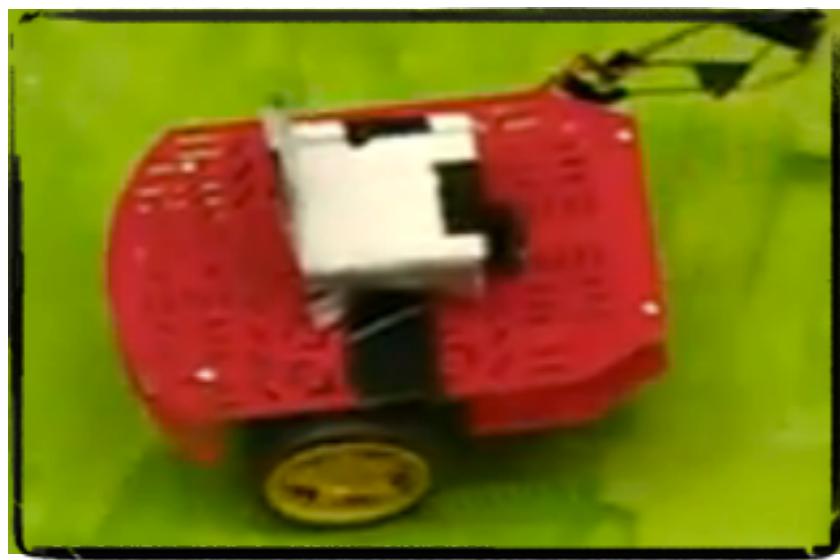


ECE 241

UNIVERSITY OF TORONTO FACULTY OF APPLIED SCIENCE AND ENGINEERING



Project Title	A controller for wheel chair prototype
Tutorial Section	108
Teaching Assistant	Mohamed Abdelfattah
Prepared By (Name and Student Numbers of the team)	Vaibhav Vijay 1000073029 Rohaila Zehra 0998371177

ECE 241

Contents

I. Introduction

- Description of goals and projects

II. The Design

III. Conclusion

- Working of the project

IV. What would you do differently

V. Appendices

ECE 241

Introduction

Our team made a controller for wheel chair prototype to help physically challenged people traverse between two points using a keyboard for input and motors as an output. The controller as in the FPGA controls the motors with respect to the keyboard readings.

For every destination there is a specific keyboard key that user pressed to reach their and an another key to return home from that specific location (**Appendix A**).

This project took approximately 60 hours to complete. The workload was divided even among each team member (**Appendix B**).

The Design

The keyboard part was involved figuring out and writing code for user interface with the wheelchair, allowing the user to communicate with the wheelchair via a PC keyboard. We used the PS2 controller module found on the EECG website to connect keyboard with the DE2 board. The keyboard interface was made possible using 3 modules. The PS2 controller module, the DATA_IN, and the COMMAND_OUT module. There was an additional (optional) module that enabled the hex make codes (of keys pressed) to be shown on the 7 segment displays. This was used as a debugging technique so that we knew that the key hits were being registered. The PS2 controller module utilized (instantiated) the other two modules so that they worked in conjunction to enable the keyboard as the input giver. The hex make codes of keys being hit on the keyboard were passed to the ps2 controller module that took the key press as input and produced the hexadecimal “make-code” of the key being pressed. We used this hex “make-code” for determining which key was pressed. The appropriate action was taken depending on which key was pressed (i.e. which hex-make code was produced by the PS2 controller).

The motors worked with the FPGA required extra motor-driving chip (**Appendix C**) to work since the FPGA was incapable of supplying enough power to drive two motors. The L293D acted as a bridge between the FPGA and the motors(**Appendix D**).

A case statement was used to determine what action should be taken depending on which hex-make code was detected (i.e. which key was pressed). For example, a hex make code of 1B would correspond to S being pressed. We drew up a “map” on a large piece of cardboard; the map consists of “roads” and “destinations (**Appendix G**).” The roads were the paths that the wheelchair was supposed to follow in reaching a “destination;” the “destinations” were locations of the board to which the wheelchair was to navigate to. We did not utilize sensors in this navigation process, rather we *guessed* how much time the wheelchair was supposed to turn and move forward and in *which* direction. The wheelchair’s movements were a sequence of turns and forward movements, the sequence unique for each “destination.” We used the CLOCK_50 system clock of the FPGA as the counter for controlling the amount of time each motor ran. The decision to turn on which motor and in which direction depends on which

ECE 241

destination was specified by the user. The wheelchair had two motors. The logic is pretty straightforward; for turning left, only the left wheel would spin in the forward direction while for turning right, only the right wheel of the chair would spin forward while the other remained stationary. After the specified wheel turned the appropriate direction to the appropriate degree, the wheelchair would stop turning and then start moving forward. After reaching the specified destination , the wheelchair would stop moving and wait for another command from the keyboard issued by the user. If the user pressed the “reverse” key for the destination, then the wheelchair would “drive” back “home.” “Home” was the original location where the wheelchair always started.

For navigating back home, the module simply carries out the same set of instructions, but *in reverse*. After receiving a command for “home” from the keyboard, the wheelchair would first take a U turn, then carry out a reverse set of instructions to what it originally did (**Appendix E**). This whole mechanism depends on correct synchronization between the motors turning and the CLOCK_50 system clock.

Each destination was reached by the wheelchair carrying out a unique sequence of turns and forward driving(**Appendix F**).

Conclusion

The wheelchair’s movements are a sequence of turns and forward movements; this sequence is unique for each “destination.” We used the CLOCK_50 system clock of the FPGA as the counter for controlling the amount of time each motor ran. After the specified wheel turned the appropriate direction to the appropriate degree, the wheelchair would stop turning and then start moving forward. After reaching the specified destination, the wheelchair would stop moving and wait for another command from the keyboard issued by the user. If the user pressed the “reverse” key for the destination, then the wheelchair would “drive” back “home.” “Home” was the original location where the wheelchair always started. For example, pressing “S” would make the wheelchair drive to the location labelled “Starbucks” on the map. If the user tapped “T”, then the wheelchair would navigate back to “Home.” Supposed “Starbucks” is located 20 cm ahead, 34 cm to the left , then 21 cm forward again. The Verilog module responsible for the wheelchair navigation would first check which key was pressed by checking which hex make-code it had received. After determining that S was pressed (for Starbucks), the wheelchair would begin travelling down the path towards Starbucks. It would run both motors forward for about 3 seconds (corresponding to a distance travelled of 20 cm), then for 1 second, turn the left motor clockwise and leave the right one at rest, one that turns (hence “turning left”); finally it would run both motors forward again for about 3 seconds (**Appendix H**).

For driving back home, the wheelchair simply carries out the same set of instructions but *in reverse*. After receiving a command for “home” from the keyboard, the wheelchair would first take a U turn, and then carry out a reverse set of instructions to what it originally did. This whole mechanism depends on correct synchronization between the motors turning and the CLOCK_50 system clock(**Appendix I**).

ECE 241

What would you do differently

The biggest problem was that the wheelchair didn't drive exactly to the specified location for all destinations. The timing of the turns and the forward driving could have been a lot better. The wheelchair's movements require more precision. We do not consider it sufficient for the wheelchair to merely move *vaguely* in the direction of a destination. We want the wheelchair's movements to be sharper and more precise. This is the main part of this project and it could be improved a lot.

ECE 241

Appendices

Appendix A

Destination	Begin Journey	Return Home
Restaurant	R	H
Supermarket	M	N
Park	A	B
Hospital	J	I
Starbucks	S	T

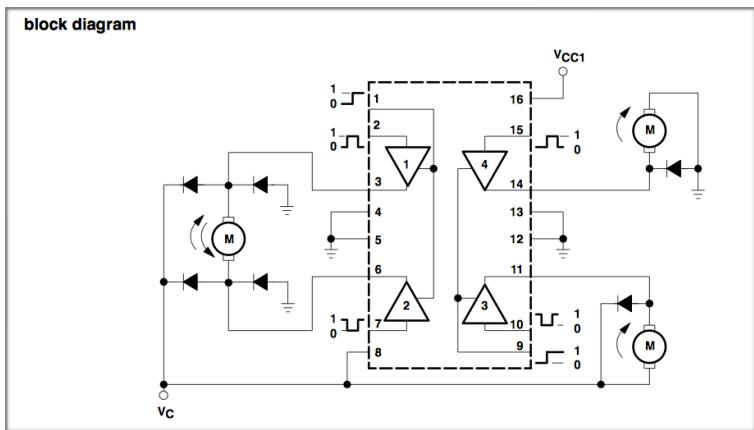
Appendix B

TASK	VAIBHAV VIJAY	ROHAILA ZEHRA
Working of Keyboard		✓
Hardware	✓	
Working of Motors	✓	
Game Logic	✓	✓
Assembling the prototype	✓	✓
Making the image		✓
Working of VGA	✓	
Drawing the map on cardboard		✓

ECE 241

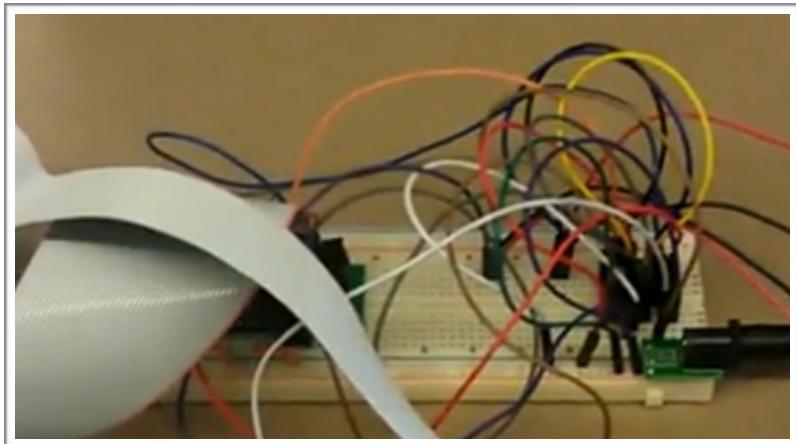
Appendix C

This is a block diagram of chip L293D



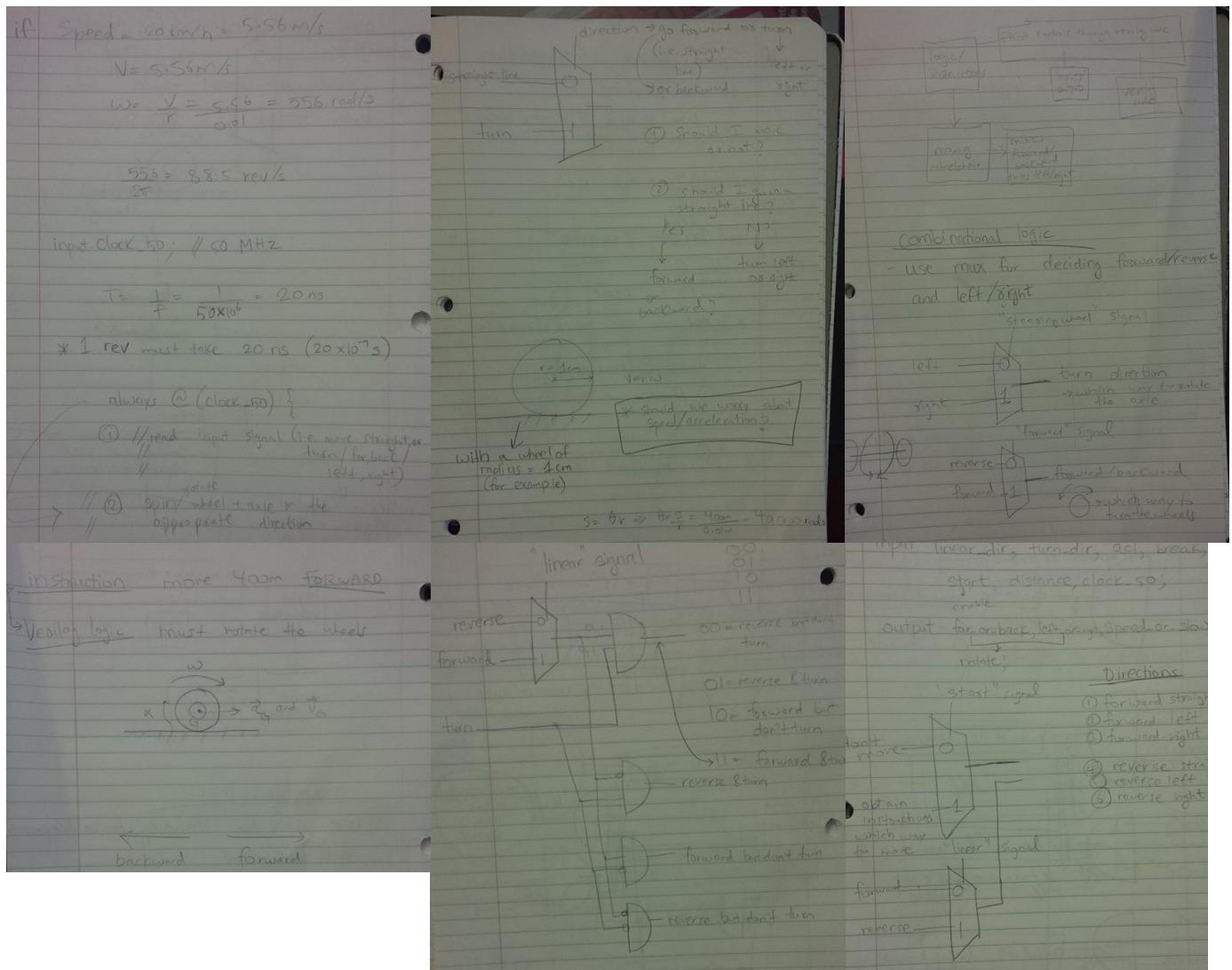
link: <http://users.ece.utexas.edu/~valvano/Datasheets/L293d.pdf>

Appendix D



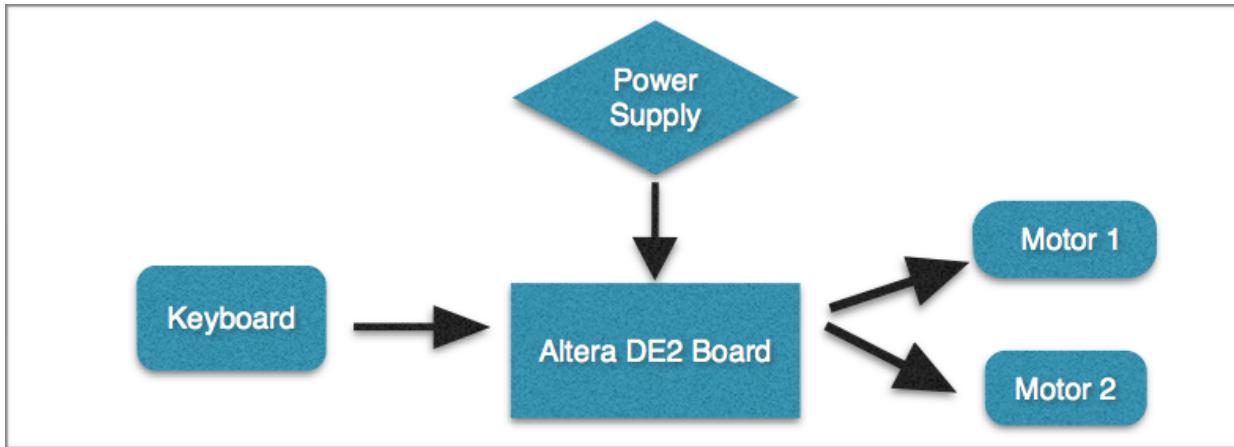
ECE 241

Appendix E



ECE 241

Appendix F

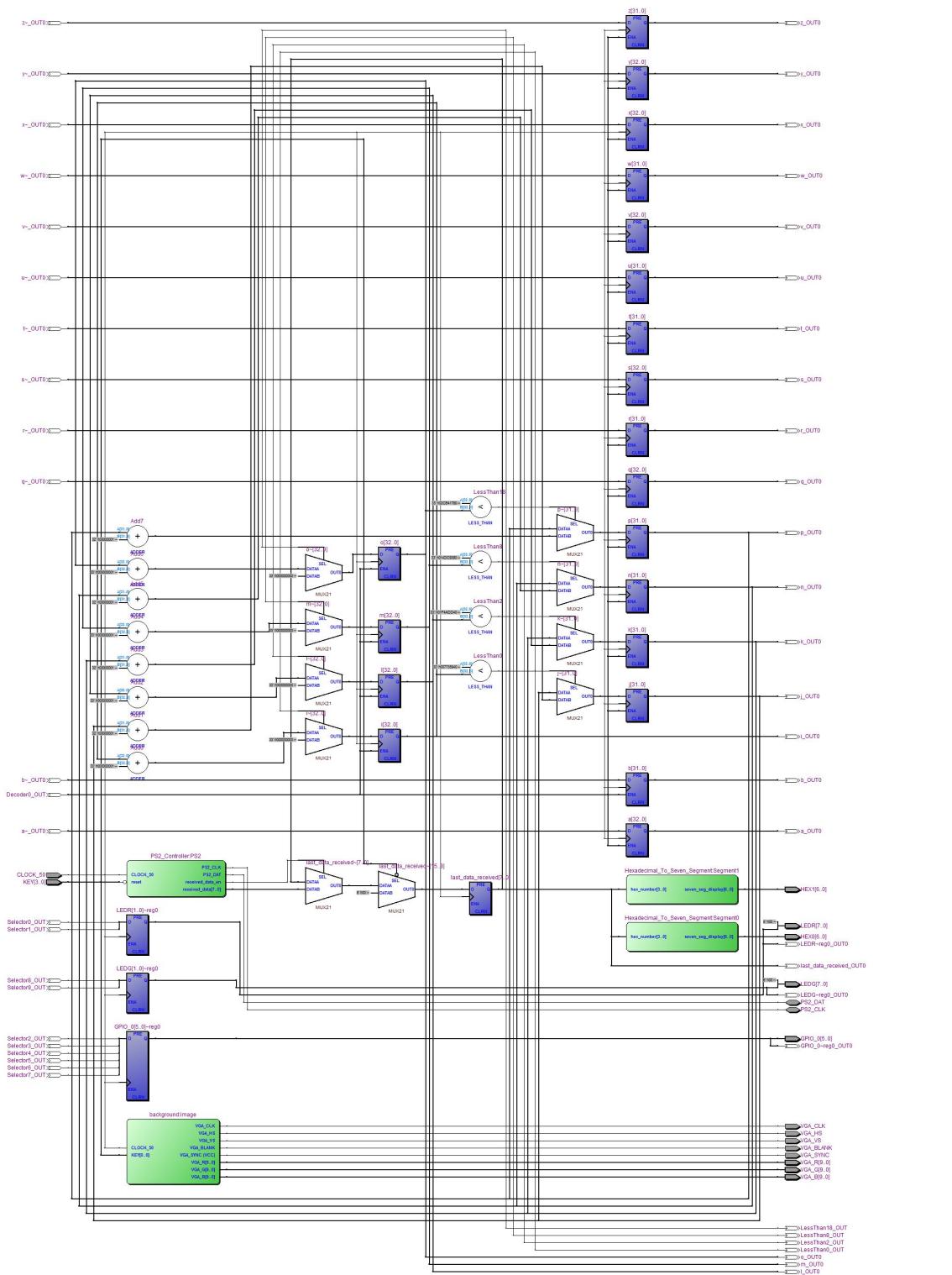


Appendix G

MOTOR 1	MOTOR 2	STATE
0	0	STOP
1	1	FORWARD
1	0	LEFT
0	1	RIGHT

ECE 241

Appendix H



ECE 241

Appendix I

```
module GPS( //PS2_Demo
    // Inputs
    CLOCK_50,
    KEY, LEDG, LEDR, SW,
    // Bidirectionals
    PS2_CLK,
    PS2_DAT,
    GPIO_0,
    // Outputs
    HEX0,
    HEX1,
    HEX2,
    HEX3,
    HEX4,
    HEX5,
    HEX6,
    HEX7,
    VGA_CLK,                                // VGA Clock
    VGA_HS,                                 // VGA H_SYNC
    VGA_VS,                                 // VGA V_SYNC
    VGA_BLANK,                             // VGA BLANK
    VGA_SYNC,                               // VGA SYNC
    VGA_R,                                  // VGA Red[9:0]
    VGA_G,                                  // VGA Green[9:0]
    VGA_B
);
//*********************************************************************
/*          Parameter Declarations           */
//*********************************************************************
//*********************************************************************
/*          Port Declarations            */
//*********************************************************************
// Inputs
input  [7:0]SW;
input          CLOCK_50;
input  [3:0]   KEY;
// Bidirectionals
inout         PS2_CLK;
inout         PS2_DAT;
output reg [7:0] LEDG;
wire enable;
assign enable = KEY[1];
// Outputs
output  [6:0]  HEX0;
output  [6:0]  HEX1;
output  [6:0]  HEX2;
output  [6:0]  HEX3;
output  [6:0]  HEX4;
output  [6:0]  HEX5;
output  [6:0]  HEX6;
output  [6:0]  HEX7;
output  reg[7:0] LEDR;
output reg [5:0]GPIO_0;
//*********************************************************************
/*          Internal Wires and Registers Declarations      */
//*********************************************************************
// Internal Wires
wire  [7:0]  ps2_key_data;
```

ECE 241

```
wire ps2_key_pressed;
// Internal Registers
reg [7:0] last_data_received;
integer j =0;
reg [32:0]i;
integer k =0;
reg [32:0]l;
integer n =0;
reg [32:0]m;
integer p =0;
reg [32:0]o;
integer r =0;
reg [32:0]q;
integer t =0;
reg [32:0]s;
integer u =0;
reg [32:0]y;
integer w =0;
reg [32:0]v;
integer z =0;
reg [32:0]x;
integer b =0;
reg [32:0]a;
// State Machine Registers
//****************************************************************************
/*          Finite State Machine(s)          */
//****************************************************************************
/*          Sequential Logic          */
//****************************************************************************
background image
(
    CLOCK_50,                                // On Board 50 MHz
    KEY,                                     // Push Button[0:0]
    VGA_CLK,                                  // VGA Clock
    VGA_HS,                                   // VGA H_SYNC
    VGA_VS,                                   // VGA V_SYNC
    VGA_BLANK,                                // VGA BLANK
    VGA_SYNC,                                 // VGA SYNC
    VGA_R,                                    // VGA Red[9:0]
    VGA_G,                                    // VGA Green[9:0]
    VGA_B,                                    // VGA Blue[9:0]
);
output VGA_CLK;                            // VGA Clock
output VGA_HS;                            // VGA H_SYNC
output VGA_VS;                            // VGA V_SYNC
output VGA_BLANK;                           // VGA BLANK
output VGA_SYNC;                           // VGA SYNC
output [9:0] VGA_R;                      // VGA Red[9:0]
output [9:0] VGA_G;                      // VGA Green[9:0]
output [9:0] VGA_B;                      // VGA Blue[9:0]

always @(posedge CLOCK_50)
begin
    if (KEY[0] == 1'b0)
        last_data_received <= 8'h00;
    else if (ps2_key_pressed == 1'b1)
        begin
            last_data_received <= ps2_key_data;
        end
end
always @ (posedge CLOCK_50)
begin
```

ECE 241

```
case(last_data_received)
  8'h2D: //R
    begin
      i <= i + 1;
      if (i > 125000000)
        begin
          LEDR[0] <= 0;
          LEDR[1] <= 0;
          GPIO_0[0] <= 0;//enable
          GPIO_0[5] <= 0;//enable
        end
      else if (i < 125000000 && j == 0) //15 move
        begin
          i <= 0;
          j = j + 1;
        end
      end
    begin
      GPIO_0[0] <= 1;//enable for motor 1
      GPIO_0[2] <= 1;//for moving the motor 1 clockwise
      GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
      GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
      GPIO_0[3] <= 1;//for moving the motor 2 clockwise
      GPIO_0[5] <= 1;//enable for motor 2
      <= 1;
      <= 1;
    end
  end
  8'h33: //H
    begin
      l <= l + 1;
      if (l > 525000000)
        begin
          LEDG[0] <= 0;
          LEDG[1] <= 0;
          GPIO_0[0] <= 0;//enable
          GPIO_0[5] <= 0;//enable
        end
      else if (l < 200000000 && k == 0) // uturn
        begin
          l <= 0;
          k = k + 1;
        end
      end
    begin
      GPIO_0[0] <= 0;//enable for motor 1
      GPIO_0[2] <= 0;//for moving the motor 1 clockwise
      GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
      GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
      GPIO_0[3] <= 1;//for moving the motor 2 clockwise
      GPIO_0[5] <= 1;//enable for motor 2
    end
  end
```

ECE 241

```
<= 0;                                     LEDG[1]
<= 1;                                     LEDG[0]
end
else if (l > 200000000 && l < 325000000 &&
begin

GPIO_0[0] <= 1;//enable for motor 1

GPIO_0[2] <= 1;//for moving the motor 1 clockwise

GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise

GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise

GPIO_0[3] <= 1;//for moving the motor 2 clockwise

GPIO_0[5] <= 1;//enable for motor 2                                     LEDG[1]
<= 1;                                     LEDG[0]
<= 1;

end
else if (l > 325000000 && l <
begin

525000000 && k == 0) // u turn

GPIO_0[0] <= 0;//enable for motor 1

GPIO_0[2] <= 0;//for moving the motor 1 clockwise

GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise

GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise

GPIO_0[3] <= 1;//for moving the motor 2 clockwise

GPIO_0[5] <= 1;//enable for motor 2                                     LEDG[1]
<= 1;                                     LEDG[0]
<= 0;

end
8'h3A:   //M
begin
m <= m + 1;
if (m > 350000000)
begin
LEDR[0] <= 0;
LEDR[1] <= 0;
GPIO_0[0] <= 0;//enable
GPIO_0[5] <= 0;//enable
m <= 0;
n = n +1;
end
else if (m < 25000000 && n == 0) //move
begin
for motor 1
for motor 2
forward

```

ECE 241

```
GPIO_0[0] <= 1;//enable for motor 1
GPIO_0[2] <= 1;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
<= 1;
<= 1;
125000000 && n == 0) //TURN left
else if (m > 25000000 && m <
begin
GPIO_0[0] <= 1;//enable for motor 1
GPIO_0[2] <= 1;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 0;//for moving the motor 2 clockwise
GPIO_0[5] <= 0;//enable for motor 2
<= 0;
<= 1;
200000000) //move forward
else if (m > 125000000 && n == 0 && m <
begin
GPIO_0[0] <= 1;//enable for motor 1
GPIO_0[2] <= 1;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
<= 1;
<= 1;
300000000) //right
else if (m > 200000000 && n == 0 && m <
begin
LEDR[1]
LEDR[0]
end
end
begin
LEDR[0]
LEDR[1]
end
begin
LEDR[1]
LEDR[0]
end
begin
```

ECE 241

```
GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[2] <= 0;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
=< 0;
=< 1;
&& m < 350000000) //forward
begin
  GPIO_0[0] <= 1;//enable for motor 1
  GPIO_0[2] <= 1;//for moving the motor 1 clockwise
  GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
  GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
  GPIO_0[3] <= 1;//for moving the motor 2 clockwise
  GPIO_0[5] <= 1;//enable for motor 2
  =< 1;
  =< 1;
end
8'h31:    //N
begin
  o <= o + 1;
  if (o > 750000000)
    begin
      LEDR[0] <= 0;
      LEDR[1] <= 0;
      GPIO_0[0] <= 0;//enable
      GPIO_0[5] <= 0;//enable
      o <= 0;
      p = p + 1;
    end
  else if (p == 0 && o < 200000000) //uturn
    begin
      GPIO_0[0] <= 1;//
      GPIO_0[2] <= 1;//
      GPIO_0[4] <= 0;//
      GPIO_0[1] <= 0;//
    end
  end
for motor 1
for motor 2
enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
begin
  end
else if (m > 300000000 && n == 0)
begin
  end
  LEDR[0]
  LEDR[1]
```

ECE 241

```
for moving the motor 2 clockwise
enable for motor 2
    GPIO_0[3] <= 0;;
    GPIO_0[5] <= 0;;
    LEDR[0] <= 1;
    LEDR[1] <= 0;
end

else if (o > 200000000 && p == 0 && o <
250000000 )//forward
begin
    GPIO_0[0] <= 1;;
    GPIO_0[2] <= 1;;
    GPIO_0[4] <= 0;;
    GPIO_0[1] <= 0;;
    GPIO_0[3] <= 1;;
    GPIO_0[5] <= 1;;
    LEDR[0] <= 1;
    LEDR[1] <= 1;
end

else if (o > 250000000 && p == 0 && o <
350000000) //left
begin
    GPIO_0[0] <= 1;;
enable for motor 1
    GPIO_0[2] <= 1;//for moving the motor 1 clockwise
    GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
    GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
    GPIO_0[3] <= 0;//for moving the motor 2 clockwise
    GPIO_0[5] <= 0;//enable for motor 2
    <= 0;
    <= 1;
end

else if (o > 350000000 && p == 0 && o <
425000000 ) //move forward
begin
    GPIO_0[0] <= 1;//enable for
motor 1
    GPIO_0[2] <= 1;;
    GPIO_0[4] <= 0;;
    GPIO_0[1] <= 0;;
    GPIO_0[3] <= 1;;
    GPIO_0[5] <= 1;;
    LEDR[0] <= 1;
```

ECE 241

```
LEDR[1] <= 1;
end
else if (o > 425000000 && p == 0 && o <
525000000) //move right
begin

GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[2] <= 0;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
<= 0;
<= 1;
end
else if (o > 525000000 && p == 0 && o <
550000000) //move forward
begin
GPIO_0[0] <= 1;//enable for
GPIO_0[2] <= 1;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
LEDR[0] <= 1;
LEDR[1] <= 1;
end
else if (o > 550000000 && p == 0 && o <
750000000) //u turn
begin
GPIO_0[0] <= 1;//enable for motor 1
GPIO_0[2] <= 1;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 0;//for moving the motor 2 clockwise
GPIO_0[5] <= 0;//enable for motor 2
LEDR[0] <= 1;
LEDR[1] <= 0;
end
end
8'h1B: //S
begin
q <= q + 1; //q and r
```

ECE 241

```
if (q > 500000000)
begin
    LEDR[0] <= 0;
    LEDR[1] <= 0;
    GPIO_0[0] <= 0;//enable
    GPIO_0[5] <= 0;//enable

for motor 1
for motor 2
    q <= 0;
    r = r +1;
end
else if (r == 0 && q < 200000000)//u turn
begin
    GPIO_0[0] <= 1;//enable
    GPIO_0[2] <= 1;//
    GPIO_0[4] <= 0;//
    GPIO_0[1] <= 0;//
    GPIO_0[3] <= 0;//
    GPIO_0[5] <= 0;//

for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2
    LEDR[0] <= 1;
    LEDR[1] <= 0;
end

else if (q > 200000000 && r == 0
&& q < 325000000 ) //forward
begin
    GPIO_0[0] <= 1;//
    GPIO_0[2] <= 1;//
    GPIO_0[4] <= 0;//
    GPIO_0[1] <= 0;//
    GPIO_0[3] <= 1;//
    GPIO_0[5] <= 1;//

enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2
    LEDR[0] <= 1;
    LEDR[1] <= 1;
end

else if (q > 325000000 && r == 0
&& q < 425000000)//left
begin
    GPIO_0[0] <= 1;//enable
for motor 1
    GPIO_0[2] <= 1;//for moving the motor 1 clockwise
    GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
    GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
    GPIO_0[3] <= 0;//for moving the motor 2 clockwise
    GPIO_0[5] <= 0;//enable for motor 2
```

ECE 241

```
<= 0;                                     LEDR[0]
<= 1;                                     LEDR[1]
                                           end
                                           else if (q > 425000000 && r == 0
                                           begin
                                               GPIO_0[0] <= 1;//enable
                                               GPIO_0[2] <= 1;//
                                               GPIO_0[4] <= 0;//
                                               GPIO_0[1] <= 0;//
                                               GPIO_0[3] <= 1;//
                                               GPIO_0[5] <= 1;//
                                               LEDR[0] <= 1;
                                               LEDR[1] <= 1;
                                           end
                                          
                                           end
                                          
8'h2C: // T
begin
    s <= s + 1;
    if (s > 500000000)
        begin
            LEDR[0] <= 0;
            LEDR[1] <= 0;
            GPIO_0[0] <= 0;//enable
            GPIO_0[5] <= 0;//enable
            s <= 0;
            t = t + 1;
        end
        else if (t == 0 && s <
        begin
            GPIO_0[0] <= 1;//
            GPIO_0[2] <= 1;//
            GPIO_0[4] <= 0;//
            GPIO_0[1] <= 0;//
            GPIO_0[3] <= 0;//
            GPIO_0[5] <= 0;//
            LEDR[0] <= 1;
            LEDR[1] <= 0;
        end
        else if (s > 200000000 && t == 0
        begin
            LEDR[0] <= 1;
            LEDR[1] <= 0;
        end
        && s < 275000000)//forward
```

ECE 241

```
enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2
GPIO_0[0] <= 1;//
GPIO_0[2] <= 1;//
GPIO_0[4] <= 0;//
GPIO_0[1] <= 0;//
GPIO_0[3] <= 1;//
GPIO_0[5] <= 1;//
LEDR[0] <= 1;
LEDR[1] <= 1;
end

else if (s > 275000000 && t == 0
begin

GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[2] <= 0;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
LEDR[0]
<= 0;
LEDR[1]
<= 1;
end

else if (s > 375000000 && t == 0
begin
GPIO_0[0] <= 1;//
GPIO_0[2] <= 1;//
GPIO_0[4] <= 0;//
GPIO_0[1] <= 0;//
GPIO_0[3] <= 1;//
GPIO_0[5] <= 1;//
LEDR[0] <= 1;
LEDR[1] <= 1;
end

enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2
end

8'h3B: //J
begin
y <= y + 1;
if (y >
begin
625000000)
```

ECE 241

```
LEDR[0] <= 0;
LEDR[1] <= 0;
GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[5] <= 0;//enable for motor 2
y <= 0;
u = u +1;
y < 200000000)//uturn
begin
    end
else if (u == 0 &&
begin
    GPIO_0[0] <= 0;//enable for motor 1
    GPIO_0[2] <= 0;//for moving the motor 1 clockwise
    GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
    GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
    GPIO_0[3] <= 1;//for moving the motor 2 clockwise
    GPIO_0[5] <= 1;//enable for motor 2
    <= 0;
    <= 1;
    LEDG[1]
    LEDG[0]
    end
else if (y >
200000000 && u == 0 && y < 250000000 ) //forward
begin
    enable for motor 1
    for moving the motor 1 clockwise
    for moving the motor 1 anti clockwise
    for moving the motor 2 anti clockwise
    for moving the motor 2 clockwise
    enable for motor 2
    LEDR[0] <= 1;
    LEDR[1] <= 1;
    end
else if (y >
250000000 && u == 0 && y < 350000000)//right
begin
    GPIO_0[0] <= 0;//enable for motor 1
    GPIO_0[2] <= 0;//for moving the motor 1 clockwise
    GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
    GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
```

ECE 241

```
GPIO_0[3] <= 1;//for moving the motor 2 clockwise  
GPIO_0[5] <= 1;//enable for motor 2  
<= 0;  
<= 1;  
  
350000000 && u == 0 && y < 450000000)//forward  
  
enable for motor 1  
for moving the motor 1 clockwise  
for moving the motor 1 anti clockwise  
for moving the motor 2 anti clockwise  
for moving the motor 2 clockwise  
enable for motor 2  
  
450000000 && u == 0 && y < 550000000)//left  
  
GPIO_0[0] <= 1;//enable for motor 1  
GPIO_0[2] <= 1;//for moving the motor 1 clockwise  
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise  
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise  
GPIO_0[3] <= 0;//for moving the motor 2 clockwise  
GPIO_0[5] <= 0;//enable for motor 2  
<= 0;  
<= 1;  
  
550000000 && u == 0 && y < 625000000)//forward  
  
enable for motor 1  
for moving the motor 1 clockwise  
for moving the motor 1 anti clockwise  
for moving the motor 2 anti clockwise  
for moving the motor 2 clockwise  
enable for motor 2  
  
LEDR[0]  
LEDR[1]  
end  
  
else if (y >  
begin  
GPIO_0[0] <= 1;//  
GPIO_0[2] <= 1;//  
GPIO_0[4] <= 0;//  
GPIO_0[1] <= 0;//  
GPIO_0[3] <= 1;//  
GPIO_0[5] <= 1;//  
LEDR[0] <= 1;  
LEDR[1] <= 1;  
end  
  
else if (y >  
begin  
GPIO_0[0] <= 1;  
GPIO_0[2] <= 1;  
GPIO_0[4] <= 0;  
GPIO_0[1] <= 0;  
GPIO_0[3] <= 1;  
GPIO_0[5] <= 1;  
LEDR[0]  
LEDR[1]  
end  
  
else if (y >  
begin  
GPIO_0[0] <= 1;//  
GPIO_0[2] <= 1;//  
GPIO_0[4] <= 0;//  
GPIO_0[1] <= 0;//  
GPIO_0[3] <= 1;//  
GPIO_0[5] <= 1;//
```

ECE 241

```
LEDR[0] <= 1;
LEDR[1] <= 1;
end
end
8'h43: // I
begin
    v <= v + 1;
    if (v >
        begin
            begin
                LEDR[0] <= 0;
                LEDR[1] <= 0;
                GPIO_0[0] <= 0;//enable for motor 1
                GPIO_0[5] <= 0;//enable for motor 2
                v <= 0;
                w = w + 1;
                if (w == 0 && v < 200000000) //uturn
                begin
                    GPIO_0[0] <= 0;//enable for motor 1
                    GPIO_0[2] <= 0;//for moving the motor 1 clockwise
                    GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
                    GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
                    GPIO_0[3] <= 1;//for moving the motor 2 clockwise
                    GPIO_0[5] <= 1;//enable for motor 2
                    <= 0;
                    <= 1;
                end
                else if (v > 200000000 && w == 0 && v < 275000000)//forward
                begin
                    enable for motor 1
                    for moving the motor 1 clockwise
                    for moving the motor 1 anti clockwise
                    for moving the motor 2 anti clockwise
                    for moving the motor 2 clockwise
                    enable for motor 2
                    LEDG[1]
                    LEDG[0]
                end
            end
        end
    end
    else if (v > 275000000 && w == 0 && v < 350000000)//backward
    begin
        enable for motor 1
        for moving the motor 1 clockwise
        for moving the motor 1 anti clockwise
        for moving the motor 2 anti clockwise
        for moving the motor 2 clockwise
        enable for motor 2
        LEDG[0]
        LEDG[1]
    end
end
end
```

ECE 241

```
end

else if (v
> 275000000 && w == 0 && v < 375000000)//right
begin

GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[2] <= 0;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
<= 0;
<= 1;

LEDR[0]
LEDR[1]
end

else if (v
> 375000000 && w == 0 && v < 475000000)//forward
begin
enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2
<= 1;
<= 1;

GPIO_0[0] <= 1;//
GPIO_0[2] <= 1;//
GPIO_0[4] <= 0;//
GPIO_0[1] <= 0;//
GPIO_0[3] <= 1;//
GPIO_0[5] <= 1;//
LEDR[0] <= 1;
LEDR[1] <= 1;

end

else if (v
> 475000000 && w == 0 && v < 575000000)//left
begin

GPIO_0[0] <= 1;//enable for motor 1
GPIO_0[2] <= 1;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 0;//for moving the motor 2 clockwise
GPIO_0[5] <= 0;//enable for motor 2
<= 0;
<= 1;

LEDR[0]
LEDR[1]
```

ECE 241

```

> 575000000 && w == 0 && v < 625000000)//forward
begin
enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2
end
200000000)
LEDR[0] <= 0;
LEDR[1] <= 0;
GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[5] <= 0;//enable for motor 2
x <= 0;
z = z +1;
25000000 && z == 0) //forward
enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2
25000000 && z == 0 && x < 125000000) //right
else if (v
else if (v
GPIO_0[0] <= 1;//
GPIO_0[2] <= 1;//
GPIO_0[4] <= 0;//
GPIO_0[1] <= 0;//
GPIO_0[3] <= 1;//
GPIO_0[5] <= 1;//
LEDR[0] <= 1;
LEDR[1] <= 1;
end
8'h1C: //A
begin
x <= x + 1;
if (x >
begin
begin
end
else if (x <
begin
GPIO_0[0] <= 1;//
GPIO_0[2] <= 1;//;
GPIO_0[4] <= 0;//;
GPIO_0[1] <= 0;//;
GPIO_0[3] <= 1;//;
GPIO_0[5] <= 1;//;
LEDR[0] <= 1;
LEDR[1] <= 1;
end
else if (x >
begin

```

ECE 241

```
GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[2] <= 0;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
=< 0;
=< 1;
125000000 && z == 0 && x < 200000000) //forward
enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2
LEDR[0] <= 1;
LEDR[1] <= 1;
end
else if (x <
begin
GPIO_0[0] <= 1;//
GPIO_0[2] <= 1;//
GPIO_0[4] <= 0;//
GPIO_0[1] <= 0;//
GPIO_0[3] <= 1;//
GPIO_0[5] <= 1;//
LEDR[0] <= 1;
LEDR[1] <= 1;
end
end
8'h32: //B
begin
a <= a + 1;
if (a >
begin
600000000)
LEDG[0] <= 0;
LEDG[1] <= 0;
GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[5] <= 0;//enable for motor 2
a <= 0;
b = b +1;
200000000 && b == 0) //u turn
end
else if (a <
begin
GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[2] <= 0;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
```

ECE 241

```
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
<= 0; LEDG[1]
<= 1; LEDG[0]
end
else if (a >
200000000 && a < 275000000 && b == 0) // forward
begin
GPIO_0[0] <= 1;//enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2
LEDR[0] <= 1;
LEDR[1] <= 1;
end
else if (a >
275000000 && a < 375000000 && b == 0) // right
begin
GPIO_0[0] <= 0;//enable for motor 1
GPIO_0[2] <= 0;//for moving the motor 1 clockwise
GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise
GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise
GPIO_0[3] <= 1;//for moving the motor 2 clockwise
GPIO_0[5] <= 1;//enable for motor 2
<= 0; LEDR[0]
<= 1; LEDR[1]
end
else if (a >
375000000 && a < 475000000 && b == 0) // forward
begin
GPIO_0[0] <= 1;//enable for motor 1
for moving the motor 1 clockwise
for moving the motor 1 anti clockwise
for moving the motor 2 anti clockwise
for moving the motor 2 clockwise
enable for motor 2

```

ECE 241

```
LEDR[0] <= 1;
LEDR[1] <= 1;
    end
else if (a >
475000000 && a < 575000000 && b == 0) // left
begin

GPIO_0[0] <= 1;//enable for motor 1

GPIO_0[2] <= 1;//for moving the motor 1 clockwise

GPIO_0[4] <= 0;//for moving the motor 1 anti clockwise

GPIO_0[1] <= 0;//for moving the motor 2 anti clockwise

GPIO_0[3] <= 0;//for moving the motor 2 clockwise

GPIO_0[5] <= 0;//enable for motor 2

<= 0;
    LEDR[0]
<= 1;
    LEDR[1]
        end
else if (a >
575000000 && a < 625000000 && b == 0) // forward
begin

enable for motor 1

for moving the motor 1 clockwise

for moving the motor 1 anti clockwise

for moving the motor 2 anti clockwise

for moving the motor 2 clockwise

enable for motor 2

    GPIO_0[0] <= 1;//
    GPIO_0[2] <= 1;//
    GPIO_0[4] <= 0;//
    GPIO_0[1] <= 0;//
    GPIO_0[3] <= 1;//
    GPIO_0[5] <= 1;//

    LEDR[0] <= 1;
    LEDR[1] <= 1;
        end
endcase
end
//*****************************************************************************
*          Combinational Logic
//*************************************************************************/
assign HEX2 = 7'h7F;
assign HEX3 = 7'h7F;
assign HEX4 = 7'h7F;
assign HEX5 = 7'h7F;
assign HEX6 = 7'h7F;
assign HEX7 = 7'h7F;
//*****************************************************************************
*          Internal Modules
//*************************************************************************/
PS2_Controller PS2 (
    // Inputs
    .CLOCK_50           (CLOCK_50),
    .reset              (~KEY[0]),
    // Bidirectionals
    .PS2_CLK            (PS2_CLK),
    .PS2_DAT            (PS2_DAT),
    // Outputs
```

ECE 241

```
.received_data          (ps2_key_data),
.received_data_en (ps2_key_pressed)
);
//assign LEDR = last_data_received;
Hexadecimal_To_Seven_Segment Segment0 (
    // Inputs
    .hex_number           (last_data_received[3:0]),
    // Bidirectional
    // Outputs
    .seven_seg_display   (HEX0)
);
Hexadecimal_To_Seven_Segment Segment1 (
    // Inputs
    .hex_number           (last_data_received[7:4]),
    // Bidirectional
    // Outputs
    .seven_seg_display   (HEX1)
);
endmodule
```