Report for Embedded Systems Lecture’s Project 1
Table Tennis

Name: Muaz Kurt
Number: 151044062

Approch

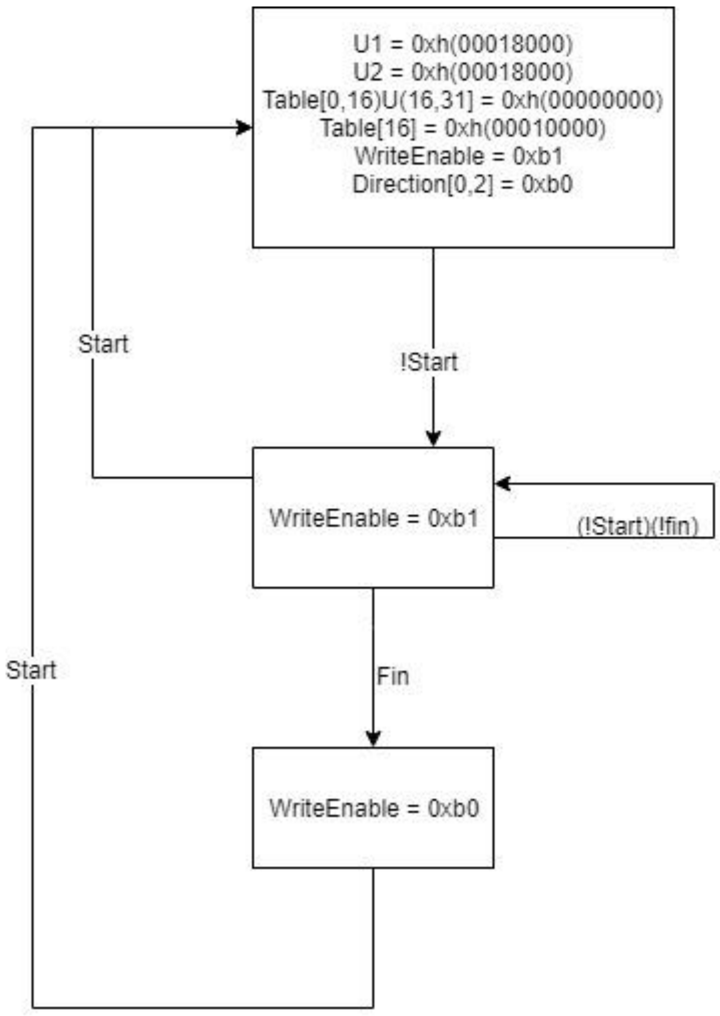
1. There is a button to Reset the game.
2. There are 2 Users and a table for playing game.
3. Each user is a 32 bit path for move as up&down way.
4. For movement, users have a joystick. Users can move in their area by pulling their own joystick to up and down.
5. Users are placed as left and right. If left user pulls it's own joystick to left, then it will be ignored. If he/she pulls the joystick to right then it has a meaning and it will be described later. The same issue exist for Right user for pulling the joysick to right/left.
6. Table is 32 bitted 32 rows. Each bit is setted as 0 expect the ball. If the ball is on one of the bits, then it will be setted as 1.
7. The ball initially moves left/right. When it is at the corner of the row (0th or 32th bit) then it checks if user's position (as up-down) and its position (up-down) are matching. If it doesn't match, this means the game is over, don't allow any movement further. Otherwise the user it matches can return the ball to other direction. If matching user doesn't move it's joystick, then ball returns as the mirror return. This means: / goes \, -> goes <-, \ goes /. Otherwise, if joystick is pulled a side but it doesn't be ignored, then for left user pulling right-top means return / way and right direction, pulling just right means return – way and right, pulling right-down means return \ way and right direction. For right user, pulling left-up means \ way and left direction, pulling left means – way and left direction, pulling left-down means / way and left direction.
8. For moving of the ball is \ or /, it can reach the top of table. If it reaches, it should not drop away from table. For this issue, if it reaches the top or bottom, then it return as the mirror return and goes the same direction (left/right).
9. If game is over, then registers can not be updated. So there will be no movements further.

C CODE

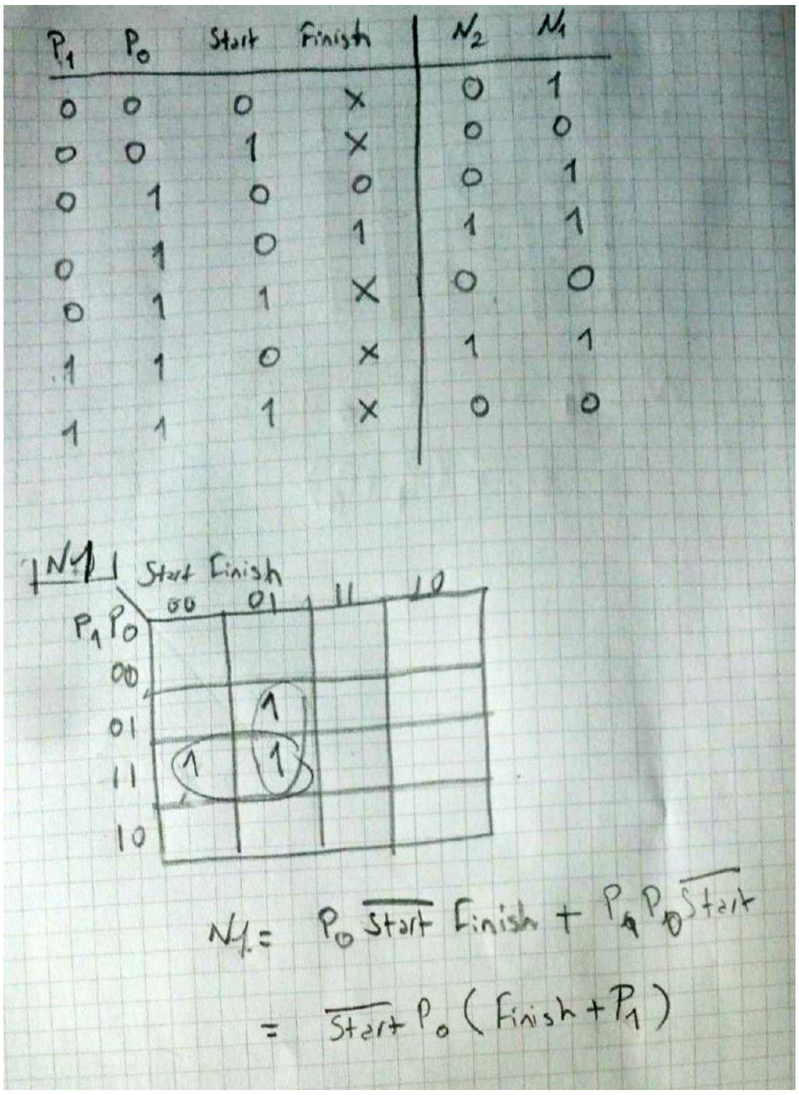
```c
/**
    This is a template C code for the implementation of Table Tenis Game on
Logisim.
    Muaz Kurt - 151044062
    muazkurt@gmail.com
*/
int main(void)
{
    int[32][32] table;
    int[32]     U1 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} // 0 -> Top, 31 -> Bottom
    int[32]     U2 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0} // ######################
    int         risk_left   = 0,
                risk_right  = 0;
    int[3]      direction   = {0, 0, 0};
    int[2]      js_left_x   = {0, 0},
                js_left_y   = {0, 0},
                js_right_x  = {0, 0}
                js_right_y  = {0, 0};
    int         fin         = 0;
    int         start       = 0;
    int         upper_bound = 0,
                lower_bound = 0;
    do
    {
S0:     direction = {0, 0, 0};
        U1[16] = U1[17] = 1;
        U2[16] = U1[17] = 1;
        table[16][0] = 1;
        while(start);
        while(!fin)
        {
            if(start) goto S0;
            for(int i = 0; i < 32; ++i)
            {
                risk_left |= table[31][i];
                risk_right |= table[0][i];
            }

            if(risk_left)
            {
                int handle = 0;
                for(int i = 0; i < 32; ++i)
                    handle |= table[i][0] & U1[i]
                if(!handle)
                {
                    while( !direction[0]
                        && !direction[1]
                        && !direction[2]
                        && (
                            !js_left_x[1] ||
                            (!js_left_x[0] && js_left_x[1] && js_left_y[1] &&
!js_left_y[0]))
                        );

                    if(js_left_x[1] & !js_left_x[0])
                    {
                        if(js_left_y[0] & !js_left_y[1] & !U1[0])
                        {
                            for(i = 1; i < 31; ++i)
                            {
                                U1[i - 1] = U1[i];
                                table[i - 1][0] = table[i][0];
                            }
                            U1[31] = 0;
                            table[31][0] = 0;
                        }
                        else if(js_left_y[0] & js_left_y[1] & !U1[31])
                        {
                            for(i = 31; i > 0; --i)
                            {
                                U1[i] = U1[i - 1];
                                table[i][0] = table[i - 1][0];
                            }
                            U1[0] = 0;
                            table[0][0] = 0;
                        }
                    }
                    else
                    {
                        direction[2] = !direction[2];
                        if( (js_left_y[1] && !js_left_y[0] && js_left_x[1] &&
!js_left_x[0]) ||
                            !js_left_x[1]);
                        else if(js_left_x[0] & js_left_x[1] & js_left_y[0] &
!js_left_y[1])
                        {
                            direction[0] = 1;
                            direction[1] = 0;
                        }
                        else if(js_left_x[0] & js_left_x[1] & !js_left_y[0] &
js_left_y[1])
                        {
                            direction[0] = 1;
                            direction[1] = 1;
                        }
                        else if(js_left_x[0] & js_left_x[1] & js_left_y[0] &
js_left_y[1])
                        {
                            direction[0] = 0;
                            direction[1] = 1;
                        }
                    }
                }
                else
                    fin = 1;
            }
            else if(risk_right)
            {
                int handle = 0;
                for(int i = 0; i < 32; ++i)
                    handle |= table[i][31] & U1[i]
                if(!handle)
                {
                    while( !direction[0]
                        && !direction[1]
                        && !direction[2]
                        && (
                             (js_right_x[0] & js_right_x[1]) ||
                            (!js_right_x[0] &&
                              js_right_x[1] &&
                              js_right_y[1] &&
                             !js_right_y[0]))
                        );
                    if(js_right_x[1] & !js_right_x[0])
                    {
                        if(js_right_y[0] & !js_right_y[1] & !U2[0])
                        {
                            for(i = 1; i < 31; ++i)
                            {
                                U2[i - 1] = U2[i];
                                table[i - 1][31] = table[i][31];
                            }
                            U1[31] = 0;
                            table[31][31] = 0;
                        }
                        else if(js_right_y[0] & js_right_y[1] & !U2[31])
                        {
                            for(i = 31; i < 0; ++i)
                            {
                                U2[i] = U2[i - 1];
                                table[i][31] = table[i - 1][31];
                            }
                            U2[0] = 0;
                            table[0][31] = 0;
                        }
                    }
                    else
                    {
                        direction[2] != direction[2];
                        if( (js_right_y[1] && !js_right_y[0] && js_right_x[1]
&& !js_right_x[0]) ||
                            (js_right_x[1] & js_right_x[1]));
                        else if(!js_right_x[1] & js_left_y[0] &
!js_right_y[1])
                        {
                            direction[0] = 1;
                            direction[1] = 0;
                        }
                        else if(!js_right_x[1] & !js_right_y[0] &
js_right_y[1])
                        {
                            direction[0] = 1;
                            direction[1] = 1;
                        }
                        else if(!js_right_x[1] & js_right_y[0] &
js_right_y[1])
                        {
                            direction[0] = 0;
                            direction[1] = 1;
                        }
                    }
                }
                else
                    fin = 1;
            }
            upper_bound = lower_bound = 0;
            for(int i = 0; i < 32; ++i)
            {
                upper_bound |= table[0][i];
                lower_bound |= table[31][i];
            }
            if(upper_bound || lower_bound)
            {
                int temp = direction[0];
                direction[0] = direction[1];
                direction[1] = temp;
            }
            if(direction[2])
                if(direction[1])
                    if(direction[0])            //
<<<<<<<<<<<<<<<<<<<-     111
                        for(int i = 0; i < 32; ++i)
```

```
                            for(int y = 0; y < 32; ++y)
                                if(y == 31)
                                    table[i][y] = 0
                                else
                                    table[i][y] = table[i][y + 1];
                    else                            //
///////////////////-     110
                        for(int i = 0; i < 32; ++i)
                            for(int y = 0; y < 32; ++y)
                                if(i == 0 || y == 31)
                                    table[i][y] = 0;
                                else
                                    table[i][y] = table[i - 1][y + 1];
                else
                    if(direction[0])            //
\\\\\\\\\\\\\\\\\\\-     101
                        for(int i = 0; i < 32; ++i)
                            for(int y = 0; y < 32; ++y)
                                if(i == 31 || y == 31)
                                    table[i][y] = 0;
                                else
                                    table[i][y] = table[i + 1][y + 1];
            else
                if(direction[1])
                    if(direction[0])            // -
>>>>>>>>>>>>>>>>>>      011
                        for(int i = 0; i < 32; ++i)
                            for(int y = 0; y < 32; ++y)
                                if(y == 0)
                                    table[i][y] = 0
                                else
                                    table[i][y] = table[i][y - 1];
                    else                            // -
\\\\\\\\\\\\\\\\\\      010
                        for(int i = 0; i < 32; ++i)
                            for(int y = 0; y < 32; ++y)
                                if(i == 0 || y == 0)
                                    table[i][y] = 0;
                                else
                                    table[i][y] = table[i - 1][y - 1];
                else
                    if(direction[0])            // -
//////////////////      001
                        for(int i = 0; i < 32; ++i)
                            for(int y = 0; y < 32; ++y)
                                if(i == 31 || y == 0)
                                    table[i][y] = 0;
                                else
                                    table[i][y] = table[i + 1][y - 1];
            if (js_right_x[1] & !js_right_x[0])
            {
                if(js_right_y[0])
                {
                    if(js_right_y[1])       // U2 >> 1
                        for(int i = 0; i < 32; ++i)
                            if(i == 31)
                                U2[i] = U2[i];
                            else
                                U2[i] = U2[i - 1];
                    else                    // U2 << 1
                        for(int i = 0; i < 32; ++i)
                            if(i == 0)
                                U2[i] = U2[i];
                            else
                                U2[i] = U2[i + 1]
                }
            }
            if (js_left_x[1] & !js_left_x[0])
            {
                if(js_left_x[0])
                {
                    if(js_left_x[1])
                        for(int i = 0; i < 32; ++i)
                            if(i == 31)
                                U1[i] = U1[i];
                            else
                                U1[i] = U1[i - 1];
                    else
                        for(int i = 0; i < 32; ++i)
                            if(i == 0)
                                U1[i] = U1[i];
                            else
                                U1[i] = U1[i + 1]
                }
            }
        }
        while(!start);
    } while(true);
}
```

FSM


U1 = 0xh(00018000)
U2 = 0xh(00018000)
Table[0,16)U(16,31] = 0xh(00000000)
Table[16] = 0xh(00010000)
WriteEnable = 0xb1
Direction[0,2] = 0xb0
Start
!Start
WriteEnable = 0xb1
(!Start)(!fin)
Start
Fin
WriteEnable = 0xb0

SIMPLIFICATION

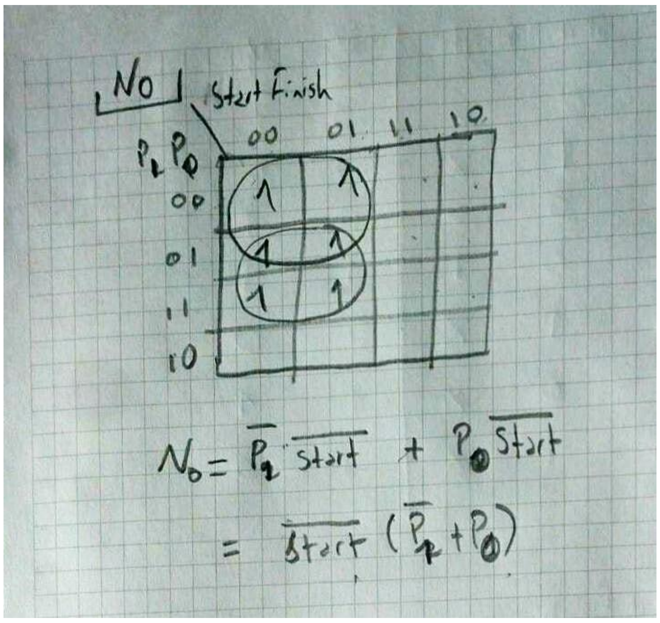


P_1	P_0	Start	Finish	N_2	N_1
0	0	0	X	0	1
0	0	1	X	0	0
0	1	0	0	0	1
0	1	0	1	1	1
0	1	1	X	0	0
1	1	0	X	1	1
1	1	1	X	0	0

N1

$P_1 P_0$ \ Start Finish	00	01	11	10
00				
01		1		
11	1	1		
10				

$$N_1 = P_0 \overline{Start}\, Finish + P_1 P_0 \overline{Start}$$
$$= \overline{Start}\, P_0 (Finish + P_1)$$



N0

$P_1 P_0$ \ Start Finish	00	01	11	10
00	1	1		
01	1	1		
11	1	1		
10				

$$N_0 = \overline{P_1}\,\overline{Start} + P_0 \overline{Start}$$
$$= \overline{Start}\,(\overline{P_1} + P_0)$$