = 10'sd778;
12'd2652: coeff = 10'sd778;
12'd2653: coeff = 10'sd777;
12'd2654: coeff = 10'sd776;
12'd2655: coeff = 10'sd776;
12'd2656: coeff = 10'sd775;
12'd2657: coeff = 10'sd774;
12'd2658: coeff = 10'sd774;
12'd2659: coeff = 10'sd773;
12'd2660: coeff = 10'sd772;
12'd2661: coeff = 10'sd771;
12'd2662: coeff = 10'sd771;
12'd2663: coeff = 10'sd770;
12'd2664: coeff = 10'sd769;
12'd2665: coeff = 10'sd769;
12'd2666: coeff = 10'sd768;
12'd2667: coeff = 10'sd767;
```

```
12'd2668: coeff = 10'sd767;
12'd2669: coeff = 10'sd766;
12'd2670: coeff = 10'sd765;
12'd2671: coeff = 10'sd765;
12'd2672: coeff = 10'sd764;
12'd2673: coeff = 10'sd763;
12'd2674: coeff = 10'sd762;
12'd2675: coeff = 10'sd762;
12'd2676: coeff = 10'sd761;
12'd2677: coeff = 10'sd760;
12'd2678: coeff = 10'sd760;
12'd2679: coeff = 10'sd759;
12'd2680: coeff = 10'sd758;
12'd2681: coeff = 10'sd757;
12'd2682: coeff = 10'sd757;
12'd2683: coeff = 10'sd756;
12'd2684: coeff = 10'sd755;
12'd2685: coeff = 10'sd755;
12'd2686: coeff = 10'sd754;
12'd2687: coeff = 10'sd753;
12'd2688: coeff = 10'sd753;
12'd2689: coeff = 10'sd752;
12'd2690: coeff = 10'sd751;
12'd2691: coeff = 10'sd750;
12'd2692: coeff = 10'sd750;
12'd2693: coeff = 10'sd749;
12'd2694: coeff = 10'sd748;
12'd2695: coeff = 10'sd748;
12'd2696: coeff = 10'sd747;
12'd2697: coeff = 10'sd746;
12'd2698: coeff = 10'sd745;
12'd2699: coeff = 10'sd745;
12'd2700: coeff = 10'sd744;
12'd2701: coeff = 10'sd743;
12'd2702: coeff = 10'sd743;
12'd2703: coeff = 10'sd742;
12'd2704: coeff = 10'sd741;
12'd2705: coeff = 10'sd740;
12'd2706: coeff = 10'sd740;
12'd2707: coeff = 10'sd739;
12'd2708: coeff = 10'sd738;
12'd2709: coeff = 10'sd737;
12'd2710: coeff = 10'sd737;
12'd2711: coeff = 10'sd736;
12'd2712: coeff = 10'sd735;
12'd2713: coeff = 10'sd735;
12'd2714: coeff = 10'sd734;
12'd2715: coeff = 10'sd733;
```

12'd2716: coeff = 10'sd732;
12'd2717: coeff = 10'sd732;
12'd2718: coeff = 10'sd731;
12'd2719: coeff = 10'sd730;
12'd2720: coeff = 10'sd730;
12'd2721: coeff = 10'sd729;
12'd2722: coeff = 10'sd728;
12'd2723: coeff = 10'sd727;
12'd2724: coeff = 10'sd727;
12'd2725: coeff = 10'sd726;
12'd2726: coeff = 10'sd725;
12'd2727: coeff = 10'sd724;
12'd2728: coeff = 10'sd724;
12'd2729: coeff = 10'sd723;
12'd2730: coeff = 10'sd722;
12'd2731: coeff = 10'sd721;
12'd2732: coeff = 10'sd721;
12'd2733: coeff = 10'sd720;
12'd2734: coeff = 10'sd719;
12'd2735: coeff = 10'sd719;
12'd2736: coeff = 10'sd718;
12'd2737: coeff = 10'sd717;
12'd2738: coeff = 10'sd716;
12'd2739: coeff = 10'sd716;
12'd2740: coeff = 10'sd715;
12'd2741: coeff = 10'sd714;
12'd2742: coeff = 10'sd713;
12'd2743: coeff = 10'sd713;
12'd2744: coeff = 10'sd712;
12'd2745: coeff = 10'sd711;
12'd2746: coeff = 10'sd710;
12'd2747: coeff = 10'sd710;
12'd2748: coeff = 10'sd709;
12'd2749: coeff = 10'sd708;
12'd2750: coeff = 10'sd707;
12'd2751: coeff = 10'sd707;
12'd2752: coeff = 10'sd706;
12'd2753: coeff = 10'sd705;
12'd2754: coeff = 10'sd704;
12'd2755: coeff = 10'sd704;
12'd2756: coeff = 10'sd703;
12'd2757: coeff = 10'sd702;
12'd2758: coeff = 10'sd701;
12'd2759: coeff = 10'sd701;
12'd2760: coeff = 10'sd700;
12'd2761: coeff = 10'sd699;
12'd2762: coeff = 10'sd698;
12'd2763: coeff = 10'sd698;

12'd2764: coeff = 10'sd697;
12'd2765: coeff = 10'sd696;
12'd2766: coeff = 10'sd695;
12'd2767: coeff = 10'sd695;
12'd2768: coeff = 10'sd694;
12'd2769: coeff = 10'sd693;
12'd2770: coeff = 10'sd692;
12'd2771: coeff = 10'sd692;
12'd2772: coeff = 10'sd691;
12'd2773: coeff = 10'sd690;
12'd2774: coeff = 10'sd689;
12'd2775: coeff = 10'sd689;
12'd2776: coeff = 10'sd688;
12'd2777: coeff = 10'sd687;
12'd2778: coeff = 10'sd686;
12'd2779: coeff = 10'sd686;
12'd2780: coeff = 10'sd685;
12'd2781: coeff = 10'sd684;
12'd2782: coeff = 10'sd683;
12'd2783: coeff = 10'sd683;
12'd2784: coeff = 10'sd682;
12'd2785: coeff = 10'sd681;
12'd2786: coeff = 10'sd680;
12'd2787: coeff = 10'sd680;
12'd2788: coeff = 10'sd679;
12'd2789: coeff = 10'sd678;
12'd2790: coeff = 10'sd677;
12'd2791: coeff = 10'sd677;
12'd2792: coeff = 10'sd676;
12'd2793: coeff = 10'sd675;
12'd2794: coeff = 10'sd674;
12'd2795: coeff = 10'sd674;
12'd2796: coeff = 10'sd673;
12'd2797: coeff = 10'sd672;
12'd2798: coeff = 10'sd671;
12'd2799: coeff = 10'sd670;
12'd2800: coeff = 10'sd670;
12'd2801: coeff = 10'sd669;
12'd2802: coeff = 10'sd668;
12'd2803: coeff = 10'sd667;
12'd2804: coeff = 10'sd667;
12'd2805: coeff = 10'sd666;
12'd2806: coeff = 10'sd665;
12'd2807: coeff = 10'sd664;
12'd2808: coeff = 10'sd664;
12'd2809: coeff = 10'sd663;
12'd2810: coeff = 10'sd662;
12'd2811: coeff = 10'sd661;

12'd2812: coeff = 10'sd660;
12'd2813: coeff = 10'sd660;
12'd2814: coeff = 10'sd659;
12'd2815: coeff = 10'sd658;
12'd2816: coeff = 10'sd657;
12'd2817: coeff = 10'sd657;
12'd2818: coeff = 10'sd656;
12'd2819: coeff = 10'sd655;
12'd2820: coeff = 10'sd654;
12'd2821: coeff = 10'sd654;
12'd2822: coeff = 10'sd653;
12'd2823: coeff = 10'sd652;
12'd2824: coeff = 10'sd651;
12'd2825: coeff = 10'sd650;
12'd2826: coeff = 10'sd650;
12'd2827: coeff = 10'sd649;
12'd2828: coeff = 10'sd648;
12'd2829: coeff = 10'sd647;
12'd2830: coeff = 10'sd647;
12'd2831: coeff = 10'sd646;
12'd2832: coeff = 10'sd645;
12'd2833: coeff = 10'sd644;
12'd2834: coeff = 10'sd643;
12'd2835: coeff = 10'sd643;
12'd2836: coeff = 10'sd642;
12'd2837: coeff = 10'sd641;
12'd2838: coeff = 10'sd640;
12'd2839: coeff = 10'sd640;
12'd2840: coeff = 10'sd639;
12'd2841: coeff = 10'sd638;
12'd2842: coeff = 10'sd637;
12'd2843: coeff = 10'sd636;
12'd2844: coeff = 10'sd636;
12'd2845: coeff = 10'sd635;
12'd2846: coeff = 10'sd634;
12'd2847: coeff = 10'sd633;
12'd2848: coeff = 10'sd633;
12'd2849: coeff = 10'sd632;
12'd2850: coeff = 10'sd631;
12'd2851: coeff = 10'sd630;
12'd2852: coeff = 10'sd629;
12'd2853: coeff = 10'sd629;
12'd2854: coeff = 10'sd628;
12'd2855: coeff = 10'sd627;
12'd2856: coeff = 10'sd626;
12'd2857: coeff = 10'sd625;
12'd2858: coeff = 10'sd625;
12'd2859: coeff = 10'sd624;

```
12'd2860: coeff = 10'sd623;
12'd2861: coeff = 10'sd622;
12'd2862: coeff = 10'sd622;
12'd2863: coeff = 10'sd621;
12'd2864: coeff = 10'sd620;
12'd2865: coeff = 10'sd619;
12'd2866: coeff = 10'sd618;
12'd2867: coeff = 10'sd618;
12'd2868: coeff = 10'sd617;
12'd2869: coeff = 10'sd616;
12'd2870: coeff = 10'sd615;
12'd2871: coeff = 10'sd614;
12'd2872: coeff = 10'sd614;
12'd2873: coeff = 10'sd613;
12'd2874: coeff = 10'sd612;
12'd2875: coeff = 10'sd611;
12'd2876: coeff = 10'sd611;
12'd2877: coeff = 10'sd610;
12'd2878: coeff = 10'sd609;
12'd2879: coeff = 10'sd608;
12'd2880: coeff = 10'sd607;
12'd2881: coeff = 10'sd607;
12'd2882: coeff = 10'sd606;
12'd2883: coeff = 10'sd605;
12'd2884: coeff = 10'sd604;
12'd2885: coeff = 10'sd603;
12'd2886: coeff = 10'sd603;
12'd2887: coeff = 10'sd602;
12'd2888: coeff = 10'sd601;
12'd2889: coeff = 10'sd600;
12'd2890: coeff = 10'sd599;
12'd2891: coeff = 10'sd599;
12'd2892: coeff = 10'sd598;
12'd2893: coeff = 10'sd597;
12'd2894: coeff = 10'sd596;
12'd2895: coeff = 10'sd595;
12'd2896: coeff = 10'sd595;
12'd2897: coeff = 10'sd594;
12'd2898: coeff = 10'sd593;
12'd2899: coeff = 10'sd592;
12'd2900: coeff = 10'sd592;
12'd2901: coeff = 10'sd591;
12'd2902: coeff = 10'sd590;
12'd2903: coeff = 10'sd589;
12'd2904: coeff = 10'sd588;
12'd2905: coeff = 10'sd588;
12'd2906: coeff = 10'sd587;
12'd2907: coeff = 10'sd586;
```

```
12'd2908: coeff = 10'sd585;
12'd2909: coeff = 10'sd584;
12'd2910: coeff = 10'sd584;
12'd2911: coeff = 10'sd583;
12'd2912: coeff = 10'sd582;
12'd2913: coeff = 10'sd581;
12'd2914: coeff = 10'sd580;
12'd2915: coeff = 10'sd580;
12'd2916: coeff = 10'sd579;
12'd2917: coeff = 10'sd578;
12'd2918: coeff = 10'sd577;
12'd2919: coeff = 10'sd576;
12'd2920: coeff = 10'sd576;
12'd2921: coeff = 10'sd575;
12'd2922: coeff = 10'sd574;
12'd2923: coeff = 10'sd573;
12'd2924: coeff = 10'sd572;
12'd2925: coeff = 10'sd572;
12'd2926: coeff = 10'sd571;
12'd2927: coeff = 10'sd570;
12'd2928: coeff = 10'sd569;
12'd2929: coeff = 10'sd568;
12'd2930: coeff = 10'sd568;
12'd2931: coeff = 10'sd567;
12'd2932: coeff = 10'sd566;
12'd2933: coeff = 10'sd565;
12'd2934: coeff = 10'sd564;
12'd2935: coeff = 10'sd564;
12'd2936: coeff = 10'sd563;
12'd2937: coeff = 10'sd562;
12'd2938: coeff = 10'sd561;
12'd2939: coeff = 10'sd560;
12'd2940: coeff = 10'sd560;
12'd2941: coeff = 10'sd559;
12'd2942: coeff = 10'sd558;
12'd2943: coeff = 10'sd557;
12'd2944: coeff = 10'sd556;
12'd2945: coeff = 10'sd556;
12'd2946: coeff = 10'sd555;
12'd2947: coeff = 10'sd554;
12'd2948: coeff = 10'sd553;
12'd2949: coeff = 10'sd552;
12'd2950: coeff = 10'sd552;
12'd2951: coeff = 10'sd551;
12'd2952: coeff = 10'sd550;
12'd2953: coeff = 10'sd549;
12'd2954: coeff = 10'sd548;
12'd2955: coeff = 10'sd548;
```

```
12'd2956: coeff = 10'sd547;
12'd2957: coeff = 10'sd546;
12'd2958: coeff = 10'sd545;
12'd2959: coeff = 10'sd544;
12'd2960: coeff = 10'sd544;
12'd2961: coeff = 10'sd543;
12'd2962: coeff = 10'sd542;
12'd2963: coeff = 10'sd541;
12'd2964: coeff = 10'sd540;
12'd2965: coeff = 10'sd540;
12'd2966: coeff = 10'sd539;
12'd2967: coeff = 10'sd538;
12'd2968: coeff = 10'sd537;
12'd2969: coeff = 10'sd536;
12'd2970: coeff = 10'sd536;
12'd2971: coeff = 10'sd535;
12'd2972: coeff = 10'sd534;
12'd2973: coeff = 10'sd533;
12'd2974: coeff = 10'sd532;
12'd2975: coeff = 10'sd532;
12'd2976: coeff = 10'sd531;
12'd2977: coeff = 10'sd530;
12'd2978: coeff = 10'sd529;
12'd2979: coeff = 10'sd528;
12'd2980: coeff = 10'sd527;
12'd2981: coeff = 10'sd527;
12'd2982: coeff = 10'sd526;
12'd2983: coeff = 10'sd525;
12'd2984: coeff = 10'sd524;
12'd2985: coeff = 10'sd523;
12'd2986: coeff = 10'sd523;
12'd2987: coeff = 10'sd522;
12'd2988: coeff = 10'sd521;
12'd2989: coeff = 10'sd520;
12'd2990: coeff = 10'sd519;
12'd2991: coeff = 10'sd519;
12'd2992: coeff = 10'sd518;
12'd2993: coeff = 10'sd517;
12'd2994: coeff = 10'sd516;
12'd2995: coeff = 10'sd515;
12'd2996: coeff = 10'sd515;
12'd2997: coeff = 10'sd514;
12'd2998: coeff = 10'sd513;
12'd2999: coeff = 10'sd512;
12'd3000: coeff = 10'sd511;
12'd3001: coeff = 10'sd511;
12'd3002: coeff = 10'sd510;
12'd3003: coeff = 10'sd509;
```

```
12'd3004: coeff = 10'sd508;
12'd3005: coeff = 10'sd507;
12'd3006: coeff = 10'sd507;
12'd3007: coeff = 10'sd506;
12'd3008: coeff = 10'sd505;
12'd3009: coeff = 10'sd504;
12'd3010: coeff = 10'sd503;
12'd3011: coeff = 10'sd503;
12'd3012: coeff = 10'sd502;
12'd3013: coeff = 10'sd501;
12'd3014: coeff = 10'sd500;
12'd3015: coeff = 10'sd499;
12'd3016: coeff = 10'sd499;
12'd3017: coeff = 10'sd498;
12'd3018: coeff = 10'sd497;
12'd3019: coeff = 10'sd496;
12'd3020: coeff = 10'sd495;
12'd3021: coeff = 10'sd495;
12'd3022: coeff = 10'sd494;
12'd3023: coeff = 10'sd493;
12'd3024: coeff = 10'sd492;
12'd3025: coeff = 10'sd491;
12'd3026: coeff = 10'sd490;
12'd3027: coeff = 10'sd490;
12'd3028: coeff = 10'sd489;
12'd3029: coeff = 10'sd488;
12'd3030: coeff = 10'sd487;
12'd3031: coeff = 10'sd486;
12'd3032: coeff = 10'sd486;
12'd3033: coeff = 10'sd485;
12'd3034: coeff = 10'sd484;
12'd3035: coeff = 10'sd483;
12'd3036: coeff = 10'sd482;
12'd3037: coeff = 10'sd482;
12'd3038: coeff = 10'sd481;
12'd3039: coeff = 10'sd480;
12'd3040: coeff = 10'sd479;
12'd3041: coeff = 10'sd478;
12'd3042: coeff = 10'sd478;
12'd3043: coeff = 10'sd477;
12'd3044: coeff = 10'sd476;
12'd3045: coeff = 10'sd475;
12'd3046: coeff = 10'sd474;
12'd3047: coeff = 10'sd474;
12'd3048: coeff = 10'sd473;
12'd3049: coeff = 10'sd472;
12'd3050: coeff = 10'sd471;
12'd3051: coeff = 10'sd470;
```

```
12'd3052: coeff = 10'sd470;
12'd3053: coeff = 10'sd469;
12'd3054: coeff = 10'sd468;
12'd3055: coeff = 10'sd467;
12'd3056: coeff = 10'sd466;
12'd3057: coeff = 10'sd466;
12'd3058: coeff = 10'sd465;
12'd3059: coeff = 10'sd464;
12'd3060: coeff = 10'sd463;
12'd3061: coeff = 10'sd462;
12'd3062: coeff = 10'sd462;
12'd3063: coeff = 10'sd461;
12'd3064: coeff = 10'sd460;
12'd3065: coeff = 10'sd459;
12'd3066: coeff = 10'sd458;
12'd3067: coeff = 10'sd458;
12'd3068: coeff = 10'sd457;
12'd3069: coeff = 10'sd456;
12'd3070: coeff = 10'sd455;
12'd3071: coeff = 10'sd454;
12'd3072: coeff = 10'sd454;
12'd3073: coeff = 10'sd453;
12'd3074: coeff = 10'sd452;
12'd3075: coeff = 10'sd451;
12'd3076: coeff = 10'sd450;
12'd3077: coeff = 10'sd450;
12'd3078: coeff = 10'sd449;
12'd3079: coeff = 10'sd448;
12'd3080: coeff = 10'sd447;
12'd3081: coeff = 10'sd446;
12'd3082: coeff = 10'sd446;
12'd3083: coeff = 10'sd445;
12'd3084: coeff = 10'sd444;
12'd3085: coeff = 10'sd443;
12'd3086: coeff = 10'sd442;
12'd3087: coeff = 10'sd442;
12'd3088: coeff = 10'sd441;
12'd3089: coeff = 10'sd440;
12'd3090: coeff = 10'sd439;
12'd3091: coeff = 10'sd438;
12'd3092: coeff = 10'sd438;
12'd3093: coeff = 10'sd437;
12'd3094: coeff = 10'sd436;
12'd3095: coeff = 10'sd435;
12'd3096: coeff = 10'sd434;
12'd3097: coeff = 10'sd434;
12'd3098: coeff = 10'sd433;
12'd3099: coeff = 10'sd432;
```

```
12'd3100: coeff = 10'sd431;
12'd3101: coeff = 10'sd430;
12'd3102: coeff = 10'sd430;
12'd3103: coeff = 10'sd429;
12'd3104: coeff = 10'sd428;
12'd3105: coeff = 10'sd427;
12'd3106: coeff = 10'sd427;
12'd3107: coeff = 10'sd426;
12'd3108: coeff = 10'sd425;
12'd3109: coeff = 10'sd424;
12'd3110: coeff = 10'sd423;
12'd3111: coeff = 10'sd423;
12'd3112: coeff = 10'sd422;
12'd3113: coeff = 10'sd421;
12'd3114: coeff = 10'sd420;
12'd3115: coeff = 10'sd419;
12'd3116: coeff = 10'sd419;
12'd3117: coeff = 10'sd418;
12'd3118: coeff = 10'sd417;
12'd3119: coeff = 10'sd416;
12'd3120: coeff = 10'sd415;
12'd3121: coeff = 10'sd415;
12'd3122: coeff = 10'sd414;
12'd3123: coeff = 10'sd413;
12'd3124: coeff = 10'sd412;
12'd3125: coeff = 10'sd411;
12'd3126: coeff = 10'sd411;
12'd3127: coeff = 10'sd410;
12'd3128: coeff = 10'sd409;
12'd3129: coeff = 10'sd408;
12'd3130: coeff = 10'sd408;
12'd3131: coeff = 10'sd407;
12'd3132: coeff = 10'sd406;
12'd3133: coeff = 10'sd405;
12'd3134: coeff = 10'sd404;
12'd3135: coeff = 10'sd404;
12'd3136: coeff = 10'sd403;
12'd3137: coeff = 10'sd402;
12'd3138: coeff = 10'sd401;
12'd3139: coeff = 10'sd400;
12'd3140: coeff = 10'sd400;
12'd3141: coeff = 10'sd399;
12'd3142: coeff = 10'sd398;
12'd3143: coeff = 10'sd397;
12'd3144: coeff = 10'sd397;
12'd3145: coeff = 10'sd396;
12'd3146: coeff = 10'sd395;
12'd3147: coeff = 10'sd394;
```

```
12'd3148: coeff = 10'sd393;
12'd3149: coeff = 10'sd393;
12'd3150: coeff = 10'sd392;
12'd3151: coeff = 10'sd391;
12'd3152: coeff = 10'sd390;
12'd3153: coeff = 10'sd390;
12'd3154: coeff = 10'sd389;
12'd3155: coeff = 10'sd388;
12'd3156: coeff = 10'sd387;
12'd3157: coeff = 10'sd386;
12'd3158: coeff = 10'sd386;
12'd3159: coeff = 10'sd385;
12'd3160: coeff = 10'sd384;
12'd3161: coeff = 10'sd383;
12'd3162: coeff = 10'sd382;
12'd3163: coeff = 10'sd382;
12'd3164: coeff = 10'sd381;
12'd3165: coeff = 10'sd380;
12'd3166: coeff = 10'sd379;
12'd3167: coeff = 10'sd379;
12'd3168: coeff = 10'sd378;
12'd3169: coeff = 10'sd377;
12'd3170: coeff = 10'sd376;
12'd3171: coeff = 10'sd376;
12'd3172: coeff = 10'sd375;
12'd3173: coeff = 10'sd374;
12'd3174: coeff = 10'sd373;
12'd3175: coeff = 10'sd372;
12'd3176: coeff = 10'sd372;
12'd3177: coeff = 10'sd371;
12'd3178: coeff = 10'sd370;
12'd3179: coeff = 10'sd369;
12'd3180: coeff = 10'sd369;
12'd3181: coeff = 10'sd368;
12'd3182: coeff = 10'sd367;
12'd3183: coeff = 10'sd366;
12'd3184: coeff = 10'sd365;
12'd3185: coeff = 10'sd365;
12'd3186: coeff = 10'sd364;
12'd3187: coeff = 10'sd363;
12'd3188: coeff = 10'sd362;
12'd3189: coeff = 10'sd362;
12'd3190: coeff = 10'sd361;
12'd3191: coeff = 10'sd360;
12'd3192: coeff = 10'sd359;
12'd3193: coeff = 10'sd359;
12'd3194: coeff = 10'sd358;
12'd3195: coeff = 10'sd357;
```

12'd3196: coeff = 10'sd356;
12'd3197: coeff = 10'sd355;
12'd3198: coeff = 10'sd355;
12'd3199: coeff = 10'sd354;
12'd3200: coeff = 10'sd353;
12'd3201: coeff = 10'sd352;
12'd3202: coeff = 10'sd352;
12'd3203: coeff = 10'sd351;
12'd3204: coeff = 10'sd350;
12'd3205: coeff = 10'sd349;
12'd3206: coeff = 10'sd349;
12'd3207: coeff = 10'sd348;
12'd3208: coeff = 10'sd347;
12'd3209: coeff = 10'sd346;
12'd3210: coeff = 10'sd346;
12'd3211: coeff = 10'sd345;
12'd3212: coeff = 10'sd344;
12'd3213: coeff = 10'sd343;
12'd3214: coeff = 10'sd343;
12'd3215: coeff = 10'sd342;
12'd3216: coeff = 10'sd341;
12'd3217: coeff = 10'sd340;
12'd3218: coeff = 10'sd339;
12'd3219: coeff = 10'sd339;
12'd3220: coeff = 10'sd338;
12'd3221: coeff = 10'sd337;
12'd3222: coeff = 10'sd336;
12'd3223: coeff = 10'sd336;
12'd3224: coeff = 10'sd335;
12'd3225: coeff = 10'sd334;
12'd3226: coeff = 10'sd333;
12'd3227: coeff = 10'sd333;
12'd3228: coeff = 10'sd332;
12'd3229: coeff = 10'sd331;
12'd3230: coeff = 10'sd330;
12'd3231: coeff = 10'sd330;
12'd3232: coeff = 10'sd329;
12'd3233: coeff = 10'sd328;
12'd3234: coeff = 10'sd327;
12'd3235: coeff = 10'sd327;
12'd3236: coeff = 10'sd326;
12'd3237: coeff = 10'sd325;
12'd3238: coeff = 10'sd324;
12'd3239: coeff = 10'sd324;
12'd3240: coeff = 10'sd323;
12'd3241: coeff = 10'sd322;
12'd3242: coeff = 10'sd321;
12'd3243: coeff = 10'sd321;

12'd3244: coeff = 10'sd320;
12'd3245: coeff = 10'sd319;
12'd3246: coeff = 10'sd318;
12'd3247: coeff = 10'sd318;
12'd3248: coeff = 10'sd317;
12'd3249: coeff = 10'sd316;
12'd3250: coeff = 10'sd315;
12'd3251: coeff = 10'sd315;
12'd3252: coeff = 10'sd314;
12'd3253: coeff = 10'sd313;
12'd3254: coeff = 10'sd312;
12'd3255: coeff = 10'sd312;
12'd3256: coeff = 10'sd311;
12'd3257: coeff = 10'sd310;
12'd3258: coeff = 10'sd310;
12'd3259: coeff = 10'sd309;
12'd3260: coeff = 10'sd308;
12'd3261: coeff = 10'sd307;
12'd3262: coeff = 10'sd307;
12'd3263: coeff = 10'sd306;
12'd3264: coeff = 10'sd305;
12'd3265: coeff = 10'sd304;
12'd3266: coeff = 10'sd304;
12'd3267: coeff = 10'sd303;
12'd3268: coeff = 10'sd302;
12'd3269: coeff = 10'sd301;
12'd3270: coeff = 10'sd301;
12'd3271: coeff = 10'sd300;
12'd3272: coeff = 10'sd299;
12'd3273: coeff = 10'sd299;
12'd3274: coeff = 10'sd298;
12'd3275: coeff = 10'sd297;
12'd3276: coeff = 10'sd296;
12'd3277: coeff = 10'sd296;
12'd3278: coeff = 10'sd295;
12'd3279: coeff = 10'sd294;
12'd3280: coeff = 10'sd293;
12'd3281: coeff = 10'sd293;
12'd3282: coeff = 10'sd292;
12'd3283: coeff = 10'sd291;
12'd3284: coeff = 10'sd290;
12'd3285: coeff = 10'sd290;
12'd3286: coeff = 10'sd289;
12'd3287: coeff = 10'sd288;
12'd3288: coeff = 10'sd288;
12'd3289: coeff = 10'sd287;
12'd3290: coeff = 10'sd286;
12'd3291: coeff = 10'sd285;

```
12'd3292: coeff = 10'sd285;
12'd3293: coeff = 10'sd284;
12'd3294: coeff = 10'sd283;
12'd3295: coeff = 10'sd283;
12'd3296: coeff = 10'sd282;
12'd3297: coeff = 10'sd281;
12'd3298: coeff = 10'sd280;
12'd3299: coeff = 10'sd280;
12'd3300: coeff = 10'sd279;
12'd3301: coeff = 10'sd278;
12'd3302: coeff = 10'sd278;
12'd3303: coeff = 10'sd277;
12'd3304: coeff = 10'sd276;
12'd3305: coeff = 10'sd275;
12'd3306: coeff = 10'sd275;
12'd3307: coeff = 10'sd274;
12'd3308: coeff = 10'sd273;
12'd3309: coeff = 10'sd273;
12'd3310: coeff = 10'sd272;
12'd3311: coeff = 10'sd271;
12'd3312: coeff = 10'sd270;
12'd3313: coeff = 10'sd270;
12'd3314: coeff = 10'sd269;
12'd3315: coeff = 10'sd268;
12'd3316: coeff = 10'sd268;
12'd3317: coeff = 10'sd267;
12'd3318: coeff = 10'sd266;
12'd3319: coeff = 10'sd265;
12'd3320: coeff = 10'sd265;
12'd3321: coeff = 10'sd264;
12'd3322: coeff = 10'sd263;
12'd3323: coeff = 10'sd263;
12'd3324: coeff = 10'sd262;
12'd3325: coeff = 10'sd261;
12'd3326: coeff = 10'sd261;
12'd3327: coeff = 10'sd260;
12'd3328: coeff = 10'sd259;
12'd3329: coeff = 10'sd258;
12'd3330: coeff = 10'sd258;
12'd3331: coeff = 10'sd257;
12'd3332: coeff = 10'sd256;
12'd3333: coeff = 10'sd256;
12'd3334: coeff = 10'sd255;
12'd3335: coeff = 10'sd254;
12'd3336: coeff = 10'sd254;
12'd3337: coeff = 10'sd253;
12'd3338: coeff = 10'sd252;
12'd3339: coeff = 10'sd251;
```

```
12'd3340: coeff = 10'sd251;
12'd3341: coeff = 10'sd250;
12'd3342: coeff = 10'sd249;
12'd3343: coeff = 10'sd249;
12'd3344: coeff = 10'sd248;
12'd3345: coeff = 10'sd247;
12'd3346: coeff = 10'sd247;
12'd3347: coeff = 10'sd246;
12'd3348: coeff = 10'sd245;
12'd3349: coeff = 10'sd245;
12'd3350: coeff = 10'sd244;
12'd3351: coeff = 10'sd243;
12'd3352: coeff = 10'sd243;
12'd3353: coeff = 10'sd242;
12'd3354: coeff = 10'sd241;
12'd3355: coeff = 10'sd240;
12'd3356: coeff = 10'sd240;
12'd3357: coeff = 10'sd239;
12'd3358: coeff = 10'sd238;
12'd3359: coeff = 10'sd238;
12'd3360: coeff = 10'sd237;
12'd3361: coeff = 10'sd236;
12'd3362: coeff = 10'sd236;
12'd3363: coeff = 10'sd235;
12'd3364: coeff = 10'sd234;
12'd3365: coeff = 10'sd234;
12'd3366: coeff = 10'sd233;
12'd3367: coeff = 10'sd232;
12'd3368: coeff = 10'sd232;
12'd3369: coeff = 10'sd231;
12'd3370: coeff = 10'sd230;
12'd3371: coeff = 10'sd230;
12'd3372: coeff = 10'sd229;
12'd3373: coeff = 10'sd228;
12'd3374: coeff = 10'sd228;
12'd3375: coeff = 10'sd227;
12'd3376: coeff = 10'sd226;
12'd3377: coeff = 10'sd226;
12'd3378: coeff = 10'sd225;
12'd3379: coeff = 10'sd224;
12'd3380: coeff = 10'sd224;
12'd3381: coeff = 10'sd223;
12'd3382: coeff = 10'sd222;
12'd3383: coeff = 10'sd222;
12'd3384: coeff = 10'sd221;
12'd3385: coeff = 10'sd220;
12'd3386: coeff = 10'sd220;
12'd3387: coeff = 10'sd219;
```

12'd3388: coeff = 10'sd218;
12'd3389: coeff = 10'sd218;
12'd3390: coeff = 10'sd217;
12'd3391: coeff = 10'sd216;
12'd3392: coeff = 10'sd216;
12'd3393: coeff = 10'sd215;
12'd3394: coeff = 10'sd214;
12'd3395: coeff = 10'sd214;
12'd3396: coeff = 10'sd213;
12'd3397: coeff = 10'sd212;
12'd3398: coeff = 10'sd212;
12'd3399: coeff = 10'sd211;
12'd3400: coeff = 10'sd211;
12'd3401: coeff = 10'sd210;
12'd3402: coeff = 10'sd209;
12'd3403: coeff = 10'sd209;
12'd3404: coeff = 10'sd208;
12'd3405: coeff = 10'sd207;
12'd3406: coeff = 10'sd207;
12'd3407: coeff = 10'sd206;
12'd3408: coeff = 10'sd205;
12'd3409: coeff = 10'sd205;
12'd3410: coeff = 10'sd204;
12'd3411: coeff = 10'sd203;
12'd3412: coeff = 10'sd203;
12'd3413: coeff = 10'sd202;
12'd3414: coeff = 10'sd201;
12'd3415: coeff = 10'sd201;
12'd3416: coeff = 10'sd200;
12'd3417: coeff = 10'sd200;
12'd3418: coeff = 10'sd199;
12'd3419: coeff = 10'sd198;
12'd3420: coeff = 10'sd198;
12'd3421: coeff = 10'sd197;
12'd3422: coeff = 10'sd196;
12'd3423: coeff = 10'sd196;
12'd3424: coeff = 10'sd195;
12'd3425: coeff = 10'sd194;
12'd3426: coeff = 10'sd194;
12'd3427: coeff = 10'sd193;
12'd3428: coeff = 10'sd193;
12'd3429: coeff = 10'sd192;
12'd3430: coeff = 10'sd191;
12'd3431: coeff = 10'sd191;
12'd3432: coeff = 10'sd190;
12'd3433: coeff = 10'sd189;
12'd3434: coeff = 10'sd189;
12'd3435: coeff = 10'sd188;

12'd3436: coeff = 10'sd188;
12'd3437: coeff = 10'sd187;
12'd3438: coeff = 10'sd186;
12'd3439: coeff = 10'sd186;
12'd3440: coeff = 10'sd185;
12'd3441: coeff = 10'sd184;
12'd3442: coeff = 10'sd184;
12'd3443: coeff = 10'sd183;
12'd3444: coeff = 10'sd183;
12'd3445: coeff = 10'sd182;
12'd3446: coeff = 10'sd181;
12'd3447: coeff = 10'sd181;
12'd3448: coeff = 10'sd180;
12'd3449: coeff = 10'sd180;
12'd3450: coeff = 10'sd179;
12'd3451: coeff = 10'sd178;
12'd3452: coeff = 10'sd178;
12'd3453: coeff = 10'sd177;
12'd3454: coeff = 10'sd177;
12'd3455: coeff = 10'sd176;
12'd3456: coeff = 10'sd175;
12'd3457: coeff = 10'sd175;
12'd3458: coeff = 10'sd174;
12'd3459: coeff = 10'sd173;
12'd3460: coeff = 10'sd173;
12'd3461: coeff = 10'sd172;
12'd3462: coeff = 10'sd172;
12'd3463: coeff = 10'sd171;
12'd3464: coeff = 10'sd170;
12'd3465: coeff = 10'sd170;
12'd3466: coeff = 10'sd169;
12'd3467: coeff = 10'sd169;
12'd3468: coeff = 10'sd168;
12'd3469: coeff = 10'sd167;
12'd3470: coeff = 10'sd167;
12'd3471: coeff = 10'sd166;
12'd3472: coeff = 10'sd166;
12'd3473: coeff = 10'sd165;
12'd3474: coeff = 10'sd165;
12'd3475: coeff = 10'sd164;
12'd3476: coeff = 10'sd163;
12'd3477: coeff = 10'sd163;
12'd3478: coeff = 10'sd162;
12'd3479: coeff = 10'sd162;
12'd3480: coeff = 10'sd161;
12'd3481: coeff = 10'sd160;
12'd3482: coeff = 10'sd160;
12'd3483: coeff = 10'sd159;

12'd3484: coeff = 10'sd159;
12'd3485: coeff = 10'sd158;
12'd3486: coeff = 10'sd158;
12'd3487: coeff = 10'sd157;
12'd3488: coeff = 10'sd156;
12'd3489: coeff = 10'sd156;
12'd3490: coeff = 10'sd155;
12'd3491: coeff = 10'sd155;
12'd3492: coeff = 10'sd154;
12'd3493: coeff = 10'sd153;
12'd3494: coeff = 10'sd153;
12'd3495: coeff = 10'sd152;
12'd3496: coeff = 10'sd152;
12'd3497: coeff = 10'sd151;
12'd3498: coeff = 10'sd151;
12'd3499: coeff = 10'sd150;
12'd3500: coeff = 10'sd149;
12'd3501: coeff = 10'sd149;
12'd3502: coeff = 10'sd148;
12'd3503: coeff = 10'sd148;
12'd3504: coeff = 10'sd147;
12'd3505: coeff = 10'sd147;
12'd3506: coeff = 10'sd146;
12'd3507: coeff = 10'sd146;
12'd3508: coeff = 10'sd145;
12'd3509: coeff = 10'sd144;
12'd3510: coeff = 10'sd144;
12'd3511: coeff = 10'sd143;
12'd3512: coeff = 10'sd143;
12'd3513: coeff = 10'sd142;
12'd3514: coeff = 10'sd142;
12'd3515: coeff = 10'sd141;
12'd3516: coeff = 10'sd140;
12'd3517: coeff = 10'sd140;
12'd3518: coeff = 10'sd139;
12'd3519: coeff = 10'sd139;
12'd3520: coeff = 10'sd138;
12'd3521: coeff = 10'sd138;
12'd3522: coeff = 10'sd137;
12'd3523: coeff = 10'sd137;
12'd3524: coeff = 10'sd136;
12'd3525: coeff = 10'sd136;
12'd3526: coeff = 10'sd135;
12'd3527: coeff = 10'sd134;
12'd3528: coeff = 10'sd134;
12'd3529: coeff = 10'sd133;
12'd3530: coeff = 10'sd133;
12'd3531: coeff = 10'sd132;

```
12'd3532: coeff = 10'sd132;
12'd3533: coeff = 10'sd131;
12'd3534: coeff = 10'sd131;
12'd3535: coeff = 10'sd130;
12'd3536: coeff = 10'sd130;
12'd3537: coeff = 10'sd129;
12'd3538: coeff = 10'sd129;
12'd3539: coeff = 10'sd128;
12'd3540: coeff = 10'sd127;
12'd3541: coeff = 10'sd127;
12'd3542: coeff = 10'sd126;
12'd3543: coeff = 10'sd126;
12'd3544: coeff = 10'sd125;
12'd3545: coeff = 10'sd125;
12'd3546: coeff = 10'sd124;
12'd3547: coeff = 10'sd124;
12'd3548: coeff = 10'sd123;
12'd3549: coeff = 10'sd123;
12'd3550: coeff = 10'sd122;
12'd3551: coeff = 10'sd122;
12'd3552: coeff = 10'sd121;
12'd3553: coeff = 10'sd121;
12'd3554: coeff = 10'sd120;
12'd3555: coeff = 10'sd120;
12'd3556: coeff = 10'sd119;
12'd3557: coeff = 10'sd119;
12'd3558: coeff = 10'sd118;
12'd3559: coeff = 10'sd118;
12'd3560: coeff = 10'sd117;
12'd3561: coeff = 10'sd117;
12'd3562: coeff = 10'sd116;
12'd3563: coeff = 10'sd116;
12'd3564: coeff = 10'sd115;
12'd3565: coeff = 10'sd114;
12'd3566: coeff = 10'sd114;
12'd3567: coeff = 10'sd113;
12'd3568: coeff = 10'sd113;
12'd3569: coeff = 10'sd112;
12'd3570: coeff = 10'sd112;
12'd3571: coeff = 10'sd111;
12'd3572: coeff = 10'sd111;
12'd3573: coeff = 10'sd110;
12'd3574: coeff = 10'sd110;
12'd3575: coeff = 10'sd109;
12'd3576: coeff = 10'sd109;
12'd3577: coeff = 10'sd108;
12'd3578: coeff = 10'sd108;
12'd3579: coeff = 10'sd107;
```

```
12'd3580: coeff = 10'sd107;
12'd3581: coeff = 10'sd107;
12'd3582: coeff = 10'sd106;
12'd3583: coeff = 10'sd106;
12'd3584: coeff = 10'sd105;
12'd3585: coeff = 10'sd105;
12'd3586: coeff = 10'sd104;
12'd3587: coeff = 10'sd104;
12'd3588: coeff = 10'sd103;
12'd3589: coeff = 10'sd103;
12'd3590: coeff = 10'sd102;
12'd3591: coeff = 10'sd102;
12'd3592: coeff = 10'sd101;
12'd3593: coeff = 10'sd101;
12'd3594: coeff = 10'sd100;
12'd3595: coeff = 10'sd100;
12'd3596: coeff = 10'sd99;
12'd3597: coeff = 10'sd99;
12'd3598: coeff = 10'sd98;
12'd3599: coeff = 10'sd98;
12'd3600: coeff = 10'sd97;
12'd3601: coeff = 10'sd97;
12'd3602: coeff = 10'sd96;
12'd3603: coeff = 10'sd96;
12'd3604: coeff = 10'sd95;
12'd3605: coeff = 10'sd95;
12'd3606: coeff = 10'sd95;
12'd3607: coeff = 10'sd94;
12'd3608: coeff = 10'sd94;
12'd3609: coeff = 10'sd93;
12'd3610: coeff = 10'sd93;
12'd3611: coeff = 10'sd92;
12'd3612: coeff = 10'sd92;
12'd3613: coeff = 10'sd91;
12'd3614: coeff = 10'sd91;
12'd3615: coeff = 10'sd90;
12'd3616: coeff = 10'sd90;
12'd3617: coeff = 10'sd89;
12'd3618: coeff = 10'sd89;
12'd3619: coeff = 10'sd89;
12'd3620: coeff = 10'sd88;
12'd3621: coeff = 10'sd88;
12'd3622: coeff = 10'sd87;
12'd3623: coeff = 10'sd87;
12'd3624: coeff = 10'sd86;
12'd3625: coeff = 10'sd86;
12'd3626: coeff = 10'sd85;
12'd3627: coeff = 10'sd85;
```

```
12'd3628: coeff = 10'sd85;
12'd3629: coeff = 10'sd84;
12'd3630: coeff = 10'sd84;
12'd3631: coeff = 10'sd83;
12'd3632: coeff = 10'sd83;
12'd3633: coeff = 10'sd82;
12'd3634: coeff = 10'sd82;
12'd3635: coeff = 10'sd81;
12'd3636: coeff = 10'sd81;
12'd3637: coeff = 10'sd81;
12'd3638: coeff = 10'sd80;
12'd3639: coeff = 10'sd80;
12'd3640: coeff = 10'sd79;
12'd3641: coeff = 10'sd79;
12'd3642: coeff = 10'sd78;
12'd3643: coeff = 10'sd78;
12'd3644: coeff = 10'sd78;
12'd3645: coeff = 10'sd77;
12'd3646: coeff = 10'sd77;
12'd3647: coeff = 10'sd76;
12'd3648: coeff = 10'sd76;
12'd3649: coeff = 10'sd75;
12'd3650: coeff = 10'sd75;
12'd3651: coeff = 10'sd75;
12'd3652: coeff = 10'sd74;
12'd3653: coeff = 10'sd74;
12'd3654: coeff = 10'sd73;
12'd3655: coeff = 10'sd73;
12'd3656: coeff = 10'sd73;
12'd3657: coeff = 10'sd72;
12'd3658: coeff = 10'sd72;
12'd3659: coeff = 10'sd71;
12'd3660: coeff = 10'sd71;
12'd3661: coeff = 10'sd71;
12'd3662: coeff = 10'sd70;
12'd3663: coeff = 10'sd70;
12'd3664: coeff = 10'sd69;
12'd3665: coeff = 10'sd69;
12'd3666: coeff = 10'sd68;
12'd3667: coeff = 10'sd68;
12'd3668: coeff = 10'sd68;
12'd3669: coeff = 10'sd67;
12'd3670: coeff = 10'sd67;
12'd3671: coeff = 10'sd66;
12'd3672: coeff = 10'sd66;
12'd3673: coeff = 10'sd66;
12'd3674: coeff = 10'sd65;
12'd3675: coeff = 10'sd65;
```

12'd3676: coeff = 10'sd65;
12'd3677: coeff = 10'sd64;
12'd3678: coeff = 10'sd64;
12'd3679: coeff = 10'sd63;
12'd3680: coeff = 10'sd63;
12'd3681: coeff = 10'sd63;
12'd3682: coeff = 10'sd62;
12'd3683: coeff = 10'sd62;
12'd3684: coeff = 10'sd61;
12'd3685: coeff = 10'sd61;
12'd3686: coeff = 10'sd61;
12'd3687: coeff = 10'sd60;
12'd3688: coeff = 10'sd60;
12'd3689: coeff = 10'sd60;
12'd3690: coeff = 10'sd59;
12'd3691: coeff = 10'sd59;
12'd3692: coeff = 10'sd58;
12'd3693: coeff = 10'sd58;
12'd3694: coeff = 10'sd58;
12'd3695: coeff = 10'sd57;
12'd3696: coeff = 10'sd57;
12'd3697: coeff = 10'sd57;
12'd3698: coeff = 10'sd56;
12'd3699: coeff = 10'sd56;
12'd3700: coeff = 10'sd55;
12'd3701: coeff = 10'sd55;
12'd3702: coeff = 10'sd55;
12'd3703: coeff = 10'sd54;
12'd3704: coeff = 10'sd54;
12'd3705: coeff = 10'sd54;
12'd3706: coeff = 10'sd53;
12'd3707: coeff = 10'sd53;
12'd3708: coeff = 10'sd53;
12'd3709: coeff = 10'sd52;
12'd3710: coeff = 10'sd52;
12'd3711: coeff = 10'sd52;
12'd3712: coeff = 10'sd51;
12'd3713: coeff = 10'sd51;
12'd3714: coeff = 10'sd50;
12'd3715: coeff = 10'sd50;
12'd3716: coeff = 10'sd50;
12'd3717: coeff = 10'sd49;
12'd3718: coeff = 10'sd49;
12'd3719: coeff = 10'sd49;
12'd3720: coeff = 10'sd48;
12'd3721: coeff = 10'sd48;
12'd3722: coeff = 10'sd48;
12'd3723: coeff = 10'sd47;

```
12'd3724: coeff = 10'sd47;
12'd3725: coeff = 10'sd47;
12'd3726: coeff = 10'sd46;
12'd3727: coeff = 10'sd46;
12'd3728: coeff = 10'sd46;
12'd3729: coeff = 10'sd45;
12'd3730: coeff = 10'sd45;
12'd3731: coeff = 10'sd45;
12'd3732: coeff = 10'sd44;
12'd3733: coeff = 10'sd44;
12'd3734: coeff = 10'sd44;
12'd3735: coeff = 10'sd43;
12'd3736: coeff = 10'sd43;
12'd3737: coeff = 10'sd43;
12'd3738: coeff = 10'sd42;
12'd3739: coeff = 10'sd42;
12'd3740: coeff = 10'sd42;
12'd3741: coeff = 10'sd41;
12'd3742: coeff = 10'sd41;
12'd3743: coeff = 10'sd41;
12'd3744: coeff = 10'sd41;
12'd3745: coeff = 10'sd40;
12'd3746: coeff = 10'sd40;
12'd3747: coeff = 10'sd40;
12'd3748: coeff = 10'sd39;
12'd3749: coeff = 10'sd39;
12'd3750: coeff = 10'sd39;
12'd3751: coeff = 10'sd38;
12'd3752: coeff = 10'sd38;
12'd3753: coeff = 10'sd38;
12'd3754: coeff = 10'sd37;
12'd3755: coeff = 10'sd37;
12'd3756: coeff = 10'sd37;
12'd3757: coeff = 10'sd37;
12'd3758: coeff = 10'sd36;
12'd3759: coeff = 10'sd36;
12'd3760: coeff = 10'sd36;
12'd3761: coeff = 10'sd35;
12'd3762: coeff = 10'sd35;
12'd3763: coeff = 10'sd35;
12'd3764: coeff = 10'sd35;
12'd3765: coeff = 10'sd34;
12'd3766: coeff = 10'sd34;
12'd3767: coeff = 10'sd34;
12'd3768: coeff = 10'sd33;
12'd3769: coeff = 10'sd33;
12'd3770: coeff = 10'sd33;
12'd3771: coeff = 10'sd33;
```

12'd3772: coeff = 10'sd32;
12'd3773: coeff = 10'sd32;
12'd3774: coeff = 10'sd32;
12'd3775: coeff = 10'sd31;
12'd3776: coeff = 10'sd31;
12'd3777: coeff = 10'sd31;
12'd3778: coeff = 10'sd31;
12'd3779: coeff = 10'sd30;
12'd3780: coeff = 10'sd30;
12'd3781: coeff = 10'sd30;
12'd3782: coeff = 10'sd29;
12'd3783: coeff = 10'sd29;
12'd3784: coeff = 10'sd29;
12'd3785: coeff = 10'sd29;
12'd3786: coeff = 10'sd28;
12'd3787: coeff = 10'sd28;
12'd3788: coeff = 10'sd28;
12'd3789: coeff = 10'sd28;
12'd3790: coeff = 10'sd27;
12'd3791: coeff = 10'sd27;
12'd3792: coeff = 10'sd27;
12'd3793: coeff = 10'sd27;
12'd3794: coeff = 10'sd26;
12'd3795: coeff = 10'sd26;
12'd3796: coeff = 10'sd26;
12'd3797: coeff = 10'sd26;
12'd3798: coeff = 10'sd25;
12'd3799: coeff = 10'sd25;
12'd3800: coeff = 10'sd25;
12'd3801: coeff = 10'sd25;
12'd3802: coeff = 10'sd24;
12'd3803: coeff = 10'sd24;
12'd3804: coeff = 10'sd24;
12'd3805: coeff = 10'sd24;
12'd3806: coeff = 10'sd23;
12'd3807: coeff = 10'sd23;
12'd3808: coeff = 10'sd23;
12'd3809: coeff = 10'sd23;
12'd3810: coeff = 10'sd22;
12'd3811: coeff = 10'sd22;
12'd3812: coeff = 10'sd22;
12'd3813: coeff = 10'sd22;
12'd3814: coeff = 10'sd21;
12'd3815: coeff = 10'sd21;
12'd3816: coeff = 10'sd21;
12'd3817: coeff = 10'sd21;
12'd3818: coeff = 10'sd21;
12'd3819: coeff = 10'sd20;

```
12'd3820: coeff = 10'sd20;
12'd3821: coeff = 10'sd20;
12'd3822: coeff = 10'sd20;
12'd3823: coeff = 10'sd19;
12'd3824: coeff = 10'sd19;
12'd3825: coeff = 10'sd19;
12'd3826: coeff = 10'sd19;
12'd3827: coeff = 10'sd19;
12'd3828: coeff = 10'sd18;
12'd3829: coeff = 10'sd18;
12'd3830: coeff = 10'sd18;
12'd3831: coeff = 10'sd18;
12'd3832: coeff = 10'sd18;
12'd3833: coeff = 10'sd17;
12'd3834: coeff = 10'sd17;
12'd3835: coeff = 10'sd17;
12'd3836: coeff = 10'sd17;
12'd3837: coeff = 10'sd16;
12'd3838: coeff = 10'sd16;
12'd3839: coeff = 10'sd16;
12'd3840: coeff = 10'sd16;
12'd3841: coeff = 10'sd16;
12'd3842: coeff = 10'sd15;
12'd3843: coeff = 10'sd15;
12'd3844: coeff = 10'sd15;
12'd3845: coeff = 10'sd15;
12'd3846: coeff = 10'sd15;
12'd3847: coeff = 10'sd15;
12'd3848: coeff = 10'sd14;
12'd3849: coeff = 10'sd14;
12'd3850: coeff = 10'sd14;
12'd3851: coeff = 10'sd14;
12'd3852: coeff = 10'sd14;
12'd3853: coeff = 10'sd13;
12'd3854: coeff = 10'sd13;
12'd3855: coeff = 10'sd13;
12'd3856: coeff = 10'sd13;
12'd3857: coeff = 10'sd13;
12'd3858: coeff = 10'sd13;
12'd3859: coeff = 10'sd12;
12'd3860: coeff = 10'sd12;
12'd3861: coeff = 10'sd12;
12'd3862: coeff = 10'sd12;
12'd3863: coeff = 10'sd12;
12'd3864: coeff = 10'sd11;
12'd3865: coeff = 10'sd11;
12'd3866: coeff = 10'sd11;
12'd3867: coeff = 10'sd11;
```

12'd3868: coeff = 10'sd11;
12'd3869: coeff = 10'sd11;
12'd3870: coeff = 10'sd10;
12'd3871: coeff = 10'sd10;
12'd3872: coeff = 10'sd10;
12'd3873: coeff = 10'sd10;
12'd3874: coeff = 10'sd10;
12'd3875: coeff = 10'sd10;
12'd3876: coeff = 10'sd10;
12'd3877: coeff = 10'sd9;
12'd3878: coeff = 10'sd9;
12'd3879: coeff = 10'sd9;
12'd3880: coeff = 10'sd9;
12'd3881: coeff = 10'sd9;
12'd3882: coeff = 10'sd9;
12'd3883: coeff = 10'sd8;
12'd3884: coeff = 10'sd8;
12'd3885: coeff = 10'sd8;
12'd3886: coeff = 10'sd8;
12'd3887: coeff = 10'sd8;
12'd3888: coeff = 10'sd8;
12'd3889: coeff = 10'sd8;
12'd3890: coeff = 10'sd7;
12'd3891: coeff = 10'sd7;
12'd3892: coeff = 10'sd7;
12'd3893: coeff = 10'sd7;
12'd3894: coeff = 10'sd7;
12'd3895: coeff = 10'sd7;
12'd3896: coeff = 10'sd7;
12'd3897: coeff = 10'sd7;
12'd3898: coeff = 10'sd6;
12'd3899: coeff = 10'sd6;
12'd3900: coeff = 10'sd6;
12'd3901: coeff = 10'sd6;
12'd3902: coeff = 10'sd6;
12'd3903: coeff = 10'sd6;
12'd3904: coeff = 10'sd6;
12'd3905: coeff = 10'sd6;
12'd3906: coeff = 10'sd5;
12'd3907: coeff = 10'sd5;
12'd3908: coeff = 10'sd5;
12'd3909: coeff = 10'sd5;
12'd3910: coeff = 10'sd5;
12'd3911: coeff = 10'sd5;
12'd3912: coeff = 10'sd5;
12'd3913: coeff = 10'sd5;
12'd3914: coeff = 10'sd5;
12'd3915: coeff = 10'sd4;

12'd3916: coeff = 10'sd4;
12'd3917: coeff = 10'sd4;
12'd3918: coeff = 10'sd4;
12'd3919: coeff = 10'sd4;
12'd3920: coeff = 10'sd4;
12'd3921: coeff = 10'sd4;
12'd3922: coeff = 10'sd4;
12'd3923: coeff = 10'sd4;
12'd3924: coeff = 10'sd4;
12'd3925: coeff = 10'sd3;
12'd3926: coeff = 10'sd3;
12'd3927: coeff = 10'sd3;
12'd3928: coeff = 10'sd3;
12'd3929: coeff = 10'sd3;
12'd3930: coeff = 10'sd3;
12'd3931: coeff = 10'sd3;
12'd3932: coeff = 10'sd3;
12'd3933: coeff = 10'sd3;
12'd3934: coeff = 10'sd3;
12'd3935: coeff = 10'sd3;
12'd3936: coeff = 10'sd3;
12'd3937: coeff = 10'sd2;
12'd3938: coeff = 10'sd2;
12'd3939: coeff = 10'sd2;
12'd3940: coeff = 10'sd2;
12'd3941: coeff = 10'sd2;
12'd3942: coeff = 10'sd2;
12'd3943: coeff = 10'sd2;
12'd3944: coeff = 10'sd2;
12'd3945: coeff = 10'sd2;
12'd3946: coeff = 10'sd2;
12'd3947: coeff = 10'sd2;
12'd3948: coeff = 10'sd2;
12'd3949: coeff = 10'sd2;
12'd3950: coeff = 10'sd2;
12'd3951: coeff = 10'sd1;
12'd3952: coeff = 10'sd1;
12'd3953: coeff = 10'sd1;
12'd3954: coeff = 10'sd1;
12'd3955: coeff = 10'sd1;
12'd3956: coeff = 10'sd1;
12'd3957: coeff = 10'sd1;
12'd3958: coeff = 10'sd1;
12'd3959: coeff = 10'sd1;
12'd3960: coeff = 10'sd1;
12'd3961: coeff = 10'sd1;
12'd3962: coeff = 10'sd1;
12'd3963: coeff = 10'sd1;

```
                    12'd3964: coeff = 10'sd1;
                    12'd3965: coeff = 10'sd1;
                    12'd3966: coeff = 10'sd1;
                    12'd3967: coeff = 10'sd1;
                    12'd3968: coeff = 10'sd1;
                    12'd3969: coeff = 10'sd1;
                    12'd3970: coeff = 10'sd1;
                    12'd3971: coeff = 10'sd0;
                    12'd3972: coeff = 10'sd0;
                    12'd3973: coeff = 10'sd0;
                    12'd3974: coeff = 10'sd0;
                    12'd3975: coeff = 10'sd0;
                    12'd3976: coeff = 10'sd0;
                    12'd3977: coeff = 10'sd0;
                    12'd3978: coeff = 10'sd0;
                    12'd3979: coeff = 10'sd0;
                    12'd3980: coeff = 10'sd0;
                    12'd3981: coeff = 10'sd0;
                    12'd3982: coeff = 10'sd0;
                    12'd3983: coeff = 10'sd0;
                    12'd3984: coeff = 10'sd0;
                    12'd3985: coeff = 10'sd0;
                    12'd3986: coeff = 10'sd0;
                    12'd3987: coeff = 10'sd0;
                    12'd3988: coeff = 10'sd0;
                    12'd3989: coeff = 10'sd0;
                    12'd3990: coeff = 10'sd0;
                    12'd3991: coeff = 10'sd0;
                    12'd3992: coeff = 10'sd0;
                    12'd3993: coeff = 10'sd0;
                    12'd3994: coeff = 10'sd0;
                    12'd3995: coeff = 10'sd0;
                    12'd3996: coeff = 10'sd0;
                    12'd3997: coeff = 10'sd0;
                    12'd3998: coeff = 10'sd0;
                    12'd3999: coeff = 10'sd0;
                    endcase
endmodule

module hannfilter(
  input wire clk,reset,ready,
  input wire signed [7:0] x,
  output reg signed [27:0] y
);

        reg signed[27:0]accumulator = 0;
        reg [11:0]offset = 0;
        reg signed [7:0] sample [400:0];
```

```
	reg [11:0]index = 0;
	reg [12:0]counter = 0;
	wire signed [9:0]coeff;

	hanncoeffs4000 coeffs(.index(index),.coeff(coeff));

	always @(posedge clk)
		begin
		if (ready)
			begin
			offset <= offset + 1;
			sample[offset] <= x;
			accumulator <= 0;
			index <= 0;
			counter <= 0;
			end
		else if (counter < 4000)
			begin
			accumulator <= accumulator + coeff*sample[offset-index];
			index <= index + 1;
			counter <= counter + 1;
			end
		else if (counter == 4000) y <= accumulator;
		end

endmodule

module Peakfinder(
	input clk, ready, reset,
	input signed [7:0]comb00,comb01,comb02,comb03,comb04,comb05,
	input signed [7:0]comb10,comb11,comb12,comb13,comb14,comb15,
	input signed [7:0]comb20,comb21,comb22,comb23,comb24,comb25,
	input signed [7:0]comb30,comb31,comb32,comb33,comb34,comb35,
	input signed [7:0]comb40,comb41,comb42,comb43,comb44,comb45,
	output reg signed
[15:0]energy60,energy90,energy120,energy180,energy210,energy240,
	output reg [7:0]tempo,
	output reg beat);

	//peakfinder is a different approach to finding beats
	//looks at energy coming in for each byte sample and compares to constant
	//THRESHOLD_ENERGY
	//asserts that a beat has occurred when energy of a byte is above
THRESHOLD_ENERGY

	parameter THRESHOLD_ENERGY = 15000;
	reg [2:0]tap = 2'd0;
	reg signed [15:0]energy;
```

```
        initial tempo = 8'd120;

//      reg signed [15:0]energy60 = 16'd0;
//      reg signed [15:0]energy90 = 16'd0;
//      reg signed [15:0]energy120 = 16'd0;
//      reg signed [15:0]energy180 = 16'd0;
//      reg signed [15:0]energy210 = 16'd0;
//      reg signed [15:0]energy240 = 16'd0;

        always @(posedge clk)
                begin
                if (ready)
                        begin
                        //calculate energy this byte for each tempo
                        energy60 <= (comb00*comb00) + (comb10*comb10)
                        + (comb20*comb20) + (comb30*comb30)
                        + (comb40*comb40);

                        energy90 <= (comb01*comb01) + (comb11*comb11)
                        + (comb21*comb21) + (comb31*comb31)
                        + (comb41*comb41);

                        energy120 <= (comb02*comb02) + (comb12*comb12)
                        + (comb22*comb22) + (comb32*comb32)
                        + (comb42*comb42);

                        energy180 <= (comb03*comb03) + (comb13*comb13)
                        + (comb23*comb23) + (comb33*comb33)
                        + (comb43*comb43);

                        energy210 <= (comb04*comb04) + (comb14*comb14)
                        + (comb24*comb24) + (comb34*comb34)
                        + (comb44*comb44);

                        energy240 <= (comb05*comb05) + (comb15*comb15)
                        + (comb25*comb25) + (comb35*comb35)
                        + (comb45*comb45);

                        tap <= 1;
                        end
                if (tap == 1)
                        begin
                        //sum energy at each tempo
                        //energy <=
energy60+energy90+energy120+energy180+energy210+energy240;
                        energy <= energy120;
                        tap <= 2;
                        end
```

```
		if (tap == 2)
			begin
			//compare byte energy to threshold
			if (energy > THRESHOLD_ENERGY)
				begin
				beat <= 1;
				tap <= 3;
				end
			end
		if (tap==3)
			begin
			tap <= 4;
			end
		if (tap==4)
			begin
			tap <= 0;
			beat <= 0;
			end
		end
endmodule

module Peakpicker(
	input clk, ready, reset,
	input signed [7:0]comb00,comb01,comb02,comb03,comb04,comb05,
	input signed [7:0]comb10,comb11,comb12,comb13,comb14,comb15,
	input signed [7:0]comb20,comb21,comb22,comb23,comb24,comb25,
	input signed [7:0]comb30,comb31,comb32,comb33,comb34,comb35,
	input signed [7:0]comb40,comb41,comb42,comb43,comb44,comb45,
	output reg signed [21:0]byte_energy60,byte_energy90,byte_energy120,
	byte_energy180,byte_energy210,byte_energy240,
	output reg [7:0]tempo,
	output reg beat);

	//sums squared inputs over time to calculate energy of each comb filter output
	//tempo with highest sum is fundemental tempo

	initial tempo = 8'd0;
	reg [3:0]tap = 0;

	reg signed [21:0]max_energy = 0;
	reg [12:0]counter_max;
	reg [12:0]counter;

	reg signed [21:0]energy60 = 0;
	reg signed [21:0]energy90 = 0;
	reg signed [21:0]energy120 = 0;
	reg signed [21:0]energy180 = 0;
	reg signed [21:0]energy210 = 0;
```

```
        reg signed [21:0]energy240 = 0;

//      reg signed [21:0]byte_energy60 = 0;
//      reg signed [21:0]byte_energy90 = 0;
//      reg signed [21:0]byte_energy120 = 0;
//      reg signed [21:0]byte_energy180 = 0;
//      reg signed [21:0]byte_energy210 = 0;
//      reg signed [21:0]byte_energy240 = 0;

                always @(posedge clk)
                        begin
                        if (reset)
                                begin
                                energy60 <= 0;
                                energy90 <= 0;
                                energy120 <= 0;
                                energy180 <= 0;
                                energy210 <= 0;
                                energy240 <= 0;
                                tempo <= 0;
                                max_energy <= 0;
                                counter_max <= 0;
                                counter <= 0;
                                end
                        else if (ready)
                                begin
                                //calculate energy of this byte for each tempo
                                byte_energy60 <= (comb00*comb00) + (comb10*comb10)
                                + (comb20*comb20) + (comb30*comb30)
                                + (comb40*comb40);

                                byte_energy90 <= (comb01*comb01) + (comb11*comb11)
                                + (comb21*comb21) + (comb31*comb31)
                                + (comb41*comb41);

                                byte_energy120 <= (comb02*comb02) + (comb12*comb12)
                                + (comb22*comb22) + (comb32*comb32)
                                + (comb42*comb42);

                                byte_energy180 <= (comb03*comb03) + (comb13*comb13)
                                + (comb23*comb23) + (comb33*comb33)
                                + (comb43*comb43);

                                byte_energy210 <= (comb04*comb04) + (comb14*comb14)
                                + (comb24*comb24) + (comb34*comb34)
                                + (comb44*comb44);

                                byte_energy240 <= (comb05*comb05) + (comb15*comb15)
```

```
        + (comb25*comb25) + (comb35*comb35)
        + (comb45*comb45);

        tap <= 1;
        end

if (tap==1)
        begin
        //add energy for byte to sum reg
        energy60 <= energy60 + byte_energy60;
        energy90 <= energy90 + byte_energy90;
        energy120 <= energy120 + byte_energy120;
        energy180 <= energy180 + byte_energy180;
        energy210 <= energy210 + byte_energy210;
        energy240 <= energy240 + byte_energy240;
        tap <= 2;
        end

        //test if any of these energies is a max, start with higher BPM
        //to prevent picking harmonics
if (tap==2)
        begin
        if (energy240>max_energy)
                begin
                tempo <= 240;
                max_energy <= energy240;
                end
        tap <= 3;
        end

if (tap==3)
        begin
        if (energy210>max_energy)
                begin
                tempo <=210;
                max_energy <= energy210;
                end
        tap <= 4;
        end

if (tap==4)
        begin
        if (energy180>max_energy)
                begin
                tempo <=180;
                max_energy <= energy180;
                end
        tap <= 5;
```

```
        end

if (tap==5)
        begin
        if (energy120>max_energy)
                begin
                tempo <=120;
                max_energy <= energy120;
                end
        tap <= 6;
        end

if (tap==6)
        begin
        if (energy90>max_energy)
                begin
                tempo <=90;
                max_energy <= energy90;
                end
        tap <= 7;
        end

if (tap==7)
        begin
        if (energy60>max_energy)
                begin
                tempo <=60;
                max_energy <= energy60;
                end
        tap <= 8;
        end

if (tap==8)
        begin
        //set counter max based on current tempo
        case(tempo)
                60: counter_max <= 6000;
                90: counter_max <= 4000;
                120: counter_max <= 3000;
                180: counter_max <= 2000;
                210: counter_max <= 1714;
                240: counter_max <= 1500;
        endcase
        tap <= 9;
        end

//set beat high at given tempo
if (counter == 0)
```

```
                        begin
                        beat <= 1;
                        counter <= counter_max;
                        end
                else
                        begin
                        counter <= counter - 1;
                        beat <= 0;
                        end

                end

endmodule

//////////////////////////////////////////////////////////////////////////////
//
// 31-tap FIR filter, 8-bit signed data, 10-bit signed coefficients.
// ready is asserted whenever there is a new sample on the X input,
// the Y output should also be sampled at the same time.  Assumes at
// least 32 clocks between ready assertions.  Note that since the
// coefficients have been scaled by 2**10, so has the output (it's
// expanded from 8 bits to 18 bits).  To get an 8-bit result from the
// filter just divide by 2**10, ie, use Y[17:10].
//
//////////////////////////////////////////////////////////////////////////////

module lowpass3k(
  input wire clock,reset,ready,
  input wire signed [7:0] x,
  output reg signed [17:0] y
);

        reg signed[17:0]accumulator = 0;
        reg [4:0]offset = 0;
        reg signed [7:0] sample [31:0];
        reg [4:0]index = 0;
        reg [5:0]counter = 0;
        wire signed [9:0]coeff;

        coeffs31 coeffs(.index(index),.coeff(coeff));

        always @(posedge clock)
                begin
                if (ready)
                        begin
                        offset <= offset + 1;
                        sample[offset] <= x;
                        accumulator <= 0;
```

```
                index <= 0;
                counter <= 0;
                end
        else if (counter < 31)
                begin
                accumulator <= accumulator + coeff*sample[offset-index];
                index <= index + 1;
                counter <= counter + 1;
                end
        else if (counter == 31) y <= accumulator;
        end

endmodule

//////////////////////////////////////////////////////////////////////////////////
//
// Coefficients for a 31-tap low-pass FIR filter with Wn=.125 (eg, 3kHz for a
// 48kHz sample rate).  Since we're doing integer arithmetic, we've scaled
// the coefficients by 2**10
// Matlab command: round(fir1(30,.125)*1024)
//
//////////////////////////////////////////////////////////////////////////////////

module coeffs31(
  input wire [4:0] index,
  output reg signed [9:0] coeff
);
  // tools will turn this into a 31x10 ROM
  always @(index)
    case (index)
      5'd0:  coeff = -10'sd1;
      5'd1:  coeff = -10'sd1;
      5'd2:  coeff = -10'sd3;
      5'd3:  coeff = -10'sd5;
      5'd4:  coeff = -10'sd6;
      5'd5:  coeff = -10'sd7;
      5'd6:  coeff = -10'sd5;
      5'd7:  coeff = 10'sd0;
      5'd8:  coeff = 10'sd10;
      5'd9:  coeff = 10'sd26;
      5'd10: coeff = 10'sd46;
      5'd11: coeff = 10'sd69;
      5'd12: coeff = 10'sd91;
      5'd13: coeff = 10'sd110;
      5'd14: coeff = 10'sd123;
      5'd15: coeff = 10'sd128;
      5'd16: coeff = 10'sd123;
      5'd17: coeff = 10'sd110;
```

```
      5'd18: coeff = 10'sd91;
      5'd19: coeff = 10'sd69;
      5'd20: coeff = 10'sd46;
      5'd21: coeff = 10'sd26;
      5'd22: coeff = 10'sd10;
      5'd23: coeff = 10'sd0;
      5'd24: coeff = -10'sd5;
      5'd25: coeff = -10'sd7;
      5'd26: coeff = -10'sd6;
      5'd27: coeff = -10'sd5;
      5'd28: coeff = -10'sd3;
      5'd29: coeff = -10'sd1;
      5'd30: coeff = -10'sd1;
      default: coeff = 10'hXXX;
    endcase
endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    20:00:01 12/06/2015
// Design Name:
// Module Name:    graphics_rotation
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module graphics_rotation(
            input clk,
            input reset,
            input init,
            input circle,
            input [2:0] addr_count,
            input [9:0] tempo,
    input [10:0] hcount,
            input [9:0] vcount,
            output [18:0] vram_addr2,
            output [18:0] vram_addr3
```

```
        );

        wire [23:0] init_pixel;
        wire [31:0] angle;
        wire [11:0] x_rot;
        wire [10:0] y_rot;
        wire [11:0] x_trans;
        wire [10:0] y_trans;
        wire [11:0] x_in;
        wire [10:0] y_in;

        wire [10:0] hcount_f = (hcount >= 1055) ? (hcount - 1055) : (hcount + 1);
  wire [9:0] vcount_f = (hcount >= 1055) ? ((vcount == 627) ? 0 : vcount +1) : vcount;

        coordinate_controller c1(.clk(clk), .tempo(tempo), .angle(angle));

        translation t1(.clk(clk), .reset(reset), .dist(10), .x({1'b0, hcount_f[10:0]}), .y({2'b0,
vcount_f[9:0]}),
                        .x_trans(x_trans), .y_trans(y_trans));

        rotation r1(.clk(clk), .reset(reset), .angle(angle), .x({1'b0, hcount_f[10:0]}), .y({2'b0,
vcount_f[9:0]}),
                        .x_rot(x_rot), .y_rot(y_rot));


        wire [18:0] vram_addr_init = {hcount[10:0] + vcount[9:0]*800};

        wire [18:0] vram_addrA = {x_rot[10:0] + y_rot[9:0]*800};
        wire [18:0] vram_addrB = {x_rot[10:0] - 1 + y_rot[9:0]*800};
        wire [18:0] vram_addrC = {x_rot[10:0] + (y_rot[9:0]-1)*800};
        wire [18:0] vram_addrD = {x_rot[10:0] - 1 + (y_rot[9:0]-1)*800};

        wire [18:0] vram_addrA1 = {x_rot[10:0] + y_rot[9:0]*800};
        wire [18:0] vram_addrB1 = {x_rot[10:0] + 1 + y_rot[9:0]*800};
        wire [18:0] vram_addrC1 = {x_rot[10:0] + (y_rot[9:0]+1)*800};
        wire [18:0] vram_addrD1 = {x_rot[10:0] + 1 + (y_rot[9:0]+1)*800};

        wire [18:0] vram_addrTrans = {x_trans[10:0] + y_trans[9:0]*800};

        mux4 addr_mux(.clk(clk), .sel(addr_count), .A(vram_addrA), .B(vram_addrB),
                        .C(vram_addrC), .D(vram_addrD), .Y(vram_addr2));

        mux4 addr_mux1(.clk(clk), .sel(addr_count), .A(vram_addrA1), .B(vram_addrB1),
                        .C(vram_addrC1), .D(vram_addrD1), .Y(vram_addr3));

        image_init im0(.clk(clk), .tempo(tempo), .circle(circle),.hcount(hcount), .vcount(vcount),
.pixel(init_pixel));
```

```
endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    13:19:15 11/29/2015
// Design Name:
// Module Name:    image_init
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module image_init #(parameter COLOR = 16'hFFFF)(
        input clk,
        input [9:0] tempo,
        input [10:0] hcount,
        input [9:0] vcount,
        input vsync,
        input circle,
        output[23:0] pixel);

        wire [23:0] circle1_pix;
        wire [23:0] circle2_pix;
        wire [23:0] circle3_pix;
        wire [23:0] circle4_pix;
        wire [23:0] circle5_pix;
        wire [23:0] circle6_pix;
        wire [23:0] circle7_pix;
        wire [23:0] circle8_pix;
        wire [23:0] circle9_pix;
        wire [23:0] circle10_pix;
        wire [23:0] circle11_pix;
        wire [23:0] pixel_circle;

        wire [23:0] pixel_blob;
        wire [15:0] pixel_top_1;
        wire [15:0] pixel_bot_1;
        wire [15:0] pixel_l_1;
```

```
        wire [15:0] pixel_r_1;

        wire [15:0] pixel_top_2;
        wire [15:0] pixel_bot_2;
        wire [15:0] pixel_l_2;
        wire [15:0] pixel_r_2;

        wire [15:0] pixel_top_3;
        wire [15:0] pixel_bot_3;
        wire [15:0] pixel_l_3;
        wire [15:0] pixel_r_3;

        blob #(.WIDTH(500), .HEIGHT(10), .COLOR(24'h0F0309)) top1(.x(150), .y(50),
.hcount(hcount), .vcount(vcount), .pixel(pixel_top_1));
        blob #(.WIDTH(510), .HEIGHT(10), .COLOR(24'h0F0309)) bottom1(.x(150), .y(550),
.hcount(hcount), .vcount(vcount), .pixel(pixel_bot_1));
        blob #(.WIDTH(10), .HEIGHT(500), .COLOR(24'h0F0309)) left1(.x(150), .y(50),
.hcount(hcount), .vcount(vcount), .pixel(pixel_l_1));
        blob #(.WIDTH(10), .HEIGHT(500), .COLOR(24'h0F0309)) right1(.x(650), .y(50),
.hcount(hcount), .vcount(vcount), .pixel(pixel_r_1));


        blob #(.WIDTH(400), .HEIGHT(10), .COLOR(24'h490099)) top2(.x(200), .y(100),
.hcount(hcount), .vcount(vcount), .pixel(pixel_top_2));
        blob #(.WIDTH(410), .HEIGHT(10), .COLOR(24'h490099)) bottom2(.x(200), .y(500),
.hcount(hcount), .vcount(vcount), .pixel(pixel_bot_2));
        blob #(.WIDTH(10), .HEIGHT(400), .COLOR(24'h490099)) left2(.x(200), .y(100),
.hcount(hcount), .vcount(vcount), .pixel(pixel_l_2));
        blob #(.WIDTH(10), .HEIGHT(400), .COLOR(24'h490099)) right2(.x(600), .y(100),
.hcount(hcount), .vcount(vcount), .pixel(pixel_r_2));

        blob #(.WIDTH(300), .HEIGHT(10), .COLOR(24'h995000)) top3(.x(250), .y(150),
.hcount(hcount), .vcount(vcount), .pixel(pixel_top_3));
        blob #(.WIDTH(310), .HEIGHT(10), .COLOR(24'h995000)) bottom3(.x(250), .y(450),
.hcount(hcount), .vcount(vcount), .pixel(pixel_bot_3));
        blob #(.WIDTH(10), .HEIGHT(300), .COLOR(24'h995000)) left3(.x(250), .y(150),
.hcount(hcount), .vcount(vcount), .pixel(pixel_l_3));
        blob #(.WIDTH(10), .HEIGHT(300), .COLOR(24'h995000)) right3(.x(550), .y(150),
.hcount(hcount), .vcount(vcount), .pixel(pixel_r_3));

        assign pixel_blob = pixel_top_1 | pixel_bot_1 | pixel_l_1 | pixel_r_1 |
                                                pixel_top_2 | pixel_bot_2 | pixel_l_2 | pixel_r_2 |
                                                pixel_top_3 | pixel_bot_3 | pixel_l_3 | pixel_r_3;

        moving_circles #(.COLOR(24'hFF0000)) circle1(.clk(clk), .reset(reset), .vsync(vsync),
.tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle1_pix));
```

```
	moving_circles #(.COLOR(24'h999900), .START(200)) circle2(.clk(clk), .reset(reset),
.vsync(vsync), .tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle2_pix));

	moving_circles #(.COLOR(24'h995000), .START(100)) circle3(.clk(clk), .reset(reset),
.vsync(vsync), .tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle3_pix));

	moving_circles #(.COLOR(24'h009900), .START(50)) circle4(.clk(clk), .reset(reset),
.vsync(vsync), .tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle4_pix));

	moving_circles #(.COLOR(24'h000099), .START(150)) circle5(.clk(clk), .reset(reset),
.vsync(vsync), .tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle5_pix));

	moving_circles #(.COLOR(24'h490099), .START(250)) circle6(.clk(clk), .reset(reset),
.vsync(vsync), .tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle6_pix));

	moving_circles #(.COLOR(24'hF03090)) circle7(.clk(clk), .reset(reset), .vsync(vsync),
.tempo(0),
                                .hcount(hcount), .vcount(vcount), .pixel(circle7_pix));

	moving_circles #(.COLOR(24'h0F0309), .X(100), .Y(100)) circle8(.clk(clk), .reset(reset),
.vsync(vsync), .tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle8_pix));

	moving_circles #(.COLOR(24'h0F0309), .X(700), .Y(500)) circle9(.clk(clk), .reset(reset),
.vsync(vsync), .tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle9_pix));

	moving_circles #(.COLOR(24'h0F0309), .X(100), .Y(500)) circle10(.clk(clk), .reset(reset),
.vsync(vsync), .tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle10_pix));

	moving_circles #(.COLOR(24'h0F0309), .X(700), .Y(100)) circle11(.clk(clk), .reset(reset),
.vsync(vsync), .tempo(tempo),
                                .hcount(hcount), .vcount(vcount), .pixel(circle11_pix));

	assign pixel_circle = (circle1_pix | circle2_pix | circle3_pix |
                        circle4_pix | circle5_pix | circle6_pix | circle7_pix |
                        circle8_pix |circle9_pix | circle10_pix | circle11_pix);

	assign pixel = circle ? pixel_circle : pixel_blob;

endmodule
	//////////////////////////////////////////////////////////////////////
```

```
//
// blob: generate rectangle on screen
//
//////////////////////////////////////////////////////////////////////////////////
module blob
  #(parameter WIDTH = 64,            // default width: 64 pixels
              HEIGHT = 64,           // default height: 64 pixels
              COLOR = 24'hFFFFFF)  // default color: white
  (input [10:0] x,hcount,
   input [9:0] y,vcount,
   output reg [23:0] pixel);

  always @ * begin
    if ((hcount >= x && hcount < (x+WIDTH)) &&
                                    (vcount >= y && vcount < (y+HEIGHT)))
                                    pixel = COLOR;
    else pixel = 0;
  end
endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:11:48:33 12/04/2015
// Design Name:
// Module Name:         color_shft
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module color_shft(
    input reset,
    input shft,
          input [23:0] vr,
          output [23:0] pixel
          );

   reg [4:0] i = 0;
```

```
    wire [7:0] shifted_red;
    wire  [7:0] shifted_green;
    wire  [7:0] shifted_blue;

    always@(posedge shft) begin
            if (reset) i <= 0;
            else i <= i + 1;
    end

    assign shifted_red = (i == 0) ? vr[23:16] : vr[23:16]*(i+3)*i;
    assign shifted_green = (i == 0) ? vr[15:8] : vr[15:8]*(i+2)*i;
    assign shifted_blue = (i == 0) ? vr[7:0] : vr[7:0]*i*i;
//    assign shifted_red = vr[23:16];
//    assign shifted_green = vr[15:8];
//    assign shifted_blue = vr[7:0];
//

    assign pixel = {shifted_red, shifted_green, shifted_blue};

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:23:52:06 11/28/2015
// Design Name:
// Module Name:         coordinate_controller
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module coordinate_controller(
     input clk,
     input [9:0] tempo,
     output [31:0] angle
           );

    // Table of angles to choose from: -16 to 16 degrees
```

```
    // Only need 0-15 actually in table

    parameter TEMPO_MIN = 59;
    parameter TEMPO_MAX = 240;
    parameter TEMPO_STEP = (TEMPO_MAX-TEMPO_MIN);

    wire signed [31:0] ang_table [0:16];
    reg signed [31:0] angle_reg;

    assign ang_table[00] = 32'b0;
    assign ang_table[01] = 32'b00000000101101100000101101100000; // 1 degree
    assign ang_table[02] = 32'b00000001011011000001011011000001; // 2 degrees
    assign ang_table[03] = 32'b00000010001000100010001000100010; // 3 degrees
    assign ang_table[04] = 32'b00000010110110000010110110000011; // etc...
    assign ang_table[05] = 32'b00000011100011100011100011100011;
    assign ang_table[06] = 32'b00000100010001000100010001000100;
    assign ang_table[07] = 32'b00000100111110100100111110100100;
    assign ang_table[08] = 32'b00000101101100000101101100000101;
    assign ang_table[09] = 32'b00000110011001100110011001100110;
    assign ang_table[10] = 32'b00000111000111000111000111000111;
    assign ang_table[11] = 32'b00000111110100100111110100100111;
    assign ang_table[12] = 32'b00001000100010001000100010001000;
    assign ang_table[13] = 32'b00001001001111101001001111101001;
    assign ang_table[14] = 32'b00001001111110100100111101001010;
    assign ang_table[15] = 32'b00001010101010101010101010101010;
    assign ang_table[16] = 32'b00001010101010101010101010101010;

    reg [4:0] i;

    always@(posedge clk) begin
            if (tempo == 0) angle_reg <= ang_table[00];
            else begin
                    for (i = 0; i < 16; i = i+1) begin
                            if ((tempo > TEMPO_MIN+TEMPO_STEP*i/16) && tempo <
TEMPO_MIN+TEMPO_STEP*(i+1)/16)
                            angle_reg <= ang_table[i+1];
                    end
            end
    end

  assign angle = angle_reg;
    //assign angle = 0;

endmodule


`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
```

```
// Company:
// Engineer:
//
// Create Date:13:15:46 12/06/2015
// Design Name:
// Module Name:         moving_circles
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module moving_circles #(parameter COLOR = 24'hFFFFFF, parameter RAD = 100,
    parameter START= 0, parameter X = 400, parameter Y = 300)(
     input clk,
     input reset,
     input vsync,
     input [9:0] tempo,
     input [10:0] hcount,
     input [9:0] vcount,
     output [23:0] pixel,
     output [9:0] count
     );

     parameter RAD_in = RAD-15;

     reg [7:0] rad_count = 0;
     reg [12:0] counter = 0;
     reg [23:0] r_sq_out;
     reg [23:0] r_sq_in;

     reg [10:0] deltax;
     reg [9:0] deltay;


     reg [23:0] pix;
     reg [23:0] dist_out;
     reg [23:0] dist_in;
     reg [10:0] radius_out;
     reg [10:0] radius_in;

     always@(posedge clk) begin
```

```
                radius_out <= RAD + rad_count;
                radius_in <= RAD_in + rad_count;
                r_sq_out <= radius_out*radius_out;
                r_sq_in <= radius_in*radius_in;

                deltax <= (hcount > X) ? (hcount-X) : (X-hcount);
                deltay <= (vcount > Y) ? (vcount-Y) : (Y-vcount);

                dist_out <= deltax*deltax+deltay*deltay;

                if(dist_out <= r_sq_out && dist_out >= r_sq_in) pix <= COLOR;
                else pix <= 0;
    end


    always@(posedge vsync) begin
                if (reset) begin
                        rad_count <= 0;
                        counter <= 0; end
                if (rad_count == 256) counter <= 0;
                else counter <= counter + 1;
                rad_count <= ((counter + tempo/16) < START) ? 0 : rad_count + tempo/16;
    end

    assign count = rad_count;
    assign pixel = pix;

endmodule
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:17:16:09 11/18/2015
// Design Name:
// Module Name:        rotation
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

```
module rotation(
    input clk,
    input reset,
    input signed [31:0] angle,
          input signed [11:0] x,
          input signed [11:0] y,
          output signed [11:0] x_rot,
          output signed [10:0] y_rot
          );
    parameter XSIZE = 12;
    parameter YSIZE = 12;

    wire signed [31:0] atan [0:12] ;
    wire signed [9:0] m;
    wire signed [10:0] n;

//    wire signed [31:0] x_rot_large;
//          wire signed [31:0] y_rot_large;
    reg signed [31:0] x_rot_large;
    reg signed [31:0] y_rot_large;

    //Want a high-precision table of arctan values; however, since our maximum iteration is 11,
    //our table needs only 11 entires.  If we wanted more precise x/y values table would be
larger.
    assign atan[00] = 32'b00100000000000000000000000000000; // 45 degrees or
atan(2^0)
    assign atan[01] = 32'b00010010111001000000010100011101; // atan(2^-1)
    assign atan[02] = 32'b00001001111110110011100001011011; // atan(2^-2)
    assign atan[03] = 32'b00000101000100010001000111010100; // atan(2^-3)
    assign atan[04] = 32'b00000010100010110000110101000011;
    assign atan[05] = 32'b00000001010001011101011111100001;
    assign atan[06] = 32'b00000000101000101111011000011110;
    assign atan[07] = 32'b00000000010100010111110001010101;
    assign atan[08] = 32'b00000000001010001011111001010011;
    assign atan[09] = 32'b00000000000101000101111100101110;
    assign atan[10] = 32'b00000000000010100010111110011000;
    assign atan[11] = 32'b00000000000001010001011111001100;
    assign atan[12] = 32'b00000000000000101000101111100110;

    // Must create shift register to store value of x, since it takes 11 clock cycles to calculate
    reg signed [XSIZE:0] x_reg [0:XSIZE-1];
    reg signed [YSIZE:0] y_reg [0:YSIZE-1];
    reg signed [31:0] z_reg [0:XSIZE-1];

//    reg signed [XSIZE:0] x_reg1 [0:XSIZE-1];
//    reg signed [YSIZE:0] y_reg1 [0:YSIZE-1];
//    reg signed [31:0] z_reg1 [0:XSIZE-1];
```

```
    always@(posedge clk) begin
        // Initialize x_reg and y_reg
        x_reg[0] <= x - 400;
        y_reg[0] <= y - 300;
//      x_reg1[0] <= x - 400;
//      y_reg1[0] <= y - 300;
        z_reg[0] <= angle;

    end

    // Want to generate new foor loop or each new input

    generate
    genvar i;
        for (i=0; i<(XSIZE-1); i= i+1)
            begin: gen1
                wire z_sign = z_reg[i][31];

                wire signed [XSIZE:0] x_shft ;
                wire signed [YSIZE:0] y_shft ;
                assign x_shft = x_reg[i] >>> i;
                assign y_shft = y_reg[i] >>> i;

                always@(posedge clk) begin
                    x_reg[i+1] <= z_sign ? x_reg[i] + y_shft : x_reg[i] - y_shft;
                    y_reg[i+1] <= z_sign ? y_reg[i] - x_shft : y_reg[i] + x_shft;
                    z_reg[i+1] <= z_sign ? z_reg[i] + atan[i] : z_reg[i] - atan[i];
//                  x_reg[i+1] <= x_reg[i];
//                  y_reg[i+1] <= y_reg[i];
//                  z_reg[i+1] <= z_reg[i];
                end
            end
    endgenerate

    // End result has gain of 1.647, want to multiply by inverse .6072. Can be approximated with
311/512 = .6074.
    assign m = 311;
    assign n = 512;

    always@(posedge clk)begin
        x_rot_large <= (x_reg[XSIZE -1] * m) >>> 9;
        y_rot_large <= (y_reg[YSIZE -1] * m) >>> 9;

//      x_rot_large1 <= (x_reg[XSIZE -1] * 2) >>> 1;
//      y_rot_large1 <= (y_reg[YSIZE -1] * 2) >>> 1;
    end
//    assign x_rot_large = x_reg[XSIZE -1] * m / n;
////    assign y_rot_large = y_reg[YSIZE -1] * m / n;
```

```
//    assign x_rot_large = x_reg[XSIZE -1];
//    assign y_rot_large = y_reg[XSIZE -1];

    wire [31:0] y_rot_norm = 300 + y_rot_large;
    wire [31:0] x_rot_norm = 400 + x_rot_large;

//    wire [31:0] y_rot_norm1 = 300 + y_rot_large1;
//    wire [31:0] x_rot_norm1 = 400 + x_rot_large1;

    assign x_rot = {x_rot_norm[31], x_rot_norm[10:0]};
    assign y_rot = {y_rot_norm[31], y_rot_norm[9:0]};

endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:23:11:50 11/29/2015
// Design Name:
// Module Name:         translation
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module translation(
            input clk,
     input reset,
     input signed [4:0] dist,
            input signed [11:0] x,
            input signed [11:0] y,
            output signed [11:0] x_trans,
            output signed [10:0] y_trans
            );

    reg signed [11:0] x_reg;
    reg signed [11:0] y_reg;

    always@(posedge clk) begin
```

```
            x_reg <= x +dist;
            y_reg <= y +dist;
    end

    assign x_trans = x_reg;
    assign y_trans = y_reg[10:0];

endmodule

//
// File:   zbt_6111.v
// Date:   27-Nov-05
// Author: I. Chuang <ichuang@mit.edu>
//
// Simple ZBT driver for the MIT 6.111 labkit, which does not hide the
// pipeline delays of the ZBT from the user.  The ZBT memories have
// two cycle latencies on read and write, and also need extra-long data hold
// times around the clock positive edge to work reliably.
//

////////////////////////////////////////////////////////////////////////////////
// Ike's simple ZBT RAM driver for the MIT 6.111 labkit
//
// Data for writes can be presented and clocked in immediately; the actual
// writing to RAM will happen two cycles later.
//
// Read requests are processed immediately, but the read data is not available
// until two cycles after the intial request.
//
// A clock enable signal is provided; it enables the RAM clock when high.

module zbt_6111(clk, cen, we, addr, write_data, read_data,
                ram_clk, ram_we_b, ram_address, ram_data, ram_cen_b);

    input clk;                  // system clock
    input cen;                  // clock enable for gating ZBT cycles
    input we;                   // write enable (active HIGH)
    input [18:0] addr;          // memory address
    input [35:0] write_data;    // data to write
    output [35:0] read_data;    // data read from memory
    output      ram_clk;    // physical line to ram clock
    output      ram_we_b;    // physical line to ram we_b
    output [18:0] ram_address;    // physical line to ram address
    inout [35:0]  ram_data;    // physical line to ram data
    output      ram_cen_b;    // physical line to ram clock enable

    // clock enable (should be synchronous and one cycle high at a time)
    wire      ram_cen_b = ~cen;
```

```
  // create delayed ram_we signal: note the delay is by two cycles!
  // ie we present the data to be written two cycles after we is raised
  // this means the bus is tri-stated two cycles after we is raised.

  reg [1:0]   we_delay;

  always @(posedge clk)
        we_delay <= cen ? {we_delay[0],we} : we_delay;

  // create two-stage pipeline for write data

  reg [35:0]  write_data_old1;
  reg [35:0]  write_data_old2;
  always @(posedge clk)
        if (cen)
        {write_data_old2, write_data_old1} <= {write_data_old1, write_data};

  // wire to ZBT RAM signals

  assign        ram_we_b = ~we;
  assign        ram_clk = 1'b0;  // gph 2011-Nov-10
                        // set to zero as place holder

//   assign       ram_clk = ~clk;          // RAM is not happy with our data hold
                        // times if its clk edges equal FPGA's
                        // so we clock it on the falling edges
                        // and thus let data stabilize longer
  assign        ram_address = addr;

  assign        ram_data = we_delay[1] ? write_data_old2 : {36{1'bZ}};
  assign        read_data = ram_data;

endmodule // zbt_6111

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:22:30:04 11/30/2015
// Design Name:
// Module Name:        mux4
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
```

```
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module mux4 (clk, sel, A, B, C, D, Y);
    input clk;
    input [1:0] sel;

    input [18:0] A, B, C, D;
    output [18:0] Y;
    reg [18:0] y;

  always @(posedge clk) begin
            case(sel)
                    2'b00: y <= A;
                    2'b01: y <= B;
                    2'b10: y <= C;
                    2'b11: y <= D;
                    default: y <= A;
            endcase
    end

    assign Y = y;
endmodule
```