EE314 DIGITAL ELECTRONICS LABORATORY
FINAL PROJECT REPORT

Project: Dipole (#3)

Doğa Veske 1937572
Baran Bodur 1936467

Ankara 2015

Table of Contents

Page Number

List of Figures and Tables

Abstract

This report presents design and implementation of a creative game on an Altera FPGA board using Verilog. The game is inspired from the Duet game on tablets and phones, and named Dipole. For visualization of the game, VGA output of the FPGA was used. In addition to the visual display, the game also includes stereo music output by utilizing two speakers connected to two different GPIO outputs of the FPGA. The project mainly focuses on creating different shaped obstacles for dipole to pass. For more variety these geometrical obstacles are rotating, and not necessarily coming from the top of the screen but also from the sides. In this report, one can find how rotation and visualization of different geometrical shapes can be implemented with a hardware programming language.

Introduction and Literature Search

An FPGA (Field Programmable Gate Array) is a reprogrammable device that can perform complex operations in hardware level and therefore requiring hardware programming. The concept of hardware programming is very different than its software counterpart. It is a lower level of programming, which needs a more detailed and patient programming approach. However, it is also more rewarding. Given enough logic blocks, many different operations can be done in parallel just in clock cycle, which results in excellent performance. Therefore, FPGA technology is used for its fast response time and its programmability. An FPGA can be programmed in two different ways. First one is directly creating a gate level design; the second is using a hardware programming language such as VHDL or Verilog.
In this project, we designed and programmed a game in Verilog that works in an Altera DE1 - SoC FPGA development board. The name of the game is Dipole and it is inspired from the game Duet which is playable on phones and tablets.
Among five possible game options, we have chosen this game because of its infinite possibilities and algorithmic challenges. During the development, we changed the software programmed Duet into a hardware programmed Dipole with extra features such as various geometrical shapes with various trajectories and rotations.
In order to visualize our game we used the VGA interface of the FPGA board. By creating necessary signals such as a clock, horizontal and vertical synchronization signals and intensity of red, green and blue colors pixel by pixel an image is produced on a monitor.
In addition to the visual interface, we also added music to our game. While most digital games on hardware use one speaker to create a monophonic melody, we created a polyphonic melody (an arrangement of Interstellar main theme) with two speakers. In order to achieve this two signals are defined separately but synchronously and given into two different speakers.
In this project we mainly focused on defining, visualizing and rotating geometrical shapes with the extensive use of analytical geometry and trigonometry. While the game can easily be implemented just with falling rectangles, a creative approach is preferred, for a more enjoyable and challenging game.
In addition to two controls for rotating the dipole to left and right, the game also has a reset and pause for continuous playability. When the game ends, the area of crash is highlighted for players to see and “GG” phrase appears on the screen to emphasis that player performed well. Also the score of the player and time is displayed during a whole game. The vertical speed of obstacles increase and their trajectories change every minute.

During the design of game we learned many subjects by search. First of them is verilog language. A hardware description language differs from a software language, for example in software you can change the value of avariable in multiple different loops. However in hardware you can only change the value of a variable only in one loop. Otherwise you are trying to connect a node to output of several gates which can be independent from each other. In addition we learned the structure of the language and tasks (corresponds to function in software languages) by searching on the internet. Secondly because of we are using a hardware we searched its user manual for it pin codes for assignments and its capabilities. Finally, our design heavily depends on analytical geometry, therefore our search on the internet includes analytic geometry related formulas and conditions such as finding the area of a triangle with the coordinates of its vertices or the condition for a point to be in a rectangle.

Design

The Original Game

The original game, which our project is based on, is called Duet. In that game, the player controls two balls that are positioned on a circle, 180 degrees apart. The player can rotate these two balls synchronously either to the left or right, and tries to avoid rectangular obstacles coming from above. The main challenge of the game is keeping track of the second ball while trying to save the first one from an obstacle.

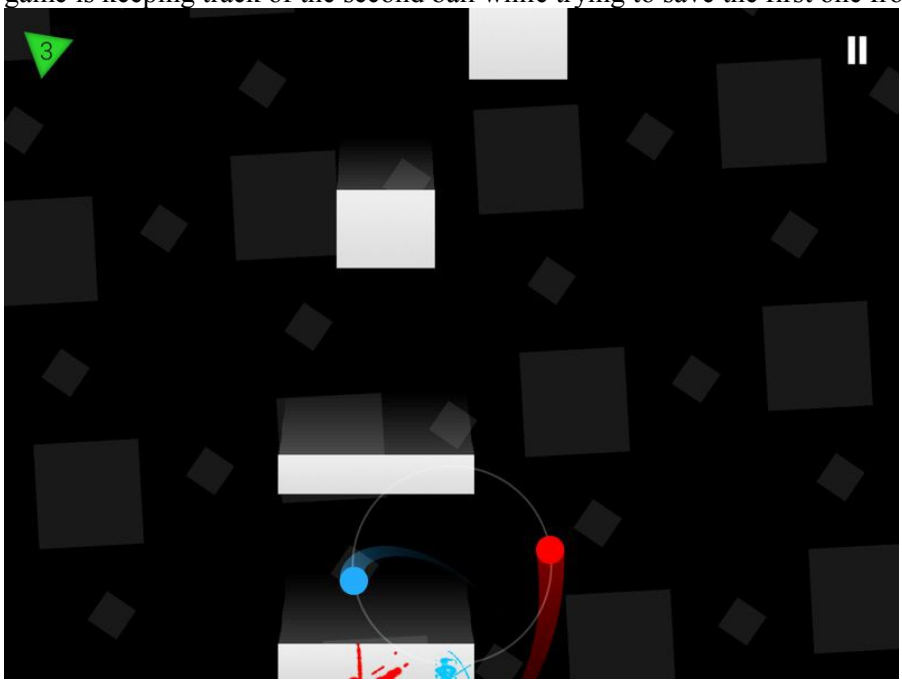

Figure 1: Original Game

While Duet is a challenging game and enjoyable to play, it utilizes only rectangles as obstacles coming from the top edge of the screen but from nowhere else. From our point of view these limitations are unnecessary and reduce the variety and surprise element in the game. Therefore we decided to get rid of these limitations and add different geometrical shapes, rotations and trajectories to our game Dipole.

Obstacles

We decided to create four different geometric obstacles with different properties, which are ellipse, rectangle, triangle and octagon. All of these obstacles have the same vertical velocity named as gravitational constant but different properties otherwise.
Ellipse is possibly the hardest obstacle in our game. It always comes from the top edge and has a changing speed in the horizontal direction. It also bounces back and forth between two vertical lines, which the player should adjust before it comes to the rotating dipole. Passing the ellipse requires careful planning and fast reflexes.

Figure 2: Ellipse obstacle

The rectangle is not like the rectangles in the Duet game. It is rotating in a fast pace, can come from the sidewalls, and has a constant horizontal velocity.

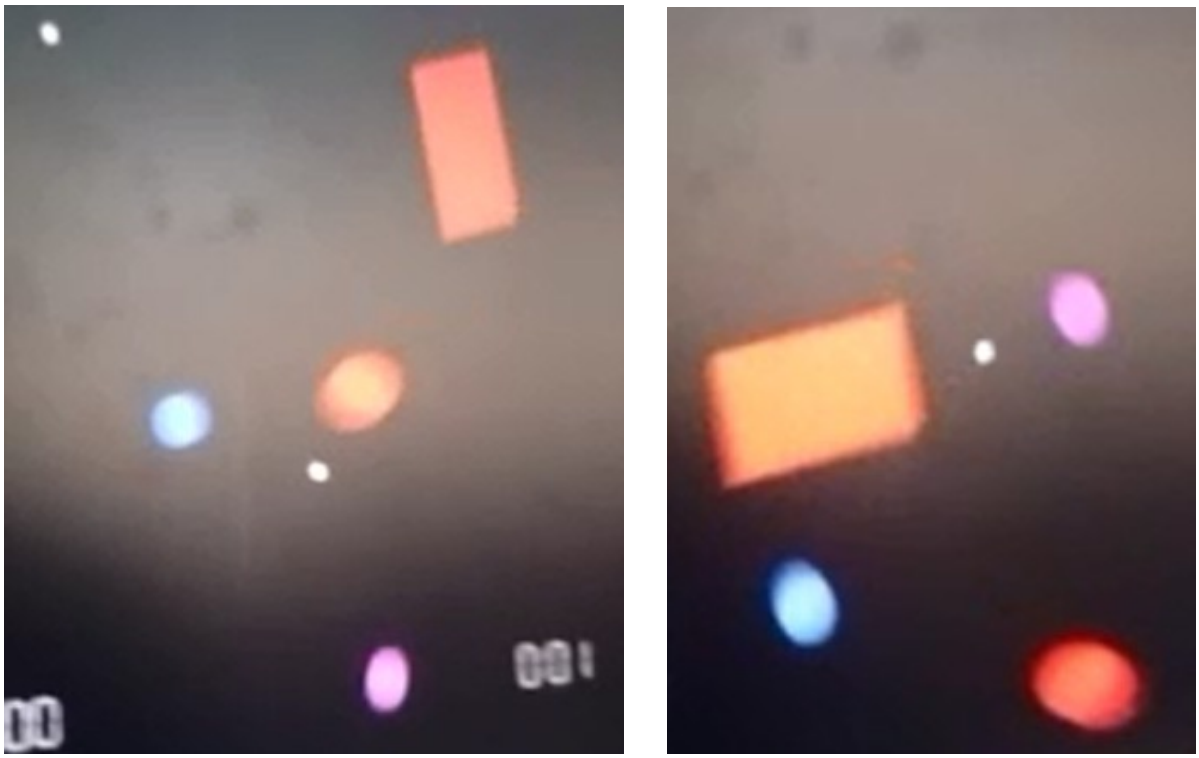

Figure 3: Rectangle obstacle

The triangles in our game have the same properties as the rectangles but they move in the opposite horizontal direction.

Figure 4: Triangle obstacle

The octagon is the largest obstacle we have. It comes from the top edge, it rotates but it does not have any horizontal velocity.

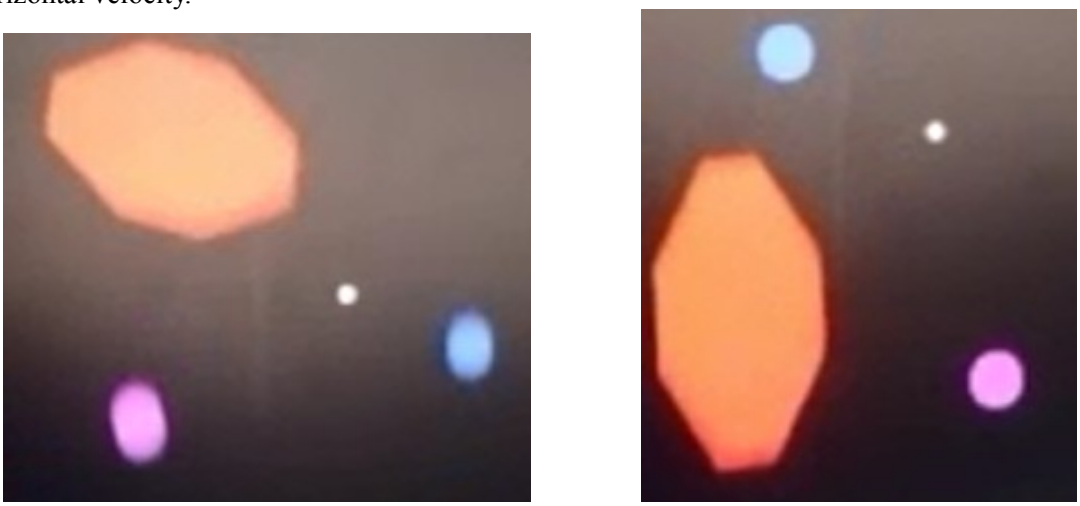

Figure 5: Octagon obstacle

Levels

The speed of the rotations (including the rotation of the dipole), horizontal velocities and the value of gravitational constant increase each minute. Therefore, a player should check the clock on the screen in order to be aware of possible changes of gameplay.

Theme

We decided to create a space theme to our game. For this purpose we created a dark background, representing empty space with several stars. In addition to the visual graphics, we added the soundtrack of the “Interstellar” by using two speakers to create a polyphonic melody. The choice of game music also supports the theme of the game.

Reset, Pause and Ending

Our game Dipole also have a pause button in order to give a player time to rest, or a chance to look at his phone without losing the game. The pause button is also useful when making a planning when the player is

in a difficult position in the game. In order to tell player how to improve and what did he did wrong, when the player loses the crash area is highlighted with green. Also initials “GG” is written on the screen, as shorthand for “a good game”.

Implementation

Clock and Note Generation

We needed several different clocks for realizing different tasks; a 25 MHz clock for VGA, a 1Hz clock for counting the time, a 125 Hz clock for controlling rotating objects and a 62.5 Hz clock for controlling the dipole. In addition to these clocks we needed to generate 16 different notes for creating the “Interstellar” soundtrack. These notes start from C of the 4^{th} octave and goes until the D of the 6^{th} octave. Since we need to generate so many different frequencies we found a general way of making frequencies by using 50 MHz clock input of the FPGA.
For this purpose we made a counter that counts until the half of the value you want to divide the 50 MHz clock and reset to zero when reaches that value. Then the generated clock will be set to its inverse, if the counter is equal to its highest value, zero otherwise. By just defining different maximums to the counter it is possible to create different frequency clocks. Here is the code for generating any frequency which is a dividend of 50 MHz clock.

```
if (count1 == const1 - 1)
      count1 <= 0;
   else
      count1 <= count1 + 1;
if (count1 == const1 - 1)
      clk_25M <= ~clk_25M;
   else
      clk_25M <= clk_25M;
```

Pause

For pausing the game we assign one of the switchs as a pause input. We state the conditions for always loops such that loop is performed when pause input is low. When it is high everything freezes except vga and clock generation parts.

Dipole generation

For the rotation of dipoles by user command, two push buttons are assigned; one for rotating it in clockwise direction, one for rotating in counterclockwise direction. Then the centre of the complete dipole system is defined. It is decided to define center of each dipole by help of the radius of the system, angles of the lines passing through the system centre and one dipole centre (actually there are only one line but we considered as there are two angles which are 180° apart from each other) and the sine and cosine values of that angles. For this purpose we generated the sine and cosine values of each angle which is an integer in degree scale with a basic MATLAB code. Then prepared a task that takes the magnitude of angle and sends the sine and cosine values. When a push button is pressed the angles are incremented/decremented by a specific amount. Center of each dipole is calculated and if $(x-x_c)^2 + (y-y_c)^2 < r^2$, the point (x,y) is inside the dipole circle where r is the radius of a dipole, x_c,y_c are the coordinates of the center of dipole.

The sine/cosine task is used in different parts of the project where rotation is performed.

Moving and Rotating Object generation

- **Moving and Rotating Rectangle**

Generating a non-rotating rectangle is easy, it is needed to specify that if x and y values of the pixel are between specific values make the pixel red. However, for rotating objects different approach should be applied. Our first idea was defining all the points of sides of rectangle and if a pixel passes one side make all the pixels red starting from that pixel to the pixel which passes another side again. This algorithm works fine for rotating rectangles except they are aligned with the horizontal or vertical lines. At these moments boundaries of rectangles start to change colour because consecutive pixels are on a side continuously. Our second idea was to question each pixel whether it is inside the rectangle by analytical geometry. If the sum of all the areas of triangles formed by a point and two consecutive vertices of rectangle is greater than the area of rectangle, then that point is outside of the rectangle, if it is equal it is inside the rectangle. Therefore with this implemetation we need to define only four vertices of rectangle; but there is an important computation for the hardware. For the gravitational fall of objects we decided to define a center of rectangle and increment the y value of it by some amount in each clock cycle. Vertices are defined by help of pre-defined sine and cosine values, diagonal distance between vertices and center and the angles of the lines passing through the center and each vertex with respect to horizontal axis. After we added a speed component in x-direction by incrementing/decrementing the x value of the center by some amount in each clock cycle.

The advantage of this approach is that it can be used for any polygon, and it is used for generating different objects in our projects as explained below.

- **Moving and Rotating Equilateral Triangle**

Everyting is same except there are three vertices.

- **Moving and Rotating Isogonal Octagon**

This shape is obtained by accident when we were trying to obtain rectangle. The area of the rectangle is defined as the twice of the exact value. Then such a shape is formed. We decided to retain this shape. The procedure is same as rectangle; but even it has eight vertices only four vertices are used to define the shape. In this way computational load to the hardware become less than normal.

- **Bouncing Ellipse**

A center is defined and its y component is incremented in each cycle. For the bouncing effect, a signed variable is defined and added to the x component of the center of the ellipse in each cycle. When the center of ellipse comes to one of the two pre-defined x coordinates, the sign of the variable is changed. For the generating the shape of ellipse, the analytic formula of ellipse is used. If $(x-x_c)^2/a + (y-y_c)^2 < r^2$, the point (x,y) is inside the ellipse where r is the secondary radius of ellipse, x_c,y_c are the coordinates of the center of ellipse, a is a parameter for the eccentricity of ellipse.

Triangle Area calculation

If coordinates of all three vertices of a triangle are known its area can be found by the formula

$| (x_1-x_3)*(y_2-y_1)- (x_1-x_2)*(y_3-y_1) |$

For this calculation absolute value operation is necessary. Therefore we defined a task for this purpose. It checks whether a signed variable is less than 0, if so it takes its 2's complement by taking its bitwise inverse (1's complement) and adding 1.

Score counter

In our game the score increases by 1 if an object exits the screen without a collision. For this purpose our code checked in each cycle whether the center of any object is at the vertical level of below 480^{th} pixel. However when the gravity increases at every minute mark, a chance of passing the 480^{th} pixel level without crossing it i.e. going directly 479^{th} pixel to 481^{st} pixel. We should have define an interval but without any supplementation to this method score will continue increasing during the entrance and exit of an object. Therefore extra variables for each object are defined. If an object is shown at the screen, corresponding variable is set to 0, when it enters the interval for scoring, score is increased by one if the variable is 0 the the variable is set to 1. By this method correct scoring is achieved.

Timer

Two variables for seconds and minutes are defined. A 1 Hz clock is generated and in each clock cycle seconds are incremented by 1. Then it is checked whether the seconds variable equal to 60, if so seconds variable is set to 0 and minutes variable is incremented by 1.

Seven segment displays for score and timer

Seven segment displays are defined by defining each line of seven segment display (i.e. y=a , b<x<c or x=a, b<y<c) and defining the condition when it will become active (i.e. the top segment will be active for the numbers 0,2,3,5,6,7,8,9). Score and seconds are seperated to their digits and each digit is set to a seven segment display.

Collision check

Three variables k, k2 and k3 are defined. As explained above in each cycle we check wheter the pixel is inside an object or a dipole. If it is inside a dipole k is set to 1, if it is inside an object k2 is set to 1. k3 is the logical AND of k and k2. When a collision happens, in other words a pixel is both inside a dipole and an object, both k and k2 are 1 then k3 is 1. The other parts of the code questions whether a collision happened or not by checking the value of k3. Also for proofing the collision to the gamer, the pixels corresponding to the collision are made green. In addition a “GG” (abbreviation of good game which is used in most of multiplayer games to indicate one side surrenders and game ends) appears on the screen when a collision happens. GG is formed by defining each line of the letter. (i.e. a<x<b , c<y<d)

VGA

The VGA (Video Graphics Array) interface is used for displaying the information on screen. A VGA monitor uses a 25 MHz clock, therefore first we generated it. With each clock pulse, a new pixel on the monitor will be displayed. In addition to red, green and blue signals that decide the color of each of these pixels, we need two synchronization signals that will arrange the changes in lines and columns. These signals are called “hsync” and “vsync” signals and should be active only in some clock pulses that are not a part of display. In order to arrange the screen, we first created two counters, one for vertical axis and one for horizontal. With each clock pulse, the count for horizontal axis incremented by one and by each 800 clock pulses (end of horizontal line) and at each 800 pulses the count for vertical axis are incremented by one.

Symbol	Parameter	Vertical Sync			Horizontal Sync	
		Time	Clocks	Lines	Time	Clocks
T_S	Sync pulse time	16.7 ms	416,800	521	32 µs	800
T_{DISP}	Display time	15.36 ms	384,000	480	25.6 µs	640
T_{PW}	Pulse width	64 µs	1,600	2	3.84 µs	96
T_{FP}	Front porch	320 µs	8,000	10	640 ns	16
T_{BP}	Back porch	928 µs	23,200	29	1.92 µs	48

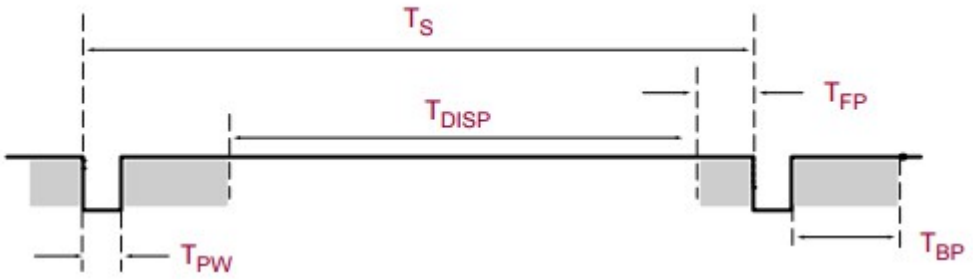

Figure 6: VGA timings

As can be seen from the figure above, both the vertical sync the horizontal sync signal should be low only during the T_{PW} and this can be done by setting the vertical sync to 0 when vertical count is lower than 2, and horizontal sync to zero only when horizontal count is lower than 96. After that all values for red, green and blue signals outside the display time should be zero. This is implemented by a simple if condition. During the display time, simply the conditions for the colors are given by the game logic.

Music

We defined 16 notes for music as clocks in specific frequencies (i.e. 440 Hz is A (La in Turkish)). Main theme of the film Interstellar is chosen as the background music of our game because of the space concept. The music is played in two different channels as stereo. Outputs are specified for each duration of notes.

```
..
else if(sure<400000000)
begin
ses1=mi;
ses2=la;
end
..
```

Reset

A push button is assigned for reseting the game. When that button is used all the variables are set to their initial conditions unless game is paused.

Test

Dipole

When testing the dipole the only problem we encountered is that there happens some jumps in the angle of dipoles. After doing different functional simulations in quartus software we found the problem. When the value of the angle is 0 and we try to decrement it, an overflow happens and its value becomes 511 in 8 bit registers. We solved the problem as when angle is 0 and we try to decrement it we assign it the value 359. Same problem appeared when we decided to increase the rotation speed of dipole in every minute mark. This time jump is happening when the angle is 1 too. We specified the cases for assigning 359 for 4 different speeds. It is found to be necessary since during tests no one could play more than 1 minute and 32 seconds which corresponds to only 1 unit speed increase.

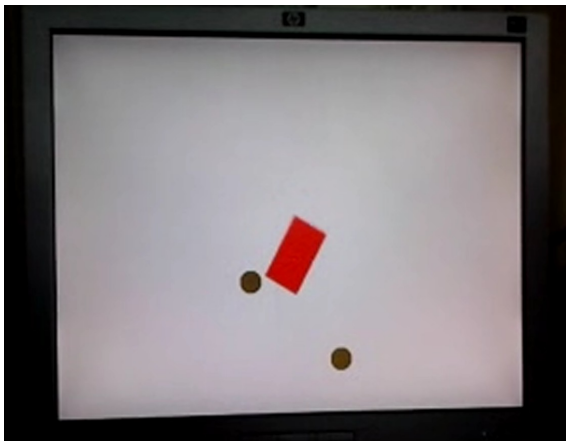

Figure 7: First prototype of rotating rectangle and dipoles

Rotating objects

Our main aim in this project is to generate a rotating rectangle. As expained above we first tried to deifne all the points of the sides of rectangle, and determine the color of a pixel depending on the informations if it is on top of a side and if the previous pixel was inside the rectangle. When we tested this method we found that when a side becomes parallel to horizontal axis, there is a series of pixels standing on a side. Therefore colors of pixels start to alternate. We decided that this method is not appropriate and we tried our second method depending on the areas. When we were trying to obtain a rectangle accidently (such accidents have important roles in the development of science)we obtain an isogonal octagon. We had calculated the area of rectangle twice of its exact value. This error invoked a new object. We decided to use this object too.

Playibility of the game

A gamer should not lose a game he/she played it perfectly. Therefore we should design our game such that there is always at least one way for tackling incoming objects. For this purpose we continuously played the game and try to progress through it. For testing deeper levels we turned off the collision in our code for not losing the game because of slight collisions. Game seemed as playable but for being on the safe side we increase the rotation speed of dipoles which increases the difficulty but for players with good eye-hand coordination this change decreases the chance of an impossible game.

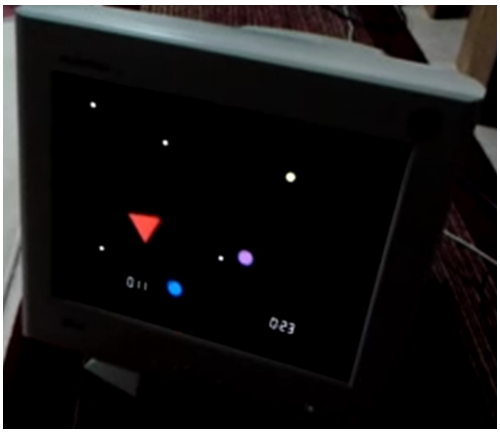

Figure 8: Screenshot of the game during tests

Biosketches

Doğa Veske

WORK EXPERIENCE

16/06/2014-11/07/2014	Internship TAI-Turkish Aerospace Industries, Inc., Ankara (Turkey) Avionics and Hardware Departments *Gained knowledge about databuses in aircrafts,* *Programmed STM32F4 board*

EDUCATION

09/2013-Present	BSc. in Physics (Double Major) Middle East Technical University, Ankara (Turkey) Advanced Program (Honors degree), CGPA: 3.76/4.00
09/2012-Present	BSc. in Electrical and Elctronics Engineering (Major) Middle East Technical University, Ankara (Turkey) CGPA: 3.66/4.00
09/2008-06/2012	High School TED Ankara College Foundation High School, Ankara (Turkey) Scholar Development Program

LANGUAGE SKILLS

Mother Tongue	Turkish
Other Languages:	English: Advanced German: Intermediate

COMPUTING SKILLS

OS:	Windows, Linux
Programming Languages:	C, MATLAB/Octave
Simulation Programs:	LTSPICE, Multisim, Electronics Workbench, COMSOL Multiphysics
CAD:	Keycreator
Test-Measurement:	Agilent-VEE

Other: Office tools, Photoshop, common Windows applications

PROJECTS

- 2015-Present, CANSAT(International competition for building a vehicle that is launched with a rocket and expected to accomplish pre-defined tasks)
- 2015, AM radio construction with automatic gain control
- 2014, Audio frequency analyzer construction
- 2014, Speed and direction control of a DC motor with pulse width modulation (PWM) depending on the temperature difference of two media
- 2013-Present, Spark chamber construction for muon detection
- 2011, TÜBİTAK (The Scientific and Technological Research Council of Turkey) High School Project Contest participation with "*Hydrogen Generation with Accumulated Static Electric on Aircraft Exterior Surfaces"* project

HONORS AND AWARDS

- 2014-Present, Scholarship from TÜBİTAK (The Scientific and Technological Research Council of Turkey) National Scholarship Program
- 2012, Turkey Licence Placement Exam (LYS) degree 303rd out of 2.137.000
- 2011, Ankara Junior Chess Championship Age Group 17, 2nd position
- 2011, Ankara Atatürk High School Team Chess Tournament 1st position as school chess team
- 2009, International Regions Mathematics League 9th place in world as school team
- 2008-2012, Full scholarship during high school education from TED Ankara College Foundation
- 2008, Turkey High School Student Selection and Placement Examn (OKS) 802nd degree out of 913.000
- 2008, European Council of International Schools (ECIS) Mathematics Competition Achievement

LICENCES

- 2013, Horse riding and jumping licence

Baran Bodur

Education:

2012 – Present: Middle East Technical University, Ankara (Turkey)
BSc. Electrical-Electronic Engineering
CGPA: 3.87

2012 – Present: Middle East Technical University, Ankara (Turkey)
BSc. Physics
CGPA: 3.86

2009-2012: TED Ankara College High School, Ankara (Turkey)
Scholar Development Program

Internships:

2014: TÜBİTAK UZAY, Ankara
Satellite Technologies Department, Ground Support Systems Group
Modeling antenna's motion with respect to Rasat Satellite

Language Skills:

English: Fluent, IELTS Score 7.0
French: Elementary, DELF A1 Certificate

Computer Skills:

Accustomed to LINUX OS
Good knowledge of Office Programs
Competent in MATLAB /Octave and C programming languages
Able to use COMSOL, LTSpice, Multisim, KeyCreator, Agilent VEE, Quartus 2 Softwares

Projects:

2015 Design of an AM Radio with automatic gain controller
2014 – Present International CANSAT project (member of METUSAT team)
2014 Design of a frequency analyzer
2013 – Present Construction of a muon detecting spark chamber
2013 Design of a temperature controlled, motor driving circuit using PWM (without transistors or processors)
2011 TÜBİTAK Project Competition for high school students (Usage of an ultrasonic sensor and a laser distance sensor in order to provide information about surroundings and stairs to visually impaired people)

Additional Achievements:

Bülent Kerim Altay Award due to academic success
Translation of "A Mathematical Theory of Communication" and "A Symbolic Analysis of Switching and Relay Circuit" by Claude Shannon
Multiple degrees in nation-wide chess championships (both team and individual)

Conclusion

In this project, we learned the difficulties and advantages of hardware programming and sharpened our mathematical and algorithmic skills.
We noticed that hardware programming is much harder than the software programming since Verilog is a lower level of programming language and we lack the usual toolbox that we used to have. This created many unexpected problems, like being unable to use sine and cosines without defining them or even being unable to define them since hardware cannot represent floating numbers. However, we found our way around with our new toolbox, such as always blocks, parallel operations and bit shifting. In addition, we also noticed that hardware can perform operations much faster than software, since it can run different operations in parallel given enough logic blocks. It is also more flexible, one can define a number as a combination of any number of bits as necessary, avoiding losing memory for not used parts.
In addition to that, this project really encouraged us to use our trigonometry and analytical geometry skills for all rotations and visualizations. Since VGA is operating in coordinates, we basically used analytical geometry expressions to define which parts should be which colors. Therefore we noticed how different areas of mathematics are necessary for programming algorithms. In our case, we saved a lot of memory by only defining the corners and centers of our geometrical shapes and finding the next positions of the shape and visualizing it just from that information.
All in all, we created several different shapes that rotate and move in all directions with a limited toolbox. Our design of geometrical shapes can be used in a better order for a more advanced game. The rotation and visualization algorithms can be used in other projects that utilize Verilog or even high level programming languages.
Making a game as the project allowed us to be creative and putting ourselves in place of the player. We did not just program an FPGA for a specific task, but tried to build a better and a more enjoyable game using all the tools we have.
In conclusion this was a very enjoyable project that taught us the new concept of hardware programming and forced us to be creative.

References

- http://en.wikipedia.org/wiki/Scientific_pitch_notation
  // for notes
- https://musescore.com/user/164942/scores/428696
  // interstellar theme
- http://www.xilinx.com/support/documentation/boards_and_kits/ug130.pdf
  //for VGA
- http://math.stackexchange.com/questions/190111/how-to-check-if-a-point-is-inside-a-rectangle
  // for game
- http://www.asic-world.com/verilog/veritut.html
  //for verilog syntax
- ftp.altera.com/up/pub/Altera_Material/13.1/Boards/DE1-SoC/DE1_SoC_User_Manual.pdf
  //user manual

Appendix 1

MATLAB code for generating sine table (similarly a cosine table is generated)

```
fileID = fopen('sin.txt','w');
for i=0:1:360
   W=[i,sind(i)*(2^10)];
   fprintf(fileID,'sin[%d]=%.0f;\n',W);
end
fclose(fileID);
```

Appendix 2

Game code

```
module dipole2 (pause, clk_50M, clk_25M, clk_50, duz, ters, r, g, b, hsig, vsig, reset,ss0,ss1,ss2,ss3,ss4,ss5,ss6,ses1,ses2);//, sinout1, cosout1, merk1y_up, theta, merk1x_up);



output ses1;
output ses2;



//controls
input clk_50M;
input duz;
input ters;
input reset;

input pause;


//game clock
output clk_50;

//for vga
output clk_25M;
output [7:0] r;
output [7:0] g;
output [7:0] b;
output hsig;
output vsig;


//seven segment

output ss0;
output ss1;
output ss2;
output ss3;
output ss4;
output ss5;
output ss6;


//çikacak
//
//output [63:0] sinout1;
//output [63:0] cosout1;
//output [63:0] theta;
//output [63:0] merk1y_up;
//output [63:0] merk1x_up;
```

```
reg  [20:0] countdo1;
reg  [20:0] countre;
reg  [20:0] countmi;
reg  [20:0] countfa;
reg  [20:0] countsol;
reg  [20:0] countla;
reg  [20:0] countsi;
reg  [20:0] countdoince;
reg  [20:0] countreince;
reg  [20:0] countmince;
reg  [20:0] countfaince;
reg  [20:0] countsolince;
reg  [20:0] countlaince;
reg  [20:0] countsince;
reg  [20:0] countdoince2;
reg  [20:0] countreince2;



reg  [40:0] sure;


reg do1;
reg re;
reg mi;
reg fa;
reg sol;
reg la;
reg si;
reg doince;
reg reince;
reg mince;
reg faince;
reg solince;
reg laince;
reg since;
reg doince2;
reg reince2;

reg ses1;
reg ses2;




// for vga

reg [9:0] vs;
reg [9:0] hs;
reg vsig;
reg hsig;
reg [7:0] r;
reg [7:0] g;
```

```
reg [7:0] b;
reg k;
reg k2;
reg [31:0] countg;
reg [5:0] saniye;
reg [3:0] dakika;
reg [3:0] saniye1;
reg [3:0] saniye2;



//clocks and counting
reg clk_25M;
reg clk_50;
reg timer;
integer count1;
integer count2;
integer count3;
integer count4;
reg clk2;

// For rectengles

//reg [9:0] i; // increment value
reg [5:0] gc; //gravitational constant
//reg [9:0] h; // rotation constant

//reg [9:0]  kx11[0:100];
//reg [9:0]  ky11[0:100];
//reg [9:0]  kx12[0:50];
//reg [9:0]  ky12[0:50];
//reg [9:0]  kx13[0:100];
//reg [9:0]  ky13[0:100];
//reg [9:0]  kx14[0:50];
//reg [9:0]  ky14[0:50];
//reg [9:0] mx1;
//reg [9:0] my1;
//reg d1;
//reg d2;
//reg d3;
reg [31:0] sinout1;
reg [31:0] cosout1;
reg [31:0] sinout2;
reg [31:0] cosout2;
reg [31:0] sinout;
reg [31:0] cosout;

reg [11:0] k1x;
reg [11:0] k2x;
reg [11:0] k3x;
reg [11:0] k4x;
reg [11:0] k1y;
reg [11:0] k2y;
reg [11:0] k3y;
```

```
reg [11:0] k4y;

reg [9:0] theta3;
reg [9:0] theta4;
reg [9:0] theta5;
reg [9:0] theta6;
reg [6:0] rad;

reg [9:0] mx;
reg [9:0] my;

reg [2:0] donk;

reg signed [25:0] c2adik;
reg signed [25:0] a1;
reg signed [25:0] a2;
reg signed [25:0] a3;
reg signed [25:0] a4;

reg k3;


//for dipole

reg [11:0] merk1y;
reg [11:0] merk1x;
reg [11:0] merk2y;
reg [11:0] merk2x;
reg [11:0] merkx;
reg [11:0] merky;
reg [11:0] yark;
//reg [15:0] donk;

reg [11:0] merk1y_up;
reg [11:0] merk1x_up;
reg [11:0] merk2y_up;
reg [11:0] merk2x_up;
reg [9:0] theta;
reg [9:0] theta2;

//ucgen

reg [11:0] k1xu;
reg [11:0] k2xu;
reg [11:0] k3xu;
reg [11:0] k4xu;
reg [11:0] k1yu;
reg [11:0] k2yu;
reg [11:0] k3yu;
reg [11:0] k4yu;

reg [9:0] theta3u;
reg [9:0] theta4u;
reg [9:0] theta5u;
reg [9:0] theta6u;
```

```
reg [6:0] radu;

reg [9:0] mxu;
reg [9:0] myu;

reg [2:0] hizu;

reg signed [25:0] c2aucgen;
reg signed [25:0] u1;
reg signed [25:0] u2;
reg signed [25:0] u3;

//daire


reg signed [9:0] merkxd;
reg signed [9:0] merkyd;
reg [9:0] yarkd;
reg signed [4:0] hizx;

//yamuk sekizgen

reg [11:0] k5x;
reg [11:0] k6x;
reg [11:0] k7x;
reg [11:0] k8x;
reg [11:0] k5y;
reg [11:0] k6y;
reg [11:0] k7y;
reg [11:0] k8y;

reg [9:0] theta7;
reg [9:0] theta8;
reg [9:0] theta9;
reg [9:0] theta10;
reg [6:0] rad2;

reg [3:0] donk2;

reg [9:0] mx2;
reg [9:0] my2;


reg signed [25:0] c2adik2;
reg signed [25:0] a5;
reg signed [25:0] a6;
reg signed [25:0] a7;
reg signed [25:0] a8;

reg [6:0] score;

reg ss0;
reg ss1;
reg ss2;
reg ss3;
```

```
reg ss4;
reg ss5;
reg ss6;

reg [6:0] score1;
reg [6:0] score2;
reg [6:0] score3;

reg st1;
reg st2;
reg st3;
reg st4;

//for pause

//reg signed [4:0] hizx_2;
//reg [5:0] gc_2;
//reg [2:0] donk_2;
//reg [3:0] donk2_2;
//reg [2:0] hizu_2;


// initialization

initial

begin

countdo1=0;
countre=0;
countmi=0;
countfa=0;
countsol=0;
countla=0;
countsi=0;
countdoince=0;
countreince=0;
countmince=0;
countfaince=0;
countsolince=0;
countlaince=0;
countsince=0;
countdoince2=0;
countreince2=0;

sure=0;
ses1=0;
ses2=0;

count2=0;


saniye=0;
dakika=0;
// for vga
```

```
countg=0;
r=0;
g=0;
b=0;
vs=0;
hs=0;
vsig=0;
hsig=0;

//for clocking
count1=0;
count2=0;

// for rectangles

//k1x=200;
//k2x=300;
//k3x=300;
//k4x=200;
//k1y=150;
//k2y=150;
//k3y=250;
//k4y=250;

theta3=30;

rad=50;

mx=600;
my=1000;

donk=1;

c2adik=8760; //aslen 8660

a1=0;
a2=0;
a3=0;
a4=0;
gc=2;


//mx1=150;
//my1=55;
//d3=0;
//for(i=0;i<101;i=i+1)
//begin
//kx11[i]=i+100;
//kx13[i]=i+100;
//ky11[i]=20;
//ky13[i]=70;
//end
//for(i=0;i<51;i=i+1)
//begin
//ky12[i]=i+20;
```

```
//ky14[i]=i+20;
//kx12[i]=100;
//kx14[i]=200;
//end

// for dipole

merkx=320;
merky=340;
merk1y=340;
merk1x=320+90;
merk2y=340;
merk2x=320-90;
//donk=1;
theta=0;
yark=13;

merk1x_up=merk1x;
merk1y_up=merk1y;
merk2x_up=merk2x;
merk2y_up=merk2y;
//ucgen

theta3u=0;
radu=40;
hizu=2;
//donku=1;
c2aucgen=4224;
u1=0;
u2=0;
u3=0;
mxu=400;
myu=500;

//daire

yarkd=16;
merkxd=200;
merkyd=100;
hizx=4;

//yamuk sekizgen

theta7=30;

rad2=25;

mx2=250;
my2=760;



c2adik2=4330; //aslen 8660

a5=0;
```

```
a6=0;
a7=0;
a7=0;

donk2=1;

score=0;

st1=0;
st2=0;
st3=0;
st4=0;

end

// Defining parameters

localparam const1=1, const2=400000, const3=25000000 , const4=200000;

localparam do1262=95566, re294=85150, mi330=75758, fa350=71429, sol392=63776, la440=56819, si494=50607, doince523=47801, reince587=42589, mince660=37879, faince699=35791, solince784=31888, laince880=28409, since988=25309, doince21047=23889, reince21175=21277;


//Generating necessary clock signals

always@(posedge clk_50M)

begin

//Count up to 1
   if (count1 == const1 - 1)
      count1 <= 0;
   else
      count1 <= count1 + 1;

//count up to 999999

            if (count2 == const2 - 1)
      count2 <= 0;
   else
      count2 <= count2 + 1;

//count timer

 if (count3 == const3 - 1)
      count3 <= 0;
   else
      count3 <= count3 + 1;

//

if (count4 == const4 - 1)
      count4 <= 0;
   else
```

```
        count4 <= count4 + 1;


// generate 25 MHz clock

            if (count1 == const1 - 1)
        clk_25M <= ~clk_25M;
    else
        clk_25M <= clk_25M;

// generate 50 Hz clock

            if (count2 == const2 - 1)
        clk_50 <= ~clk_50;
    else
        clk_50 <= clk_50;

//generate timer

if (count3 == const3 - 1)
        timer <= ~timer;
    else
        timer <= timer;

//

if (count4 == const4 - 1)
        clk2 <= ~clk2;
    else
        clk2 <= clk2;

//Count for do1
    if (countdo1 == do1262 - 1)
        countdo1 <= 0;
    else
        countdo1 <= countdo1 + 1;

//Count for re
    if (countre == re294 - 1)
        countre <= 0;
    else
        countre <= countre + 1;

//Count for mi
    if (countmi == mi330 - 1)
        countmi <= 0;
    else
        countmi <= countmi + 1;

//Count for fa
    if (countfa == fa350 - 1)
        countfa <= 0;
    else
        countfa <= countfa + 1;
```

```
//Count for sol
   if (countsol == sol392- 1)
      countsol <= 0;
   else
      countsol <= countsol + 1;

//Count for la
   if (countla == la440 - 1)
      countla <= 0;
   else
      countla <= countla + 1;

//Count for si
   if (countsi == si494 - 1)
      countsi <= 0;
   else
      countsi <= countsi + 1;

//Count for doince
   if (countdoince == doince523 - 1)
      countdoince <= 0;
   else
      countdoince <= countdoince + 1;

//Count for reince
   if (countreince == reince587 - 1)
      countreince <= 0;
   else
      countreince <= countreince + 1;

//Count for mince
   if (countmince == mince660 - 1)
      countmince <= 0;
   else
      countmince <= countmince + 1;

//Count for faince
   if (countfaince == faince699 - 1)
      countfaince <= 0;
   else
      countfaince <= countfaince + 1;

//Count for solince
   if (countsolince == solince784 - 1)
      countsolince <= 0;
   else
      countsolince <= countsolince + 1;

//Count for laince
   if (countlaince == laince880 - 1)
      countlaince <= 0;
   else
      countlaince <= countlaince + 1;

//Count for since
```

```
    if (countsince == since988 - 1)
       countsince <= 0;
    else
       countsince <= countsince + 1;

//Count for doince2
    if (countdoince2 == doince21047 - 1)
       countdoince2 <= 0;
    else
       countdoince2<= countdoince2 + 1;

//Count for reince2
    if (countreince2 == reince21175 - 1)
       countreince2 <= 0;
    else
       countreince2<= countreince2 + 1;


// generate do

           if (countdo1 == do1262- 1)
       do1 <= ~do1;
    else
       do1<= do1;

// generate re

           if (countre == re294 - 1)
       re <= ~re;
    else
       re<= re;

// generate mi

           if (countmi == mi330 - 1)
       mi <= ~mi;
    else
       mi<= mi;

// generate fa

           if (countfa == fa350 - 1)
       fa <= ~fa;
    else
       fa<= fa;

// generate sol

           if (countsol == sol392 - 1)
       sol <= ~sol;
    else
       sol<= sol;

// generate la
```

```
            if (countla == la440 - 1)
        la <= ~la;
    else
        la<= la;

// generate si

            if (countsi == si494 - 1)
        si <= ~si;
    else
        si<= si;

// generate doince

            if (countdoince == doince523 - 1)
        doince <= ~doince;
    else
        doince<= doince;

// generate reince

            if (countreince == reince587 - 1)
        reince <= ~reince;
    else
        reince<= reince;


// generate mince

            if (countmince == mince660 - 1)
        mince <= ~mince;
    else
        mince<= mince;

// generate faince

            if (countfaince == faince699 - 1)
        faince <= ~faince;
    else
        faince<= faince;

// generate solince

            if (countsolince == solince784 - 1)
        solince <= ~solince;
    else
        solince<= solince;

// generate laince

            if (countlaince == laince880 - 1)
        laince <= ~laince;
    else
        laince<= laince;
```

```
// generate since

            if (countsince == since988 - 1)
        since <= ~since;
    else
        since<= since;

// generate doince2

            if (countdoince2 == doince21047 - 1)
        doince2 <= ~doince2;
    else
        doince2<= doince2;

// generate reince2

            if (countreince2 == reince21175 - 1)
        reince2 <= ~reince2;
    else
        reince2<= reince2;

//lets make some music


sure=sure+1;
if(sure>40'd5856000000)
sure=0;


//1
if(sure<16000000)
ses1=mi;
else if(sure<32000000)
ses1=do1;
else if(sure<48000000)
ses1=mi;
else if(sure<64000000)
ses1=do1;
else if(sure<80000000)
ses1=mi;
else if(sure<96000000)
ses1=do1;
//2
else if(sure<112000000)
ses1=mi;
else if(sure<128000000)
ses1=do1;
else if(sure<144000000)
ses1=mi;
else if(sure<160000000)
ses1=do1;
else if(sure<176000000)
ses1=mi;
else if(sure<192000000)
ses1=do1;
```

```
//3
else if(sure<208000000)
ses1=mi;
else if(sure<224000000)
ses1=do1;
else if(sure<240000000)
ses1=mi;
else if(sure<256000000)
ses1=do1;
else if(sure<272000000)
ses1=mi;
else if(sure<288000000)
ses1=do1;
//4
else if(sure<304000000)
ses1=mi;
else if(sure<320000000)
ses1=do1;
else if(sure<336000000)
ses1=mi;
else if(sure<352000000)
ses1=do1;
else if(sure<368000000)
ses1=mi;
else if(sure<384000000)
ses1=do1;
//5
else if(sure<400000000)
begin
ses1=mi;
ses2=la;
end
else if(sure<416000000)
begin
ses1=do1;
ses2=la;
end
else if(sure<432000000)
begin
ses1=mi;
ses2=la;
end
else if(sure<448000000)
begin
ses1=do1;
ses2=la;
end
else if(sure<464000000)
begin
ses1=mi;
ses2=la;
end
else if(sure<480000000)
begin
ses1=do1;
```

```
ses2=la;
end
//6
else if(sure<496000000)
begin
ses1=mi;
ses2=si;
end
else if(sure<512000000)
begin
ses1=re;
ses2=si;
end
else if(sure<528000000)
begin
ses1=mi;
ses2=si;
end
else if(sure<544000000)
begin
ses1=re;
ses2=si;
end
else if(sure<560000000)
begin
ses1=mi;
ses2=si;
end
else if(sure<576000000)
begin
ses1=re;
ses2=si;
end
//7
else if(sure<592000000)
begin
ses1=mi;
ses2=0;
end
else if(sure<608000000)
begin
ses1=re;
ses2=0;
end
else if(sure<624000000)
begin
ses1=mi;
ses2=0;
end
else if(sure<640000000)
begin
ses1=re;
ses2=0;
end
else if(sure<656000000)
```

```
begin
ses1=mi;
ses2=0;
end
else if(sure<672000000)
begin
ses1=re;
ses2=0;
end
//8
else if(sure<688000000)
begin
ses1=mi;
ses2=la;
end
else if(sure<704000000)
begin
ses1=re;
ses2=la;
end
else if(sure<720000000)
begin
ses1=mi;
ses2=si;
end
else if(sure<736000000)
begin
ses1=re;
ses2=si;
end
else if(sure<752000000)
begin
ses1=mi;
ses2=doince;
end
else if(sure<768000000)
begin
ses1=re;
ses2=doince;
end
//9
else if(sure<784000000)
begin
ses1=mi;
ses2=si;
end
else if(sure<800000000)
begin
ses1=re;
ses2=si;
end
else if(sure<816000000)
begin
ses1=mi;
ses2=la;
```

```
end
else if(sure<832000000)
begin
ses1=re;
ses2=la;
end
else if(sure<848000000)
begin
ses1=mi;
ses2=si;
end
else if(sure<864000000)
begin
ses1=re;
ses2=si;
end
//10
else if(sure<880000000)
begin
ses1=mi;
ses2=doince;
end
else if(sure<896000000)
begin
ses1=re;
ses2=doince;
end
else if(sure<912000000)
begin
ses1=mi;
ses2=doince;
end
else if(sure<928000000)
begin
ses1=re;
ses2=doince;
end
else if(sure<944000000)
begin
ses1=mi;
ses2=doince;
end
else if(sure<960000000)
begin
ses1=re;
ses2=doince;
end
//11
else if(sure<976000000)
begin
ses1=mi;
ses2=si;
end
else if(sure<992000000)
begin
```

```
ses1=re;
ses2=si;
end
else if(sure<1008000000)
begin
ses1=mi;
ses2=si;
end
else if(sure<1024000000)
begin
ses1=re;
ses2=si;
end
else if(sure<1040000000)
begin
ses1=mi;
ses2=si;
end
else if(sure<1056000000)
begin
ses1=re;
ses2=si;
end
//12
else if(sure<1088000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<1152000000)
begin
ses1=mince;
ses2=fa;
end
//13
else if(sure<1184000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<1248000000)
begin
ses1=mince;
ses2=fa;
end
//14
else if(sure<1280000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<1344000000)
begin
ses1=mince;
ses2=sol;
```

```
end
//15
else if(sure<1376000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<1440000000)
begin
ses1=mince;
ses2=sol;
end
//16
else if(sure<1472000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<1536000000)
begin
ses1=mince;
ses2=la;
end
//17
else if(sure<1568000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<1632000000)
begin
ses1=mince;
ses2=la;
end
//18
else if(sure<1664000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<1728000000)
begin
ses1=mince;
ses2=sol;
end
//19
else if(sure<1760000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<1792000000)
begin
ses1=mince;
ses2=sol;
```

```
end
else if(sure<1824000000)
begin
ses1=si;
ses2=reince;
end
//20
else if(sure<1856000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<1888000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<1920000000)
begin
ses1=mince;
ses2=fa;
end
//21
else if(sure<1952000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<1984000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<2016000000)
begin
ses1=mince;
ses2=fa;
end
//22
else if(sure<2048000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<2080000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<2112000000)
begin
ses1=mince;
ses2=sol;
end
//23
```

```
else if(sure<2144000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<2176000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<2208000000)
begin
ses1=mince;
ses2=sol;
end
//24
else if(sure<2240000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<2272000000)
begin
ses1=mince;
ses2=la;
end
else if(sure<2304000000)
begin
ses1=mince;
ses2=la;
end
//25
else if(sure<2336000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<2368000000)
begin
ses1=mince;
ses2=la;
end
else if(sure<2400000000)
begin
ses1=mince;
ses2=la;
end
//26
else if(sure<2432000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<2464000000)
begin
```

```
ses1=mince;
ses2=sol;
end
else if(sure<2496000000)
begin
ses1=mince;
ses2=sol;
end
//27
else if(sure<2528000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<2560000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<2592000000)
begin
ses1=si;
ses2=sol;
end
//28
else if(sure<2624000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<2656000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<2688000000)
begin
ses1=la;
ses2=fa;
end
//29
else if(sure<2720000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<2752000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<2784000000)
begin
ses1=la;
ses2=fa;
```

```
end
//30
else if(sure<2816000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<2848000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<2880000000)
begin
ses1=si;
ses2=sol;
end
//31
else if(sure<2912000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<2944000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<2976000000)
begin
ses1=si;
ses2=sol;
end
//32
else if(sure<3008000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<3040000000)
begin
ses1=mince;
ses2=la;
end
else if(sure<3072000000)
begin
ses1=doince;
ses2=la;
end
//33
else if(sure<3104000000)
begin
ses1=doince;
ses2=la;
end
```

```
else if(sure<3136000000)
begin
ses1=mince;
ses2=la;
end
else if(sure<3168000000)
begin
ses1=doince;
ses2=la;
end
//34
else if(sure<3200000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<3232000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<3264000000)
begin
ses1=reince;
ses2=sol;
end
//35
else if(sure<3296000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<3328000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<3360000000)
begin
ses1=si;
ses2=sol;
end
//36
else if(sure<3392000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<3424000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<3456000000)
begin
```

```
ses1=la;
ses2=fa;
end
//37
else if(sure<3488000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<3520000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<3552000000)
begin
ses1=la;
ses2=fa;
end
//38
else if(sure<3584000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<3616000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<3648000000)
begin
ses1=si;
ses2=sol;
end
//39
else if(sure<3680000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<3712000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<3744000000)
begin
ses1=si;
ses2=sol;
end
//40
else if(sure<3776000000)
begin
ses1=doince;
```

```
ses2=la;
end
else if(sure<3808000000)
begin
ses1=mince;
ses2=la;
end
else if(sure<3840000000)
begin
ses1=doince;
ses2=la;
end
//41
else if(sure<3872000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<3904000000)
begin
ses1=mince;
ses2=la;
end
else if(sure<3936000000)
begin
ses1=doince;
ses2=la;
end
//42
else if(sure<3968000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<4000000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<4032000000)
begin
ses1=reince;
ses2=sol;
end
//43
else if(sure<4064000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<4096000000)
begin
ses1=mince;
ses2=sol;
end
```

```
else if(sure<4128000000)
begin
ses1=si;
ses2=sol;
end
//44
else if(sure<40'd4160000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<40'd4192000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<40'd4224000000)
begin
ses1=la;
ses2=fa;
end
//45
else if(sure<40'd4256000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<40'd4288000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<40'd4320000000)
begin
ses1=la;
ses2=fa;
end
//46
else if(sure<40'd4352000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<40'd4384000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<40'd4416000000)
begin
ses1=si;
ses2=sol;
end
//47
else if(sure<40'd4448000000)
```

```
begin
ses1=si;
ses2=sol;
end
else if(sure<40'd4480000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<40'd4512000000)
begin
ses1=si;
ses2=sol;
end
//48
else if(sure<40'd4544000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<40'd4576000000)
begin
ses1=mince;
ses2=la;
end
else if(sure<40'd4608000000)
begin
ses1=doince;
ses2=la;
end
//49
else if(sure<40'd4640000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<40'd4672000000)
begin
ses1=mince;
ses2=la;
end
else if(sure<40'd4704000000)
begin
ses1=doince;
ses2=la;
end
//50
else if(sure<40'd4736000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<40'd4768000000)
begin
ses1=mince;
```

```
ses2=sol;
end
else if(sure<40'd4800000000)
begin
ses1=reince;
ses2=sol;
end
//51
else if(sure<40'd4832000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<40'd4864000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<40'd4896000000)
begin
ses1=si;
ses2=sol;
end
//52
else if(sure<40'd4928000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<40'd4960000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<40'd4992000000)
begin
ses1=la;
ses2=fa;
end
//53
else if(sure<40'd5024000000)
begin
ses1=la;
ses2=fa;
end
else if(sure<40'd5056000000)
begin
ses1=mince;
ses2=fa;
end
else if(sure<40'd5088000000)
begin
ses1=la;
ses2=fa;
end
```

```
//54
else if(sure<40'd5120000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<40'd5152000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<40'd5184000000)
begin
ses1=si;
ses2=sol;
end
//55
else if(sure<40'd5216000000)
begin
ses1=si;
ses2=sol;
end
else if(sure<40'd5248000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<40'd5280000000)
begin
ses1=si;
ses2=sol;
end
//56
else if(sure<40'd5312000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<40'd5344000000)
begin
ses1=mince;
ses2=la;
end
else if(sure<40'd5376000000)
begin
ses1=doince;
ses2=la;
end
//57
else if(sure<40'd5408000000)
begin
ses1=doince;
ses2=la;
end
else if(sure<40'd5440000000)
```

```
begin
ses1=mince;
ses2=la;
end
else if(sure<40'd5472000000)
begin
ses1=doince;
ses2=la;
end
//58
else if(sure<40'd5504000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<40'd5536000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<40'd5568000000)
begin
ses1=reince;
ses2=sol;
end
//59
else if(sure<40'd5600000000)
begin
ses1=reince;
ses2=sol;
end
else if(sure<40'd5632000000)
begin
ses1=mince;
ses2=sol;
end
else if(sure<40'd5664000000)
begin
ses1=si;
ses2=sol;
end
//60
else if(sure<40'd5696000000)
begin
ses1=mince;
ses2=mi;
end
else if(sure<40'd5728000000)
begin
ses1=mince;
ses2=mi;
end
else if(sure<40'd5760000000)
begin
ses1=mince;
```

```
ses2=mi;
end

end // end of always1




task inverse;

// task parameters

input signed [30:0] a;
output signed [30:0] b;
//task start
begin
        if (a<0)
                b=~(a)+1;
        else
                b=a;
end

endtask //task end

task tasksine;

//task parameters
input h;
input [63:0] index;
input signed [63:0] in_y;
input signed [63:0] in_x;
reg  signed [63:0] in;
reg [63:0]  sin[0:360];
reg [63:0]  cos[0:360];
reg [63:0]  tan[0:180];



reg [63:0] a;
output [63:0] sinout;
output [63:0] cosout;

begin //task start

// define sine,cosine and tangent angles

sin[0]=0;
sin[1]=18;
sin[2]=36;
sin[3]=54;
sin[4]=71;
sin[5]=89;
sin[6]=107;
```

```
sin[7]=125;
sin[8]=143;
sin[9]=160;
sin[10]=178;
sin[11]=195;
sin[12]=213;
sin[13]=230;
sin[14]=248;
sin[15]=265;
sin[16]=282;
sin[17]=299;
sin[18]=316;
sin[19]=333;
sin[20]=350;
sin[21]=367;
sin[22]=384;
sin[23]=400;
sin[24]=416;
sin[25]=433;
sin[26]=449;
sin[27]=465;
sin[28]=481;
sin[29]=496;
sin[30]=512;
sin[31]=527;
sin[32]=543;
sin[33]=558;
sin[34]=573;
sin[35]=587;
sin[36]=602;
sin[37]=616;
sin[38]=630;
sin[39]=644;
sin[40]=658;
sin[41]=672;
sin[42]=685;
sin[43]=698;
sin[44]=711;
sin[45]=724;
sin[46]=737;
sin[47]=749;
sin[48]=761;
sin[49]=773;
sin[50]=784;
sin[51]=796;
sin[52]=807;
sin[53]=818;
sin[54]=828;
sin[55]=839;
sin[56]=849;
sin[57]=859;
sin[58]=868;
sin[59]=878;
sin[60]=887;
sin[61]=896;
```

```
sin[62]=904;
sin[63]=912;
sin[64]=920;
sin[65]=928;
sin[66]=935;
sin[67]=943;
sin[68]=949;
sin[69]=956;
sin[70]=962;
sin[71]=968;
sin[72]=974;
sin[73]=979;
sin[74]=984;
sin[75]=989;
sin[76]=994;
sin[77]=998;
sin[78]=1002;
sin[79]=1005;
sin[80]=1008;
sin[81]=1011;
sin[82]=1014;
sin[83]=1016;
sin[84]=1018;
sin[85]=1020;
sin[86]=1022;
sin[87]=1023;
sin[88]=1023;
sin[89]=1024;
sin[90]=1024;
sin[91]=1024;
sin[92]=1023;
sin[93]=1023;
sin[94]=1022;
sin[95]=1020;
sin[96]=1018;
sin[97]=1016;
sin[98]=1014;
sin[99]=1011;
sin[100]=1008;
sin[101]=1005;
sin[102]=1002;
sin[103]=998;
sin[104]=994;
sin[105]=989;
sin[106]=984;
sin[107]=979;
sin[108]=974;
sin[109]=968;
sin[110]=962;
sin[111]=956;
sin[112]=949;
sin[113]=943;
sin[114]=935;
sin[115]=928;
sin[116]=920;
```

```
sin[117]=912;
sin[118]=904;
sin[119]=896;
sin[120]=887;
sin[121]=878;
sin[122]=868;
sin[123]=859;
sin[124]=849;
sin[125]=839;
sin[126]=828;
sin[127]=818;
sin[128]=807;
sin[129]=796;
sin[130]=784;
sin[131]=773;
sin[132]=761;
sin[133]=749;
sin[134]=737;
sin[135]=724;
sin[136]=711;
sin[137]=698;
sin[138]=685;
sin[139]=672;
sin[140]=658;
sin[141]=644;
sin[142]=630;
sin[143]=616;
sin[144]=602;
sin[145]=587;
sin[146]=573;
sin[147]=558;
sin[148]=543;
sin[149]=527;
sin[150]=512;
sin[151]=496;
sin[152]=481;
sin[153]=465;
sin[154]=449;
sin[155]=433;
sin[156]=416;
sin[157]=400;
sin[158]=384;
sin[159]=367;
sin[160]=350;
sin[161]=333;
sin[162]=316;
sin[163]=299;
sin[164]=282;
sin[165]=265;
sin[166]=248;
sin[167]=230;
sin[168]=213;
sin[169]=195;
sin[170]=178;
sin[171]=160;
```

```
sin[172]=143;
sin[173]=125;
sin[174]=107;
sin[175]=89;
sin[176]=71;
sin[177]=54;
sin[178]=36;
sin[179]=18;
sin[180]=0;
sin[181]=-18;
sin[182]=-36;
sin[183]=-54;
sin[184]=-71;
sin[185]=-89;
sin[186]=-107;
sin[187]=-125;
sin[188]=-143;
sin[189]=-160;
sin[190]=-178;
sin[191]=-195;
sin[192]=-213;
sin[193]=-230;
sin[194]=-248;
sin[195]=-265;
sin[196]=-282;
sin[197]=-299;
sin[198]=-316;
sin[199]=-333;
sin[200]=-350;
sin[201]=-367;
sin[202]=-384;
sin[203]=-400;
sin[204]=-416;
sin[205]=-433;
sin[206]=-449;
sin[207]=-465;
sin[208]=-481;
sin[209]=-496;
sin[210]=-512;
sin[211]=-527;
sin[212]=-543;
sin[213]=-558;
sin[214]=-573;
sin[215]=-587;
sin[216]=-602;
sin[217]=-616;
sin[218]=-630;
sin[219]=-644;
sin[220]=-658;
sin[221]=-672;
sin[222]=-685;
sin[223]=-698;
sin[224]=-711;
sin[225]=-724;
sin[226]=-737;
```

```
sin[227]=-749;
sin[228]=-761;
sin[229]=-773;
sin[230]=-784;
sin[231]=-796;
sin[232]=-807;
sin[233]=-818;
sin[234]=-828;
sin[235]=-839;
sin[236]=-849;
sin[237]=-859;
sin[238]=-868;
sin[239]=-878;
sin[240]=-887;
sin[241]=-896;
sin[242]=-904;
sin[243]=-912;
sin[244]=-920;
sin[245]=-928;
sin[246]=-935;
sin[247]=-943;
sin[248]=-949;
sin[249]=-956;
sin[250]=-962;
sin[251]=-968;
sin[252]=-974;
sin[253]=-979;
sin[254]=-984;
sin[255]=-989;
sin[256]=-994;
sin[257]=-998;
sin[258]=-1002;
sin[259]=-1005;
sin[260]=-1008;
sin[261]=-1011;
sin[262]=-1014;
sin[263]=-1016;
sin[264]=-1018;
sin[265]=-1020;
sin[266]=-1022;
sin[267]=-1023;
sin[268]=-1023;
sin[269]=-1024;
sin[270]=-1024;
sin[271]=-1024;
sin[272]=-1023;
sin[273]=-1023;
sin[274]=-1022;
sin[275]=-1020;
sin[276]=-1018;
sin[277]=-1016;
sin[278]=-1014;
sin[279]=-1011;
sin[280]=-1008;
sin[281]=-1005;
```

```
sin[282]=-1002;
sin[283]=-998;
sin[284]=-994;
sin[285]=-989;
sin[286]=-984;
sin[287]=-979;
sin[288]=-974;
sin[289]=-968;
sin[290]=-962;
sin[291]=-956;
sin[292]=-949;
sin[293]=-943;
sin[294]=-935;
sin[295]=-928;
sin[296]=-920;
sin[297]=-912;
sin[298]=-904;
sin[299]=-896;
sin[300]=-887;
sin[301]=-878;
sin[302]=-868;
sin[303]=-859;
sin[304]=-849;
sin[305]=-839;
sin[306]=-828;
sin[307]=-818;
sin[308]=-807;
sin[309]=-796;
sin[310]=-784;
sin[311]=-773;
sin[312]=-761;
sin[313]=-749;
sin[314]=-737;
sin[315]=-724;
sin[316]=-711;
sin[317]=-698;
sin[318]=-685;
sin[319]=-672;
sin[320]=-658;
sin[321]=-644;
sin[322]=-630;
sin[323]=-616;
sin[324]=-602;
sin[325]=-587;
sin[326]=-573;
sin[327]=-558;
sin[328]=-543;
sin[329]=-527;
sin[330]=-512;
sin[331]=-496;
sin[332]=-481;
sin[333]=-465;
sin[334]=-449;
sin[335]=-433;
sin[336]=-416;
```

```
sin[337]=-400;
sin[338]=-384;
sin[339]=-367;
sin[340]=-350;
sin[341]=-333;
sin[342]=-316;
sin[343]=-299;
sin[344]=-282;
sin[345]=-265;
sin[346]=-248;
sin[347]=-230;
sin[348]=-213;
sin[349]=-195;
sin[350]=-178;
sin[351]=-160;
sin[352]=-143;
sin[353]=-125;
sin[354]=-107;
sin[355]=-89;
sin[356]=-71;
sin[357]=-54;
sin[358]=-36;
sin[359]=-18;
sin[360]=0;


// defining cosines

cos[0]=1024;
cos[1]=1024;
cos[2]=1023;
cos[3]=1023;
cos[4]=1022;
cos[5]=1020;
cos[6]=1018;
cos[7]=1016;
cos[8]=1014;
cos[9]=1011;
cos[10]=1008;
cos[11]=1005;
cos[12]=1002;
cos[13]=998;
cos[14]=994;
cos[15]=989;
cos[16]=984;
cos[17]=979;
cos[18]=974;
cos[19]=968;
cos[20]=962;
cos[21]=956;
cos[22]=949;
cos[23]=943;
cos[24]=935;
cos[25]=928;
cos[26]=920;
```

```
cos[27]=912;
cos[28]=904;
cos[29]=896;
cos[30]=887;
cos[31]=878;
cos[32]=868;
cos[33]=859;
cos[34]=849;
cos[35]=839;
cos[36]=828;
cos[37]=818;
cos[38]=807;
cos[39]=796;
cos[40]=784;
cos[41]=773;
cos[42]=761;
cos[43]=749;
cos[44]=737;
cos[45]=724;
cos[46]=711;
cos[47]=698;
cos[48]=685;
cos[49]=672;
cos[50]=658;
cos[51]=644;
cos[52]=630;
cos[53]=616;
cos[54]=602;
cos[55]=587;
cos[56]=573;
cos[57]=558;
cos[58]=543;
cos[59]=527;
cos[60]=512;
cos[61]=496;
cos[62]=481;
cos[63]=465;
cos[64]=449;
cos[65]=433;
cos[66]=416;
cos[67]=400;
cos[68]=384;
cos[69]=367;
cos[70]=350;
cos[71]=333;
cos[72]=316;
cos[73]=299;
cos[74]=282;
cos[75]=265;
cos[76]=248;
cos[77]=230;
cos[78]=213;
cos[79]=195;
cos[80]=178;
cos[81]=160;
```

```
cos[82]=143;
cos[83]=125;
cos[84]=107;
cos[85]=89;
cos[86]=71;
cos[87]=54;
cos[88]=36;
cos[89]=18;
cos[90]=0;
cos[91]=-18;
cos[92]=-36;
cos[93]=-54;
cos[94]=-71;
cos[95]=-89;
cos[96]=-107;
cos[97]=-125;
cos[98]=-143;
cos[99]=-160;
cos[100]=-178;
cos[101]=-195;
cos[102]=-213;
cos[103]=-230;
cos[104]=-248;
cos[105]=-265;
cos[106]=-282;
cos[107]=-299;
cos[108]=-316;
cos[109]=-333;
cos[110]=-350;
cos[111]=-367;
cos[112]=-384;
cos[113]=-400;
cos[114]=-416;
cos[115]=-433;
cos[116]=-449;
cos[117]=-465;
cos[118]=-481;
cos[119]=-496;
cos[120]=-512;
cos[121]=-527;
cos[122]=-543;
cos[123]=-558;
cos[124]=-573;
cos[125]=-587;
cos[126]=-602;
cos[127]=-616;
cos[128]=-630;
cos[129]=-644;
cos[130]=-658;
cos[131]=-672;
cos[132]=-685;
cos[133]=-698;
cos[134]=-711;
cos[135]=-724;
cos[136]=-737;
```

```
cos[137]=-749;
cos[138]=-761;
cos[139]=-773;
cos[140]=-784;
cos[141]=-796;
cos[142]=-807;
cos[143]=-818;
cos[144]=-828;
cos[145]=-839;
cos[146]=-849;
cos[147]=-859;
cos[148]=-868;
cos[149]=-878;
cos[150]=-887;
cos[151]=-896;
cos[152]=-904;
cos[153]=-912;
cos[154]=-920;
cos[155]=-928;
cos[156]=-935;
cos[157]=-943;
cos[158]=-949;
cos[159]=-956;
cos[160]=-962;
cos[161]=-968;
cos[162]=-974;
cos[163]=-979;
cos[164]=-984;
cos[165]=-989;
cos[166]=-994;
cos[167]=-998;
cos[168]=-1002;
cos[169]=-1005;
cos[170]=-1008;
cos[171]=-1011;
cos[172]=-1014;
cos[173]=-1016;
cos[174]=-1018;
cos[175]=-1020;
cos[176]=-1022;
cos[177]=-1023;
cos[178]=-1023;
cos[179]=-1024;
cos[180]=-1024;
cos[181]=-1024;
cos[182]=-1023;
cos[183]=-1023;
cos[184]=-1022;
cos[185]=-1020;
cos[186]=-1018;
cos[187]=-1016;
cos[188]=-1014;
cos[189]=-1011;
cos[190]=-1008;
cos[191]=-1005;
```

```
cos[192]=-1002;
cos[193]=-998;
cos[194]=-994;
cos[195]=-989;
cos[196]=-984;
cos[197]=-979;
cos[198]=-974;
cos[199]=-968;
cos[200]=-962;
cos[201]=-956;
cos[202]=-949;
cos[203]=-943;
cos[204]=-935;
cos[205]=-928;
cos[206]=-920;
cos[207]=-912;
cos[208]=-904;
cos[209]=-896;
cos[210]=-887;
cos[211]=-878;
cos[212]=-868;
cos[213]=-859;
cos[214]=-849;
cos[215]=-839;
cos[216]=-828;
cos[217]=-818;
cos[218]=-807;
cos[219]=-796;
cos[220]=-784;
cos[221]=-773;
cos[222]=-761;
cos[223]=-749;
cos[224]=-737;
cos[225]=-724;
cos[226]=-711;
cos[227]=-698;
cos[228]=-685;
cos[229]=-672;
cos[230]=-658;
cos[231]=-644;
cos[232]=-630;
cos[233]=-616;
cos[234]=-602;
cos[235]=-587;
cos[236]=-573;
cos[237]=-558;
cos[238]=-543;
cos[239]=-527;
cos[240]=-512;
cos[241]=-496;
cos[242]=-481;
cos[243]=-465;
cos[244]=-449;
cos[245]=-433;
cos[246]=-416;
```

```
cos[247]=-400;
cos[248]=-384;
cos[249]=-367;
cos[250]=-350;
cos[251]=-333;
cos[252]=-316;
cos[253]=-299;
cos[254]=-282;
cos[255]=-265;
cos[256]=-248;
cos[257]=-230;
cos[258]=-213;
cos[259]=-195;
cos[260]=-178;
cos[261]=-160;
cos[262]=-143;
cos[263]=-125;
cos[264]=-107;
cos[265]=-89;
cos[266]=-71;
cos[267]=-54;
cos[268]=-36;
cos[269]=-18;
cos[270]=0;
cos[271]=18;
cos[272]=36;
cos[273]=54;
cos[274]=71;
cos[275]=89;
cos[276]=107;
cos[277]=125;
cos[278]=143;
cos[279]=160;
cos[280]=178;
cos[281]=195;
cos[282]=213;
cos[283]=230;
cos[284]=248;
cos[285]=265;
cos[286]=282;
cos[287]=299;
cos[288]=316;
cos[289]=333;
cos[290]=350;
cos[291]=367;
cos[292]=384;
cos[293]=400;
cos[294]=416;
cos[295]=433;
cos[296]=449;
cos[297]=465;
cos[298]=481;
cos[299]=496;
cos[300]=512;
cos[301]=527;
```

```
cos[302]=543;
cos[303]=558;
cos[304]=573;
cos[305]=587;
cos[306]=602;
cos[307]=616;
cos[308]=630;
cos[309]=644;
cos[310]=658;
cos[311]=672;
cos[312]=685;
cos[313]=698;
cos[314]=711;
cos[315]=724;
cos[316]=737;
cos[317]=749;
cos[318]=761;
cos[319]=773;
cos[320]=784;
cos[321]=796;
cos[322]=807;
cos[323]=818;
cos[324]=828;
cos[325]=839;
cos[326]=849;
cos[327]=859;
cos[328]=868;
cos[329]=878;
cos[330]=887;
cos[331]=896;
cos[332]=904;
cos[333]=912;
cos[334]=920;
cos[335]=928;
cos[336]=935;
cos[337]=943;
cos[338]=949;
cos[339]=956;
cos[340]=962;
cos[341]=968;
cos[342]=974;
cos[343]=979;
cos[344]=984;
cos[345]=989;
cos[346]=994;
cos[347]=998;
cos[348]=1002;
cos[349]=1005;
cos[350]=1008;
cos[351]=1011;
cos[352]=1014;
cos[353]=1016;
cos[354]=1018;
cos[355]=1020;
cos[356]=1022;
```

```
cos[357]=1023;
cos[358]=1023;
cos[359]=1024;
cos[360]=1024;



// 10000* tangent + 100000000 values from -90 to 90
//
//tan[0]=0;
//tan[1]=99427100;
//tan[2]=99713637;
//tan[3]=99809189;
//tan[4]=99856993;
//tan[5]=99885699;
//tan[6]=99904856;
//tan[7]=99918557;
//tan[8]=99928846;
//tan[9]=99936862;
//tan[10]=99943287;
//tan[11]=99948554;
//tan[12]=99952954;
//tan[13]=99956685;
//tan[14]=99959892;
//tan[15]=99962679;
//tan[16]=99965126;
//tan[17]=99967291;
//tan[18]=99969223;
//tan[19]=99970958;
//tan[20]=99972525;
//tan[21]=99973949;
//tan[22]=99975249;
//tan[23]=99976441;
//tan[24]=99977540;
//tan[25]=99978555;
//tan[26]=99979497;
//tan[27]=99980374;
//tan[28]=99981193;
//tan[29]=99981960;
//tan[30]=99982679;
//tan[31]=99983357;
//tan[32]=99983997;
//tan[33]=99984601;
//tan[34]=99985174;
//tan[35]=99985719;
//tan[36]=99986236;
//tan[37]=99986730;
//tan[38]=99987201;
//tan[39]=99987651;
//tan[40]=99988082;
//tan[41]=99988496;
//tan[42]=99988894;
//tan[43]=99989276;
//tan[44]=99989645;
```

```
//tan[45]=99990000;
//tan[46]=99990343;
//tan[47]=99990675;
//tan[48]=99990996;
//tan[49]=99991307;
//tan[50]=99991609;
//tan[51]=99991902;
//tan[52]=99992187;
//tan[53]=99992464;
//tan[54]=99992735;
//tan[55]=99992998;
//tan[56]=99993255;
//tan[57]=99993506;
//tan[58]=99993751;
//tan[59]=99993991;
//tan[60]=99994226;
//tan[61]=99994457;
//tan[62]=99994683;
//tan[63]=99994905;
//tan[64]=99995123;
//tan[65]=99995337;
//tan[66]=99995548;
//tan[67]=99995755;
//tan[68]=99995960;
//tan[69]=99996161;
//tan[70]=99996360;
//tan[71]=99996557;
//tan[72]=99996751;
//tan[73]=99996943;
//tan[74]=99997133;
//tan[75]=99997321;
//tan[76]=99997507;
//tan[77]=99997691;
//tan[78]=99997874;
//tan[79]=99998056;
//tan[80]=99998237;
//tan[81]=99998416;
//tan[82]=99998595;
//tan[83]=99998772;
//tan[84]=99998949;
//tan[85]=99999125;
//tan[86]=99999301;
//tan[87]=99999476;
//tan[88]=99999651;
//tan[89]=99999825;
//tan[90]=100000000;
//tan[91]=100000175;
//tan[92]=100000349;
//tan[93]=100000524;
//tan[94]=100000699;
//tan[95]=100000875;
//tan[96]=100001051;
//tan[97]=100001228;
//tan[98]=100001405;
//tan[99]=100001584;
```

```
//tan[100]=100001763;
//tan[101]=100001944;
//tan[102]=100002126;
//tan[103]=100002309;
//tan[104]=100002493;
//tan[105]=100002679;
//tan[106]=100002867;
//tan[107]=100003057;
//tan[108]=100003249;
//tan[109]=100003443;
//tan[110]=100003640;
//tan[111]=100003839;
//tan[112]=100004040;
//tan[113]=100004245;
//tan[114]=100004452;
//tan[115]=100004663;
//tan[116]=100004877;
//tan[117]=100005095;
//tan[118]=100005317;
//tan[119]=100005543;
//tan[120]=100005774;
//tan[121]=100006009;
//tan[122]=100006249;
//tan[123]=100006494;
//tan[124]=100006745;
//tan[125]=100007002;
//tan[126]=100007265;
//tan[127]=100007536;
//tan[128]=100007813;
//tan[129]=100008098;
//tan[130]=100008391;
//tan[131]=100008693;
//tan[132]=100009004;
//tan[133]=100009325;
//tan[134]=100009657;
//tan[135]=100010000;
//tan[136]=100010355;
//tan[137]=100010724;
//tan[138]=100011106;
//tan[139]=100011504;
//tan[140]=100011918;
//tan[141]=100012349;
//tan[142]=100012799;
//tan[143]=100013270;
//tan[144]=100013764;
//tan[145]=100014281;
//tan[146]=100014826;
//tan[147]=100015399;
//tan[148]=100016003;
//tan[149]=100016643;
//tan[150]=100017321;
//tan[151]=100018040;
//tan[152]=100018807;
//tan[153]=100019626;
//tan[154]=100020503;
```

```
//tan[155]=100021445;
//tan[156]=100022460;
//tan[157]=100023559;
//tan[158]=100024751;
//tan[159]=100026051;
//tan[160]=100027475;
//tan[161]=100029042;
//tan[162]=100030777;
//tan[163]=100032709;
//tan[164]=100034874;
//tan[165]=100037321;
//tan[166]=100040108;
//tan[167]=100043315;
//tan[168]=100047046;
//tan[169]=100051446;
//tan[170]=100056713;
//tan[171]=100063138;
//tan[172]=100071154;
//tan[173]=100081443;
//tan[174]=100095144;
//tan[175]=100114301;
//tan[176]=100143007;
//tan[177]=100190811;
//tan[178]=100286363;
//tan[179]=100572900;
//tan[180]=200000000;

a=0;
in=0;

// end of definitions

if (h==0)
                    begin //  means we directly give the index
                    a=index;
                    end
//else begin // this means we obtained a tangent value
//
//          in=10000*in_y+100000000*in_x;
//
//           if(in>(tan[1]*in_x))
//                    a=1;
//           if(in>(tan[2]*in_x))
//                    a=2;
//           if(in>(tan[3]*in_x))
//                    a=3;
//           if(in>(tan[4]*in_x))
//                    a=4;
//           if(in>(tan[5]*in_x))
//                    a=5;
//           if(in>(tan[6]*in_x))
//                    a=6;
//           if(in>(tan[7]*in_x))
//                    a=7;
//           if(in>(tan[8]*in_x))
```

```
//                  a=8;
//          if(in>(tan[9]*in_x))
//                  a=9;
//          if(in>(tan[10]*in_x))
//                  a=10;
//          if(in>(tan[11]*in_x))
//                  a=11;
//          if(in>(tan[12]*in_x))
//                  a=12;
//          if(in>(tan[13]*in_x))
//                  a=13;
//          if(in>(tan[14]*in_x))
//                  a=14;
//          if(in>(tan[15]*in_x))
//                  a=15;
//          if(in>(tan[16]*in_x))
//                  a=16;
//          if(in>(tan[17]*in_x))
//                  a=17;
//          if(in>(tan[18]*in_x))
//                  a=18;
//          if(in>(tan[19]*in_x))
//                  a=19;
//          if(in>(tan[20]*in_x))
//                  a=20;
//          if(in>(tan[21]*in_x))
//                  a=21;
//          if(in>(tan[22]*in_x))
//                  a=22;
//          if(in>(tan[23]*in_x))
//                  a=23;
//          if(in>(tan[24]*in_x))
//                  a=24;
//          if(in>(tan[25]*in_x))
//                  a=25;
//          if(in>(tan[26]*in_x))
//                  a=26;
//          if(in>(tan[27]*in_x))
//                  a=27;
//          if(in>(tan[28]*in_x))
//                  a=28;
//          if(in>(tan[29]*in_x))
//                  a=29;
//          if(in>(tan[30]*in_x))
//                  a=30;
//          if(in>(tan[31]*in_x))
//                  a=31;
//          if(in>(tan[32]*in_x))
//                  a=32;
//          if(in>(tan[33]*in_x))
//                  a=33;
//          if(in>(tan[34]*in_x))
//                  a=34;
//          if(in>(tan[35]*in_x))
//                  a=35;
```

```
//          if(in>(tan[36]*in_x))
//                  a=36;
//          if(in>(tan[37]*in_x))
//                  a=37;
//          if(in>(tan[38]*in_x))
//                  a=38;
//          if(in>(tan[39]*in_x))
//                  a=39;
//          if(in>(tan[40]*in_x))
//                  a=40;
//          if(in>(tan[41]*in_x))
//                  a=41;
//          if(in>(tan[42]*in_x))
//                  a=42;
//          if(in>(tan[43]*in_x))
//                  a=43;
//          if(in>(tan[44]*in_x))
//                  a=44;
//          if(in>(tan[45]*in_x))
//                  a=45;
//          if(in>(tan[46]*in_x))
//                  a=46;
//          if(in>(tan[47]*in_x))
//                  a=47;
//          if(in>(tan[48]*in_x))
//                  a=48;
//          if(in>(tan[49]*in_x))
//                  a=49;
//          if(in>(tan[50]*in_x))
//                  a=50;
//          if(in>(tan[51]*in_x))
//                  a=51;
//          if(in>(tan[52]*in_x))
//                  a=52;
//          if(in>(tan[53]*in_x))
//                  a=53;
//          if(in>(tan[54]*in_x))
//                  a=54;
//          if(in>(tan[55]*in_x))
//                  a=55;
//          if(in>(tan[56]*in_x))
//                  a=56;
//          if(in>(tan[57]*in_x))
//                  a=57;
//          if(in>(tan[58]*in_x))
//                  a=58;
//          if(in>(tan[59]*in_x))
//                  a=59;
//          if(in>(tan[60]*in_x))
//                  a=60;
//          if(in>(tan[61]*in_x))
//                  a=61;
//          if(in>(tan[62]*in_x))
//                  a=62;
//          if(in>(tan[63]*in_x))
```

```
//                  a=63;
//          if(in>(tan[64]*in_x))
//                  a=64;
//          if(in>(tan[65]*in_x))
//                  a=65;
//          if(in>(tan[66]*in_x))
//                  a=66;
//          if(in>(tan[67]*in_x))
//                  a=67;
//          if(in>(tan[68]*in_x))
//                  a=68;
//          if(in>(tan[69]*in_x))
//                  a=69;
//          if(in>(tan[70]*in_x))
//                  a=70;
//          if(in>(tan[71]*in_x))
//                  a=71;
//          if(in>(tan[72]*in_x))
//                  a=72;
//          if(in>(tan[73]*in_x))
//                  a=73;
//          if(in>(tan[74]*in_x))
//                  a=74;
//          if(in>(tan[75]*in_x))
//                  a=75;
//          if(in>(tan[76]*in_x))
//                  a=76;
//          if(in>(tan[77]*in_x))
//                  a=77;
//          if(in>(tan[78]*in_x))
//                  a=78;
//          if(in>(tan[79]*in_x))
//                  a=79;
//          if(in>(tan[80]*in_x))
//                  a=80;
//          if(in>(tan[81]*in_x))
//                  a=81;
//          if(in>(tan[82]*in_x))
//                  a=82;
//          if(in>(tan[83]*in_x))
//                  a=83;
//          if(in>(tan[84]*in_x))
//                  a=84;
//          if(in>(tan[85]*in_x))
//                  a=85;
//          if(in>(tan[86]*in_x))
//                  a=86;
//          if(in>(tan[87]*in_x))
//                  a=87;
//          if(in>(tan[88]*in_x))
//                  a=88;
//          if(in>(tan[89]*in_x))
//                  a=89;
//          if(in>(tan[90]*in_x))
//                  a=90;
```

```
//          if(in>(tan[91]*in_x))
//                  a=91;
//          if(in>(tan[92]*in_x))
//                  a=92;
//          if(in>(tan[93]*in_x))
//                  a=93;
//          if(in>(tan[94]*in_x))
//                  a=94;
//          if(in>(tan[95]*in_x))
//                  a=95;
//          if(in>(tan[96]*in_x))
//                  a=96;
//          if(in>(tan[97]*in_x))
//                  a=97;
//          if(in>(tan[98]*in_x))
//                  a=98;
//          if(in>(tan[99]*in_x))
//                  a=99;
//          if(in>(tan[100]*in_x))
//                  a=100;
//          if(in>(tan[101]*in_x))
//                  a=101;
//          if(in>(tan[102]*in_x))
//                  a=102;
//          if(in>(tan[103]*in_x))
//                  a=103;
//          if(in>(tan[104]*in_x))
//                  a=104;
//          if(in>(tan[105]*in_x))
//                  a=105;
//          if(in>(tan[106]*in_x))
//                  a=106;
//          if(in>(tan[107]*in_x))
//                  a=107;
//          if(in>(tan[108]*in_x))
//                  a=108;
//          if(in>(tan[109]*in_x))
//                  a=109;
//          if(in>(tan[110]*in_x))
//                  a=110;
//          if(in>(tan[111]*in_x))
//                  a=111;
//          if(in>(tan[112]*in_x))
//                  a=112;
//          if(in>(tan[113]*in_x))
//                  a=113;
//          if(in>(tan[114]*in_x))
//                  a=114;
//          if(in>(tan[115]*in_x))
//                  a=115;
//          if(in>(tan[116]*in_x))
//                  a=116;
//          if(in>(tan[117]*in_x))
//                  a=117;
//          if(in>(tan[118]*in_x))
```

```
//                  a=118;
//          if(in>(tan[119]*in_x))
//                  a=119;
//          if(in>(tan[120]*in_x))
//                  a=120;
//          if(in>(tan[121]*in_x))
//                  a=121;
//          if(in>(tan[122]*in_x))
//                  a=122;
//          if(in>(tan[123]*in_x))
//                  a=123;
//          if(in>(tan[124]*in_x))
//                  a=124;
//          if(in>(tan[125]*in_x))
//                  a=125;
//          if(in>(tan[126]*in_x))
//                  a=126;
//          if(in>(tan[127]*in_x))
//                  a=127;
//          if(in>(tan[128]*in_x))
//                  a=128;
//          if(in>(tan[129]*in_x))
//                  a=129;
//          if(in>(tan[130]*in_x))
//                  a=130;
//          if(in>(tan[131]*in_x))
//                  a=131;
//          if(in>(tan[132]*in_x))
//                  a=132;
//          if(in>(tan[133]*in_x))
//                  a=133;
//          if(in>(tan[134]*in_x))
//                  a=134;
//          if(in>(tan[135]*in_x))
//                  a=135;
//          if(in>(tan[136]*in_x))
//                  a=136;
//          if(in>(tan[137]*in_x))
//                  a=137;
//          if(in>(tan[138]*in_x))
//                  a=138;
//          if(in>(tan[139]*in_x))
//                  a=139;
//          if(in>(tan[140]*in_x))
//                  a=140;
//          if(in>(tan[141]*in_x))
//                  a=141;
//          if(in>(tan[142]*in_x))
//                  a=142;
//          if(in>(tan[143]*in_x))
//                  a=143;
//          if(in>(tan[144]*in_x))
//                  a=144;
//          if(in>(tan[145]*in_x))
//                  a=145;
```

```
//          if(in>(tan[146]*in_x))
//                  a=146;
//          if(in>(tan[147]*in_x))
//                  a=147;
//          if(in>(tan[148]*in_x))
//                  a=148;
//          if(in>(tan[149]*in_x))
//                  a=149;
//          if(in>(tan[150]*in_x))
//                  a=150;
//          if(in>(tan[151]*in_x))
//                  a=151;
//          if(in>(tan[152]*in_x))
//                  a=152;
//          if(in>(tan[153]*in_x))
//                  a=153;
//          if(in>(tan[154]*in_x))
//                  a=154;
//          if(in>(tan[155]*in_x))
//                  a=155;
//          if(in>(tan[156]*in_x))
//                  a=156;
//          if(in>(tan[157]*in_x))
//                  a=157;
//          if(in>(tan[158]*in_x))
//                  a=158;
//          if(in>(tan[159]*in_x))
//                  a=159;
//          if(in>(tan[160]*in_x))
//                  a=160;
//          if(in>(tan[161]*in_x))
//                  a=161;
//          if(in>(tan[162]*in_x))
//                  a=162;
//          if(in>(tan[163]*in_x))
//                  a=163;
//          if(in>(tan[164]*in_x))
//                  a=164;
//          if(in>(tan[165]*in_x))
//                  a=165;
//          if(in>(tan[166]*in_x))
//                  a=166;
//          if(in>(tan[167]*in_x))
//                  a=167;
//          if(in>(tan[168]*in_x))
//                  a=168;
//          if(in>(tan[169]*in_x))
//                  a=169;
//          if(in>(tan[170]*in_x))
//                  a=170;
//          if(in>(tan[171]*in_x))
//                  a=171;
//          if(in>(tan[172]*in_x))
//                  a=172;
//          if(in>(tan[173]*in_x))
```

```
//                      a=173;
//          if(in>(tan[174]*in_x))
//                      a=174;
//          if(in>(tan[175]*in_x))
//                      a=175;
//          if(in>(tan[176]*in_x))
//                      a=176;
//          if(in>(tan[177]*in_x))
//                      a=177;
//          if(in>(tan[178]*in_x))
//                      a=178;
//          if(in>(tan[179]*in_x))
//                      a=179;
//          if(in>(tan[180]*in_x))
//                      a=180;
//
//
//          if (in_x>0 && in_y>0)
//                      a=a-90;
//          if (in_x<0 && in_y>0)
//                      a=a+90;
//          if (in_x<0 && in_y<0)
//                      a=a+90;
//          if (in_x>0 && in_y<0)
//                      a=a+270;
//          a=a+1;
//end

sinout=sin[a];
cosout=cos[a];

end //task ending

endtask //end of tasksine


always @(posedge timer && pause==0)

begin

if(reset==0)
begin
saniye=0;
dakika=0;
end
else if(k3==0)
begin
saniye=saniye+1;
if(saniye==60)
begin
saniye=0;
dakika=dakika+1;
end
end
saniye1=saniye%10;
```

```
saniye2=(saniye-saniye1)/10;
end

always @(posedge clk2 && pause==0)
begin

// changing angle

//if(reset==1)
//gc=gc+1;
if(reset==1)// && pause==0)
begin
if((duz==1)&&(ters==0))
theta=theta+donk;
else if((duz==0)&&(ters==1))
if(theta==0 || (theta==1 && donk==2) || (theta==2 && donk==3) || (theta==3 && donk==4) || (theta==4 &&
donk==5) )
theta=359;
else
theta=theta-donk;

if(theta>360)
theta=theta-360;

if(theta<361 && theta>180)
theta2=theta-180;
else
theta2=theta+180;


//updating coordinates
tasksine(0,theta,1,1,sinout1,cosout1);
merk1y_up=(((90)*sinout1)>>>10)+merky;
merk1x_up=(((90)*cosout1)>>>10)+merkx;


tasksine(0,theta2,1,1,sinout2,cosout2);
merk2y_up=(((90)*sinout2)>>>10)+merky;
merk2x_up=(((90)*cosout2)>>>10)+merkx;
end

else// if (reset==0 && pause==0)
begin
merkx=320;
merky=340;
merk1y=340;
merk1x=320+90;
merk2y=340;
merk2x=320-90;

theta=0;
yark=13;

merk1x_up=merk1x;
merk1y_up=merk1y;
```

```
merk2x_up=merk2x;
merk2y_up=merk2y;
end


end
always @(posedge clk_50 && pause==0) //moving rectangle

begin



if(reset==1)// && pause==0)
begin

//case(score)
//
//0: begin ss0=0; ss1=0; ss2=0; ss3=0; ss4=0; ss5=0; ss6=1; end
//1: begin ss0=1; ss1=0; ss2=0; ss3=1; ss4=1; ss5=1; ss6=1; end
//2: begin ss0=0; ss1=0; ss2=1; ss3=0; ss4=0; ss5=1; ss6=0; end
//3: begin ss0=0; ss1=0; ss2=0; ss3=0; ss4=1; ss5=1; ss6=0; end
//4: begin ss0=1; ss1=0; ss2=0; ss3=1; ss4=1; ss5=0; ss6=0; end
//5: begin ss0=0; ss1=1; ss2=0; ss3=0; ss4=1; ss5=0; ss6=0; end
//6: begin ss0=0; ss1=1; ss2=0; ss3=0; ss4=0; ss5=0; ss6=0; end
//7: begin ss0=0; ss1=0; ss2=0; ss3=1; ss4=1; ss5=1; ss6=1; end
//8: begin ss0=0; ss1=0; ss2=0; ss3=0; ss4=0; ss5=0; ss6=0; end
//9: begin ss0=0; ss1=0; ss2=0; ss3=0; ss4=1; ss5=0; ss6=0; end
//
//endcase

if(my>10 && my<400)
st1=0;
if(myu>10 && myu<400)
st2=0;
if(my2>10 && my2<400)
st3=0;
if(merkyd>10 && merkyd<400)
st4=0;


if(my>480 && st1==0 && my<500)
begin
score=score+1;
st1=1;
end
if(myu>480 && st2==0 && myu<500)
begin
score=score+1;
st2=1;
end
if(my2>480 && st3==0 && my2<500)
begin
score=score+1;
st3=1;
end
```

```
if(merkyd>480 && st4==0 && merkyd<500)
begin
score=score+1;
st4=1;
end

score1=score%10;
score2=((score-score1)/10)%10;
score3=((score-score1-score2*10)/100)%10;


if(k3)
                                        begin
                                        gc=0;
                                        donk=0;
                                        hizx=0;
                                        donk2=0;
                                        countg=0;
                                        hizu=0;
                                                end

                                                else
                                                begin
                                                countg=countg+1;
                                                if(countg==3750)
                                                begin
                                                countg=0;
                                                gc=gc+1;
                                                donk2=donk2+1;
                                                hizu=hizu+2;
                                                donk=donk+1;
                                                end
                                                        end

if(theta3>240)
theta4=theta3-240;
else
theta4=theta3+120;

if(theta3>180)
theta5=theta3-180;
else
theta5=theta3+180;

if(theta3>60)
theta6=theta3-60;
else
theta6=theta3+300;

tasksine(0,theta3,1,1,sinout,cosout);
k1x=mx+((rad*cosout)>>>10);
k1y=my+((rad*sinout)>>>10);

tasksine(0,theta4,1,1,sinout,cosout);
k2x=mx+((rad*cosout)>>>10);
```

```
k2y=my+((rad*sinout)>>>10);

tasksine(0,theta5,1,1,sinout,cosout);
k3x=mx+((rad*cosout)>>>10);
k3y=my+((rad*sinout)>>>10);

tasksine(0,theta6,1,1,sinout,cosout);
k4x=mx+((rad*cosout)>>>10);
k4y=my+((rad*sinout)>>>10);

theta3=theta3+donk2;
my=my+gc;
mx=mx-hizu;


if(theta3>360)
theta3=theta3-360;


//// changing angle
//
////if(reset==1)
////gc=gc+1;
//
//if((duz==1)&&(ters==0))
//theta=theta+donk;
//else if((duz==0)&&(ters==1))
//if(theta==0 || (theta==1 && gc==2) || (theta==2 && gc==3) )
//theta=359;
//else
//theta=theta-donk;
//
//if(theta>360)
//theta=theta-360;
//
//if(theta<361 && theta>180)
//theta2=theta-180;
//else
//theta2=theta+180;
//
//
////updating coordinates
//tasksine(0,theta,1,1,sinout1,cosout1);
//merk1y_up=(((90)*sinout1)>>>10)+merky;
//merk1x_up=(((90)*cosout1)>>>10)+merkx;
//
//
//tasksine(0,theta2,1,1,sinout2,cosout2);
//merk2y_up=(((90)*sinout2)>>>10)+merky;
//merk2x_up=(((90)*cosout2)>>>10)+merkx;


//ucgen

if(theta3u>240)
```

```
theta4u=theta3u-240;
else
theta4u=theta3u+120;

if(theta3u>120)
theta5u=theta3u-120;
else
theta5u=theta3u+240;


tasksine(0,theta3u,1,1,sinout,cosout);
k1xu=mxu+((radu*cosout)>>>10);
k1yu=myu+((radu*sinout)>>>10);

tasksine(0,theta4u,1,1,sinout,cosout);
k2xu=mxu+((radu*cosout)>>>10);
k2yu=myu+((radu*sinout)>>>10);

tasksine(0,theta5u,1,1,sinout,cosout);
k3xu=mxu+((radu*cosout)>>>10);
k3yu=myu+((radu*sinout)>>>10);

theta3u=theta3u+donk2;
myu=myu+gc;
mxu=mxu+hizu;

if(theta3u>360)
theta3u=theta3u-360;

//daire

        if(merkxd<200 || merkxd>440)
hizx=-hizx;


merkyd=merkyd+gc;
merkxd=merkxd+hizx;

//yamuk sekizgen

if(theta7>240)
theta8=theta7-240;
else
theta8=theta7+120;

if(theta7>180)
theta9=theta7-180;
else
theta9=theta7+180;

if(theta7>60)
theta10=theta7-60;
else
theta10=theta7+300;
```

```
tasksine(0,theta7,1,1,sinout,cosout);
k5x=mx2+((rad2*cosout)>>>10);
k5y=my2+((rad2*sinout)>>>10);

tasksine(0,theta8,1,1,sinout,cosout);
k6x=mx2+((rad2*cosout)>>>10);
k6y=my2+((rad2*sinout)>>>10);

tasksine(0,theta9,1,1,sinout,cosout);
k7x=mx2+((rad2*cosout)>>>10);
k7y=my2+((rad2*sinout)>>>10);

tasksine(0,theta10,1,1,sinout,cosout);
k8x=mx2+((rad2*cosout)>>>10);
k8y=my2+((rad2*sinout)>>>10);

theta7=theta7+donk2;
my2=my2+gc;

if(theta7>360)
theta7=theta7-360;

end

else// if(reset==0 && pause==0)

begin

st1=0;
st2=0;
st3=0;
st4=0;

score=0;
countg=0;
gc=2;
donk=1;
donk2=1;
hizx=4;
hizu=2;

theta3=30;

rad=50;

mx=600;
my=1000;


c2adik=8760;//aslen8660

a1=0;
a2=0;
a3=0;
a4=0;
```

```
// for dipole

//merkx=320;
//merky=340;
//merk1y=340;
//merk1x=320-80;
//merk2y=340;
//merk2x=320+80;
//
//theta=0;
//yark=15;
//
//merk1x_up=merk1x;
//merk1y_up=merk1y;
//merk2x_up=merk2x;
//merk2y_up=merk2y;

theta3u=0;
radu=40;
//donku=1;
c2aucgen=4224;
u1=0;
u2=0;
u3=0;
mxu=400;
myu=500;

yarkd=16;
merkxd=200;
merkyd=100;


//yamuk sekizgen

theta7=30;

rad2=25;

mx2=250;
my2=760;



c2adik2=4330; //aslen 8660

a5=0;
a6=0;
a7=0;
a8=0;


end

end // end of always4
```

```
//always @(posedge clk_25M)
//begin
//
//
//
//
//end // end of always 5

//vga

always@(posedge clk_25M)

begin

	if (hs<800)
		hs=hs+1;
	else
		begin
		hs=0;
		if(vs<525)
			vs=vs+1;
		else
			vs=0;
		end
end

always@(posedge clk_25M)

begin

	if(vs<2)
		vsig=0;
	else
		vsig=1;
	if(hs<95)
		hsig=0;
	else
		hsig=1;

end // end of always

always@ (posedge clk_25M)

begin

		if(hs<144 || hs>784 || vs<35 || vs>515)
			begin

			r=0;
			g=0;
			b=0;


			end
```

```
else //input drawing conditions here
        begin

                                        //doga

                a1= (hs-145-k2x)*(k1y-vs+36)-(hs-145-k1x)*(k2y-vs+36);
                inverse(a1,a1);
                a2= (hs-145-k3x)*(k2y-vs+36)-(hs-145-k2x)*(k3y-vs+36);
                inverse(a2,a2);
                a3= (hs-145-k4x)*(k3y-vs+36)-(hs-145-k3x)*(k4y-vs+36);
                inverse(a3,a3);
                a4= (hs-145-k1x)*(k4y-vs+36)-(hs-145-k4x)*(k1y-vs+36);
                inverse(a4,a4);

                u1= (hs-145-k2xu)*(k1yu-vs+36)-(hs-145-k1xu)*(k2yu-vs+36);
                inverse(u1,u1);
                u2= (hs-145-k3xu)*(k2yu-vs+36)-(hs-145-k2xu)*(k3yu-vs+36);
                inverse(u2,u2);
                u3= (hs-145-k1xu)*(k3yu-vs+36)-(hs-145-k3xu)*(k1yu-vs+36);
                inverse(u3,u3);


                a5= (hs-145-k6x)*(k5y-vs+36)-(hs-145-k5x)*(k6y-vs+36);
                inverse(a5,a5);
                a6= (hs-145-k7x)*(k6y-vs+36)-(hs-145-k6x)*(k7y-vs+36);
                inverse(a6,a6);
                a7= (hs-145-k8x)*(k7y-vs+36)-(hs-145-k7x)*(k8y-vs+36);
                inverse(a7,a7);
                a8= (hs-145-k5x)*(k8y-vs+36)-(hs-145-k8x)*(k5y-vs+36);
                inverse(a8,a8);



        if(  ((((hs-145)-(merk1x_up))*((hs-145)-(merk1x_up))+((vs-36)-(merk1y_up))*((vs-
36)-(merk1y_up)))<(yark*yark)) )
                begin
                b=210;
                r=210;
                g=70;
                k=1;
                end
        else                    if((((hs-145)-(merk2x_up))*((hs-145)-(merk2x_up))+((vs-36)-
(merk2y_up))*((vs-36)-(merk2y_up))<(yark*yark) ))
        begin
        r=111;
        g=111;
        b=255;
        k=1;
        end
        else
                begin
                b=0;
                r=0;
                g=0;
                k=0;
```

```
                                end

                                if( (((((hs-145)-(50))*((hs-145)-(50)))+((vs-36)-(50))*((vs-36)-(50)))<(15))
||  ((((((hs-145)-(200))*((hs-145)-(200))))+((vs-36)-(90))*((vs-36)-(90)))<(15)) || ((((((hs-145)-(340))*((hs-
145)-(340))))+((vs-36)-(300))*((vs-36)-(300)))<(15))  ||  ((((((hs-145)-(80))*((hs-145)-(80))))+((vs-36)-
(350))*((vs-36)-(350)))<(15))  )
                                begin
                                r=255;
                                g=255;
                                b=255;
                                end

                                if((((((hs-145)-(450))*((hs-145)-(450))))+((vs-36)-(100))*((vs-36)-
(100)))<(60))
                                begin
                                r=255;
                                g=255;
                                b=0;
                                end

                          if( ~((a1+a2+a3+a4)>(c2adik)) || ~((u1+u2+u3)>(c2aucgen)) || ((((((hs-145)-
(merkxd))*((hs-145)-(merkxd)))>>1)+((vs-36)-(merkyd))*((vs-36)-(merkyd)))<(yarkd*yarkd)) ||
~((a5+a6+a7+a8)>(c2adik2)) )
                                begin
                                r=255;
                                g=0;
                                b=0;
                                k2=1;
                                end
                                else
                                k2=0;


if(   ( (hs-145)<510 && (hs-145)>500 && (vs-36)==400) ) case(saniye1) 0,2,3,5,6,7,8,9: begin r=255;
g=255; b=255; end endcase
if(  ( (hs-145)<510 && (hs-145)>500 && (vs-36)==410) ) case(saniye1) 2,3,4,5,6,8,9: begin r=255; g=255;
b=255; end endcase
if(  ( (hs-145)<510 && (hs-145)>500 && (vs-36)==420) ) case(saniye1) 0,2,3,5,6,8,9: begin r=255; g=255;
b=255; end endcase

if(  ( (vs-36)<410 && (vs-36)>400 && (hs-145)==510) ) case(saniye1) 0,1,2,3,4,7,8,9: begin r=255; g=255;
b=255; end endcase
if(   ( (vs-36)<420 && (vs-36)>410 && (hs-145)==510) ) case(saniye1) 0,1,3,4,5,6,7,8,9: begin r=255;
g=255; b=255; end endcase
if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==500) ) case(saniye1) 0,4,5,6,8,9: begin r=255; g=255;
b=255; end endcase
if(  ( (vs-36)<420 && (vs-36)>410 && (hs-145)==500) ) case(saniye1) 0,2,6,8: begin r=255; g=255; b=255;
end endcase

if(   ( (hs-145)<495 && (hs-145)>485 && (vs-36)==400) ) case(saniye2) 0,2,3,5,6,7,8,9: begin r=255;
g=255; b=255; end endcase
if(  ( (hs-145)<495 && (hs-145)>485 && (vs-36)==410) ) case(saniye2) 2,3,4,5,6,8,9: begin r=255; g=255;
b=255; end endcase
if(  ( (hs-145)<495 && (hs-145)>485 && (vs-36)==420) ) case(saniye2) 0,2,3,5,6,8,9: begin r=255; g=255;
b=255; end endcase
```

```
if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==495) ) case(saniye2) 0,1,2,3,4,7,8,9: begin r=255; g=255; b=255; end endcase
if(    ( (vs-36)<420 && (vs-36)>410 && (hs-145)==495) ) case(saniye2) 0,1,3,4,5,6,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==485) ) case(saniye2) 0,4,5,6,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<420 && (vs-36)>410 && (hs-145)==485) ) case(saniye2) 0,2,6,8: begin r=255; g=255; b=255; end endcase

if(   ( (hs-145)<475 && (hs-145)>465 && (vs-36)==400) ) case(dakika) 0,2,3,5,6,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (hs-145)<475 && (hs-145)>465 && (vs-36)==410) ) case(dakika) 2,3,4,5,6,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (hs-145)<475 && (hs-145)>465 && (vs-36)==420) ) case(dakika) 0,2,3,5,6,8,9: begin r=255; g=255; b=255; end endcase

if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==475) ) case(dakika) 0,1,2,3,4,7,8,9: begin r=255; g=255; b=255; end endcase
if(    ( (vs-36)<420 && (vs-36)>410 && (hs-145)==475) ) case(dakika) 0,1,3,4,5,6,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==465) ) case(dakika) 0,4,5,6,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<420 && (vs-36)>410 && (hs-145)==465) ) case(dakika) 0,2,6,8: begin r=255; g=255; b=255; end endcase

if(((vs-36)==405 || (vs-36)==415)&&(hs-145)==480)
begin
r=255;
g=255;
b=255;
end

if(   ( (hs-145)<185 && (hs-145)>175 && (vs-36)==400) ) case(score1) 0,2,3,5,6,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (hs-145)<185 && (hs-145)>175 && (vs-36)==410) ) case(score1) 2,3,4,5,6,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (hs-145)<185 && (hs-145)>175 && (vs-36)==420) ) case(score1) 0,2,3,5,6,8,9: begin r=255; g=255; b=255; end endcase

if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==185) ) case(score1) 0,1,2,3,4,7,8,9: begin r=255; g=255; b=255; end endcase
if(    ( (vs-36)<420 && (vs-36)>410 && (hs-145)==185) ) case(score1) 0,1,3,4,5,6,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==175) ) case(score1) 0,4,5,6,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<420 && (vs-36)>410 && (hs-145)==175) ) case(score1) 0,2,6,8: begin r=255; g=255; b=255; end endcase

if(   ( (hs-145)<170 && (hs-145)>160 && (vs-36)==400) ) case(score2) 0,2,3,5,6,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (hs-145)<170 && (hs-145)>160 && (vs-36)==410) ) case(score2) 2,3,4,5,6,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (hs-145)<170 && (hs-145)>160 && (vs-36)==420) ) case(score2) 0,2,3,5,6,8,9: begin r=255; g=255; b=255; end endcase
```

```
if(  ( (vs-36)<410 && (vs-36)>400 && (hs-145)==170) ) case(score2) 0,1,2,3,4,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<420 && (vs-36)>410 && (hs-145)==170) ) case(score2) 0,1,3,4,5,6,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==160) ) case(score2) 0,4,5,6,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<420 && (vs-36)>410 && (hs-145)==160) ) case(score2) 0,2,6,8: begin r=255; g=255; b=255; end endcase

if(   ( (hs-145)<155 && (hs-145)>145 && (vs-36)==400) ) case(score3) 0,2,3,5,6,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (hs-145)<155 && (hs-145)>145 && (vs-36)==410) ) case(score3) 2,3,4,5,6,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (hs-145)<155 && (hs-145)>145 && (vs-36)==420) ) case(score3) 0,2,3,5,6,8,9: begin r=255; g=255; b=255; end endcase

if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==155) ) case(score3) 0,1,2,3,4,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<420 && (vs-36)>410 && (hs-145)==155) ) case(score3) 0,1,3,4,5,6,7,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<410 && (vs-36)>400 && (hs-145)==145) ) case(score3) 0,4,5,6,8,9: begin r=255; g=255; b=255; end endcase
if(   ( (vs-36)<420 && (vs-36)>410 && (hs-145)==145) ) case(score3) 0,2,6,8: begin r=255; g=255; b=255; end endcase




//                              if(   (((hs-145)==k1x)&&((vs-36)==k1y))  ||  (((hs-145)==k2x)&&((vs-36)==k2y))  ||  (((hs-145)==k3x)&&((vs-36)==k3y))  ||  (((hs-145)==k4x)&&((vs-36)==k4y))  ||  (((hs-145)==k5x)&&((vs-36)==k5y))||  (((hs-145)==k6x)&&((vs-36)==k6y))||  (((hs-145)==k7x)&&((vs-36)==k7y))|| (((hs-145)==k8x)&&((vs-36)==k8y)))
//                              begin
//                              r=0;
//                              g=0;
//                              b=255;
//                              end
//                              if( ((hs-145)==mx2)&&((vs-36)==my2))
//                              begin
//                              r=0;
//                              g=255;
//                              b=0;
//                              end
                                if((k&&k2))
                                begin
                                r=0;
                                g=255;
                                b=0;
                                k3=1;
                                end
                                if(reset==0)
                                k3=0;
                                if(k3==1)
```

```
                                        begin
                                                if((((hs-145)>200 && (hs-145)<300) && ((vs-36)>40 &&
(vs-36)<75))
|| (((hs-145)>200 && (hs-145)<225) && ((vs-36)>40 && (vs-36)<240))
 || (((hs-145)>200 && (hs-145)<300) && ((vs-36)>190 && (vs-36)<240))
 || (((hs-145)>275 && (hs-145)<300) && ((vs-36)>140 && (vs-36)<240))
|| (((hs-145)>330 && (hs-145)<430) && ((vs-36)>40 && (vs-36)<75))
|| (((hs-145)>330 && (hs-145)<355) && ((vs-36)>40 && (vs-36)<240))
 || (((hs-145)>330 && (hs-145)<430) && ((vs-36)>190 && (vs-36)<240))
 || (((hs-145)>405 && (hs-145)<430) && ((vs-36)>140 && (vs-36)<240))
  || (((hs-145)>250 && (hs-145)<300) && ((vs-36)>140 && (vs-36)<160))
 || (((hs-145)>380 && (hs-145)<430) && ((vs-36)>140 && (vs-36)<160)))
begin
r=180;
g=255;
b=20;
end
end




                        end // end of display

end  // end of always

endmodule
```