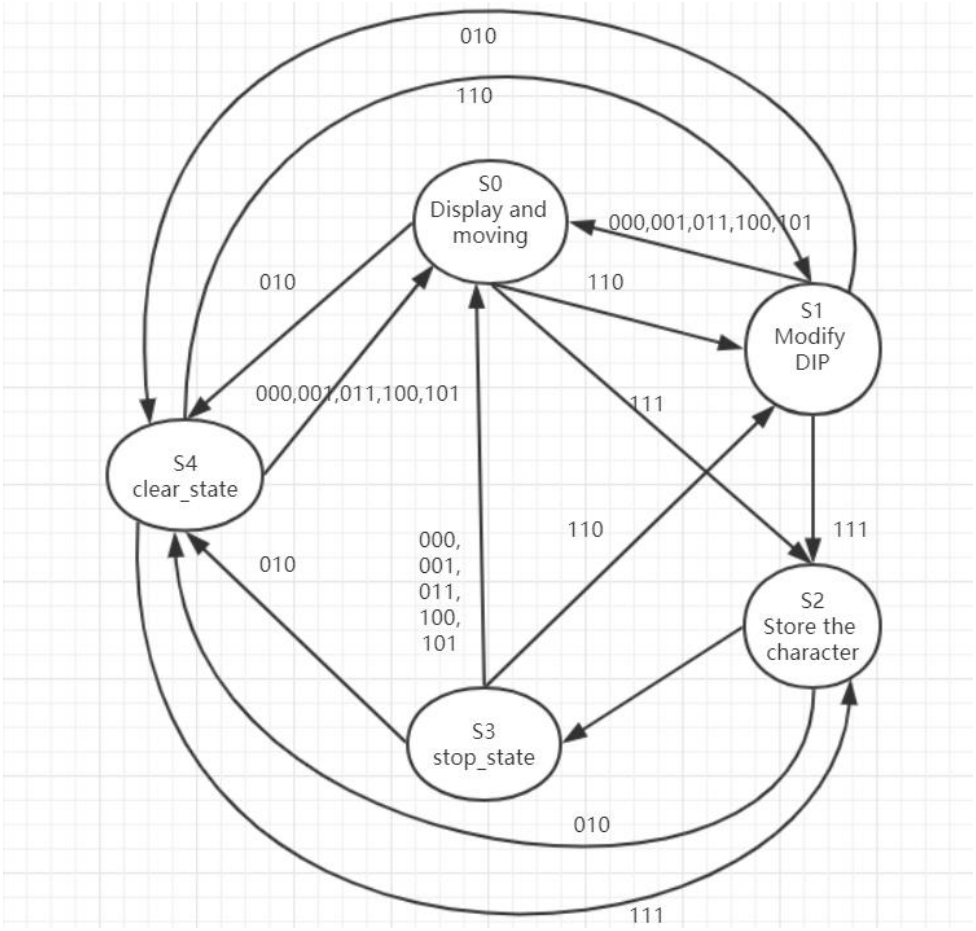
LAB 6 notebook

Jiajun, Guan

**a). Comments on the storing and moving text design:**




For the state machine diagram, the input corresponding to the button. When (BTN2, BTN1, BTN0) = (110), it would be the state 1. The user modifies the switches to the letter they want to be loaded and stored. Then, when (BTN2, BTN1) = (11) and BTN0 goes from 0 to 1, the state is S2 and the letter is loaded and stored into the proper register, in my program which is noted as “store”. To load and store the next letter the user only needs to set BTN0 = 0 again modify the switches, and then to switch the BTN0 from 0 to 1. When the letter is stored, the state machine will immediately jump into S3. If input is 010, the program will enter state S4, and all the letter stored in the register would be cleared. In the other case, the FPGA will stay at S0 and display all the letters in store and moving the text.

B) RTL file and testbench file

```
module lab4_top(


    output reg [6:0] seven_segl,
    output reg [3:0] Activate,
    input [6:0] a,
    input clk,
    input [2:0] btn


    );



    wire cout;   //Used to connect the clock_divider and LFSR
    wire coutl;
    wire cout2;
    reg [1:0] counter = 2'b00;
    reg [4:0] slow_counter =5'b00000;
    reg [6:0] store [0:19] ={ 7'b1111111,7'b1111111,7'b1111111,7'b1111111,7'b1111111,
                    7'b1111111,7'b1111111,7'b1111111,7'b1111111,7'b1111111,
                    7'b1111111,7'b1111111,7'b1111111,7'b1111111,7'b1111111,
                    7'b1111111,7'b1111111,7'b1111111,7'b1111111,7'b1111111};

    parameter S0 = 3'b000, S1 = 3'b001,S2 = 3'b010, S3 = 3'b011, S4 = 3'b100;

    reg [2:0] state = 3'b000;
    reg [2:0] next_state = 3'b000;

    integer i = 0;


    always @(posedge coutl)
    begin
       if (counter > 2'b11) counter <= 2'b00;
       counter <= counter+1;
    end

    always @(posedge cout)
    begin
        if (state == S0) begin
            if (slow_counter == input_count) begin slow_counter <= 5'b00000; end
            else slow_counter <= slow_counter+1; end
        else if (state != S0)
            slow_counter <= 5'b00000;
    end
```

```
always @(*)
begin
    case(counter)
    2'b00: begin
        Activate = 4'b0111;
        // activate LED1 and Deactivate LED2, LED3, LED4

        case (state)
            S0:
                if (input_count>4) begin
                    if (slow_counter==input_count)
                        seven_segl = 7'b1111111;
                    else
                        seven_segl = store[slow_counter];
                end

                else begin
                    seven_segl = store[0];
                end

            S1:
                seven_segl = 7'b1110001;
            S2:
                seven_segl = 7'b1100010;
            S3:
                seven_segl = 7'b1100010;

        endcase
        end
     2'b01: begin
        Activate = 4'b1011;
        // activate LED2 and Deactivate LED1, LED3, LED4

        case (state)
            S0:
            if (input_count>4) begin
                if((slow_counter+1)== input_count)
                    seven_segl = 7'b1111111;
                else if (slow_counter == input_count)
                    seven_segl = store[0];
                else
                    seven_segl = store[slow_counter+1];
```

```
        else begin
            seven_segl = store[1];
        end

        S1:
            seven_segl = 7'b1111111;
        S2:
            seven_segl = 7'b1111111;
        S3:
            seven_segl = 7'b1111111;
    endcase

    end
2'b10: begin
    Activate = 4'b1101;
    // activate LED3 and Deactivate LED2, LED1, LED4

    case (state)
        S0:
        if (input_count >4) begin
            if((slow_counter+2)== input_count)
                seven_segl = 7'b1111111;
            else if ((slow_counter+1)== input_count)
                seven_segl = store[0];
            else if (slow_counter== input_count)
                    seven_segl = store[1];
            else
                 seven_segl = store[slow_counter+2];

        end

        else begin
            seven_segl = store[2];
        end

        S1:
            seven_segl = 7'b1111111;
        S2:
            seven_segl = 7'b1111111;
        S3:
            seven_segl = 7'b1111111;
    endcase
```

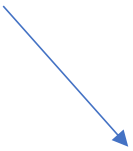
```
        end
    2'b11: begin
        Activate = 4'b1110;
        // activate LED4 and Deactivate LED2, LED3, LED1
        case (state)
            S0:
            if (input_count >4) begin
                if((slow_counter+3)== input_count)
                    seven_segl = 7'b1111111;
                else if ((slow_counter+2)== input_count)
                    seven_segl = store[0];
                else if ((slow_counter+1)== input_count)
                    seven_segl = store[1];
                else if (slow_counter == input_count)
                    seven_segl = store[2];
                else
                    seven_segl = store[slow_counter+3];
            end

            S1:
                seven_segl = a;
            S2:
                seven_segl = a;
            S3:
                seven_segl = a;
        endcase


        end

    endcase
end



always @(*) //q,state?
begin

    case(state)
        S0: begin
```

```
            S0: begin

                if (btn == 3'b110)
                    next_state = S1;

                else if (btn == 3'b111)
                    next_state = S2;

                else if (btn == 3'b010)
                    next_state = S4;

                else
                    next_state = S0;

            end

            S1: begin
                if (btn == 3'b111)
                    next_state = S2;
                else if (btn == 3'b110) begin
                    next_state = S1;    end
                else if (btn == 3'b010)
                    next_state = S4;
                else
                    next_state = S0;

            end

            S2: begin

                next_state = S3;
            end

            S3: begin
                if (btn == 3'b110)
                    next_state = S1;

                else if (btn == 3'b111) begin
                    next_state = S3;
                    end

                else if (btn == 3'b010)
                    next state = S4;
```

```
            else if (btn == 3'b010)
                next_state = S4;

            else
                next_state = S0;

            end



        endcase

    end

    always @ (posedge cout2)
    begin
        if (state == S2)begin
            store[input_count] <= a;
            input_count <= input_count+1;

        end

        else if (state == S4) begin
            input_count <= 0;
            for (i=0;i<20;i=i+1) begin
                store[i]<=7'b1111111;
            end
        end

        else begin  end

        state <= next_state;
    end

endmodule
```

2. Testbench file

```
module lab4_toptb( );

reg clk;
wire [3:0] Activate;
wire [6:0] seven_segl;
reg [7:0] a;
reg [2:0] btn;

always #0.3 clk = ~clk;

initial begin
clk = 0;
a = 7'b0110000;
a = #230 7'b1001100;
a = #100 7'b0000110;
a = #100 7'b0000001;
a = #100 7'b1001111;
end

initial begin
btn = 3'b000;
btn = #40 3'b110; //30
btn = #30 3'b111; //60
btn = #30 3'b000; //90
btn = #40 3'b110; //140
btn = #30 3'b111; //170
btn = #30 3'b000; //200

btn = #40 3'b110; //240
btn = #30 3'b111; //270
btn = #30 3'b000; //300

btn = #40 3'b110; //240
btn = #30 3'b111; //270
btn = #30 3'b000; //300

btn = #40 3'b110; //240
btn = #30 3'b111; //270
btn = #30 3'b000; //300

btn = #3660 3'b010; //240
btn = #30 3'b000; //300
end

lab4_top utb(seven_segl,Activate,a,clk,btn);
//led_segment LS(display,seven_seg1);

endmodule
```



If we only store 2 numbers, we delete these three blocks of code

C) Constrain file

```
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
# leds
set_property PACKAGE_PIN V17 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[0]}]
set_property PACKAGE_PIN V16 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[1]}]
set_property PACKAGE_PIN W16 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[2]}]
set_property PACKAGE_PIN W17 [get_ports {a[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[3]}]
set_property PACKAGE_PIN W15 [get_ports {a[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[4]}]
set_property PACKAGE_PIN V15 [get_ports {a[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[5]}]
set_property PACKAGE_PIN W14 [get_ports {a[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[6]}]


set_property PACKAGE_PIN U1 [get_ports {btn[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {btn[0]}]
set_property PACKAGE_PIN T1 [get_ports {btn[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {btn[1]}]
set_property PACKAGE_PIN R2 [get_ports {btn[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {btn[2]}]


set_property PACKAGE_PIN W7 [get_ports {seven_segl[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seven_segl[6]}]
set_property PACKAGE_PIN W6 [get_ports {seven_segl[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seven_segl[5]}]
set_property PACKAGE_PIN U8 [get_ports {seven_segl[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seven_segl[4]}]
set_property PACKAGE_PIN V8 [get_ports {seven_segl[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seven_segl[3]}]
set_property PACKAGE_PIN U5 [get_ports {seven_segl[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seven_segl[2]}]
set_property PACKAGE_PIN V5 [get_ports {seven_segl[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {seven_segl[1]}]
set property PACKAGE PIN U7 [get ports {seven segl[0]}]
```

```
set_property PACKAGE_PIN U2 [get_ports {Activate[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Activate[0]}]
set_property PACKAGE_PIN U4 [get_ports {Activate[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Activate[1]}]
set_property PACKAGE_PIN V4 [get_ports {Activate[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Activate[2]}]
set_property PACKAGE_PIN W4 [get_ports {Activate[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Activate[3]}]

#Buttons

create_clock -period 20.000 -name clk -waveform {0.000 10.000}
set_input_delay -clock [get_clocks clk] 0.000 [get_ports -filter { NAME =~ "*" && DIRECTION == "IN" }]
set_output_delay -clock [get_clocks clk] 0.000 [get_ports -filter { NAME =~ "*" && DIRECTION == "OUT" }]
```

D) 1.Utilization report

1. Slice Logic

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Slice LUTs* | 462 | 0 | 20800 | 2.22 |
| LUT as Logic | 462 | 0 | 20800 | 2.22 |
| LUT as Memory | 0 | 0 | 9600 | 0.00 |
| Slice Registers | 555 | 0 | 41600 | 1.33 |
| Register as Flip Flop | 545 | 0 | 41600 | 1.31 |
| Register as Latch | 10 | 0 | 41600 | 0.02 |
| F7 Muxes | 14 | 0 | 16300 | 0.09 |
| F8 Muxes | 7 | 0 | 8150 | 0.09 |

2. Memory

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Block RAM Tile | 0 | 0 | 50 | 0.00 |
| RAMB36/FIFO* | 0 | 0 | 50 | 0.00 |
| RAMB18 | 0 | 0 | 100 | 0.00 |

* Note: Each Block RAM Tile only has one FIFO logic availab

3. DSP

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| DSPs | 0 | 0 | 90 | 0.00 |

4. IO and GT Specific

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| Bonded IOB | 22 | 0 | 106 | 20.75 |
| Bonded IPADs | 0 | 0 | 10 | 0.00 |
| Bonded OPADs | 0 | 0 | 4 | 0.00 |
| PHY_CONTROL | 0 | 0 | 5 | 0.00 |
| PHASER_REF | 0 | 0 | 5 | 0.00 |
| OUT_FIFO | 0 | 0 | 20 | 0.00 |
| IN_FIFO | 0 | 0 | 20 | 0.00 |
| IDELAYCTRL | 0 | 0 | 5 | 0.00 |
| IBUFDS | 0 | 0 | 104 | 0.00 |
| GTPE2_CHANNEL | 0 | 0 | 2 | 0.00 |
| PHASER_OUT/PHASER_OUT_PHY | 0 | 0 | 20 | 0.00 |
| PHASER_IN/PHASER_IN_PHY | 0 | 0 | 20 | 0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY | 0 | 0 | 250 | 0.00 |
| IBUFDS_GTE2 | 0 | 0 | 2 | 0.00 |
| ILOGIC | 0 | 0 | 106 | 0.00 |
| OLOGIC | 0 | 0 | 106 | 0.00 |

5. Clocking

| Site Type | Used | Fixed | Available | Util% |
|---|---|---|---|---|
| BUFGCTRL | 2 | 0 | 32 | 6.25 |
| BUFIO | 0 | 0 | 20 | 0.00 |
| MMCME2_ADV | 0 | 0 | 5 | 0.00 |
| PLLE2_ADV | 0 | 0 | 5 | 0.00 |
| BUFMRCE | 0 | 0 | 10 | 0.00 |
| BUFHCE | 0 | 0 | 72 | 0.00 |
| BUFR | 0 | 0 | 20 | 0.00 |

2.Power report

1. Summary

| | |
|---|---|
| Total On-Chip Power (W) | 0.079 |
| Dynamic (W) | 0.008 |
| Device Static (W) | 0.072 |
| Effective TJA (C/W) | 5.0 |
| Max Ambient (C) | 84.6 |
| Junction Temperature (C) | 25.4 |
| Confidence Level | Low |
| Setting File | --- |
| Simulation Activity File | --- |
| Design Nets Matched | NA |

3.Timing report

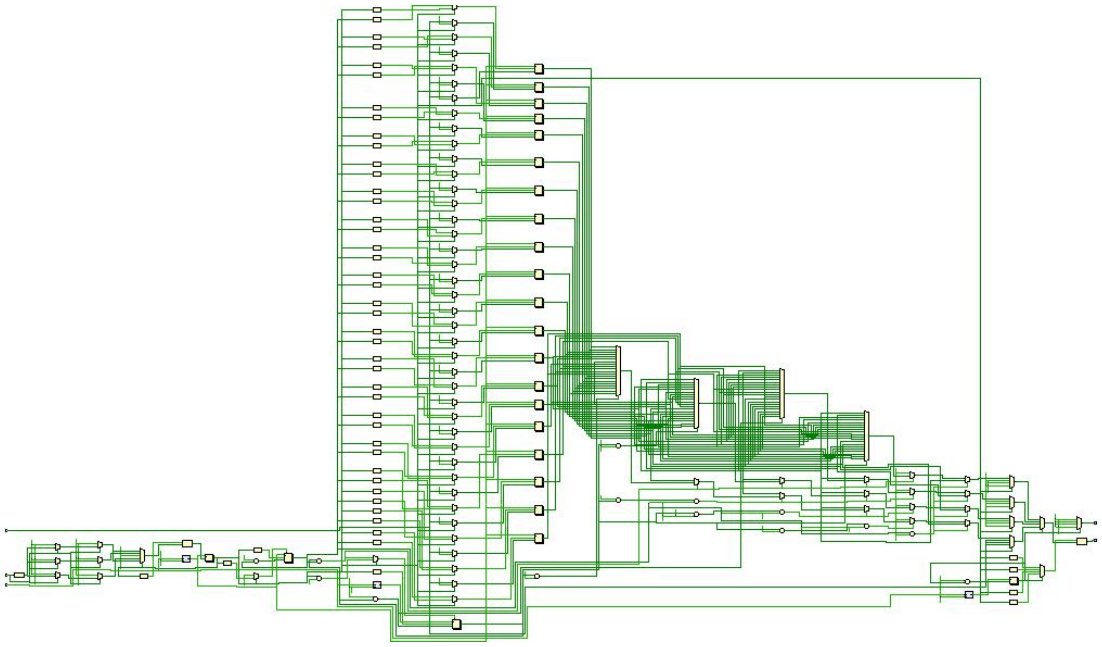
Design Timing Summary

| WNS(ns) | TNS(ns) | TNS Failing Endpoints | TNS Total Endpoints | WHS(ns) | THS(ns) | T |
|---|---|---|---|---|---|---|
| 3.606 | 0.000 | 0 | 1062 | 0.133 | 0.000 | - |

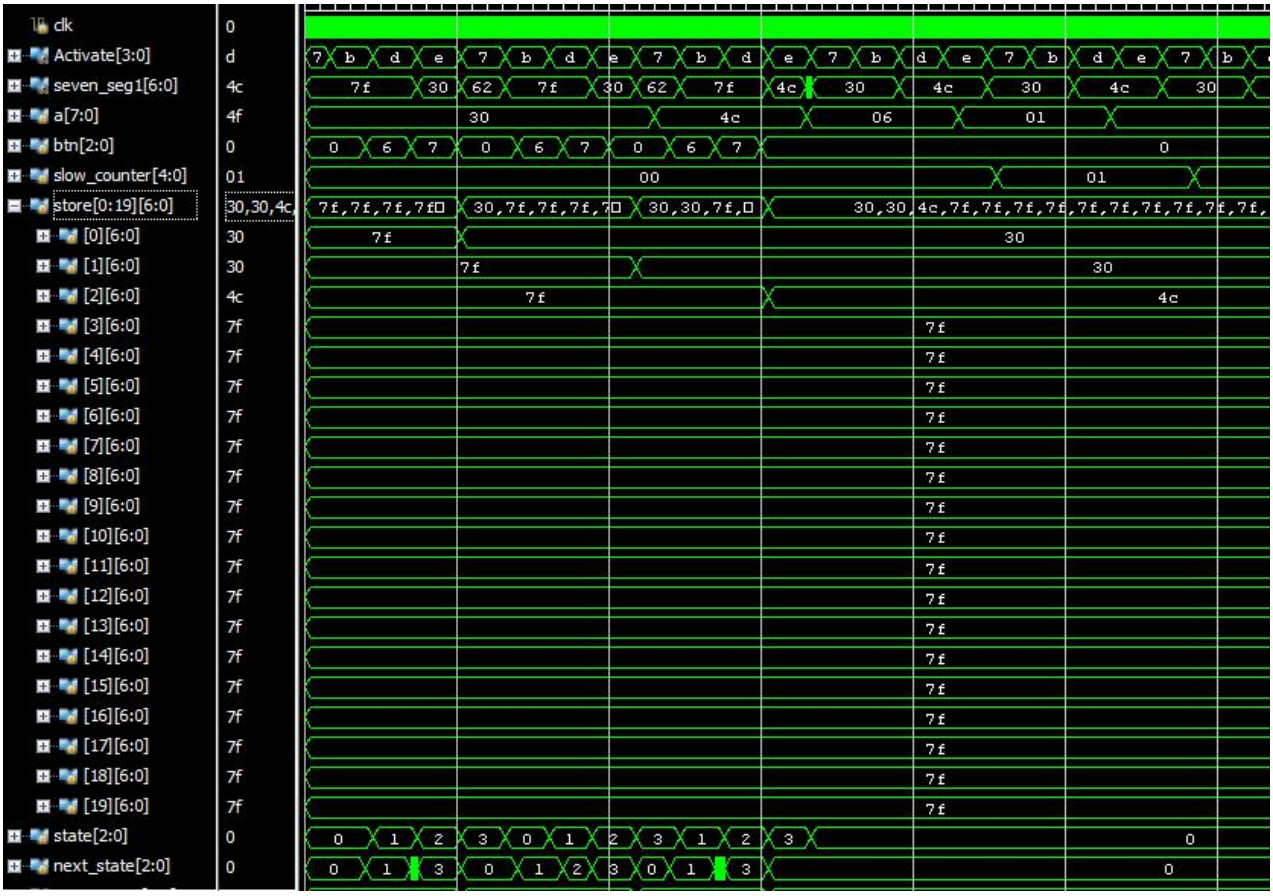
| THS Failing Endpoints | THS Total Endpoints | WPWS(ns) | TPWS(ns) | TPWS Failing Endpoints | TPWS Total Endpoints |
|---|---|---|---|---|---|
| 0 | 1062 | 4.500 | 0.000 | 0 | 391 |

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             3.606ns  (required time - arrival time)
  Source:                 CDIV/count0_reg[26]/C
                            (rising edge-triggered cell FDRE clocked by sys_clk_pin  {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            CDIV/count3_reg[20]/R
                            (rising edge-triggered cell FDRE clocked by sys_clk_pin  {rise@0.000ns fall@5.000ns period=10.000ns})
  Path Group:             sys_clk_pin
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            10.000ns  (sys_clk_pin rise@10.000ns - sys_clk_pin rise@0.000ns)
  Data Path Delay:        5.807ns  (logic 0.952ns (16.394%)  route 4.855ns (83.606%))
  Logic Levels:           4  (LUT4=2 LUT5=2)
  Clock Path Skew:        -0.028ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    4.776ns = ( 14.776 - 10.000 )
    Source Clock Delay      (SCD):    5.078ns
    Clock Pessimism Removal (CPR):    0.274ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns

                        (clock sys_clk_pin rise edge)
                                                    0.000     0.000 r
  W5                                                0.000     0.000 r  clk (IN)
                        net (fo=0)                  0.000     0.000    clk
  W5                    IBUF (Prop_ibuf_I_O)        1.458     1.458 r  clk_IBUF_inst/O
                        net (fo=1, routed)          1.967     3.425    clk_IBUF
  BUFGCTRL_X0Y0         BUFG (Prop_bufg_I_O)        0.096     3.521 r  clk_IBUF_BUFG_inst/O
                        net (fo=390, routed)        1.557     5.078    CDIV/clk_IBUF_BUFG
  SLICE_X55Y21          FDRE                                       r  CDIV/count0_reg[26]/C
----------------------------------------------------------------------   -------------------
  SLICE_X55Y21          FDRE (Prop_fdre_C_Q)        0.456     5.534 r  CDIV/count0_reg[26]/Q
                        net (fo=2, routed)          0.981     6.515    CDIV/count0_reg[26]
  SLICE_X57Y18          LUT4 (Prop_lut4_I2_O)       0.124     6.639 r  CDIV/count0[0]_i_11__1/O
                        net (fo=1, routed)          0.520     7.159    CDIV/count0[0]_i_11__1_n_0
  SLICE_X56Y18          LUT5 (Prop_lut5_I4_O)       0.124     7.283 r  CDIV/count0[0]_i_3__1/O
                        net (fo=1, routed)          0.689     7.973    CDIV/count0[0]_i_3__1_n_0
  SLICE_X56Y18          LUT4 (Prop_lut4_I0_O)       0.124     8.097 f  CDIV/count0[0]_i_1__1/O
                        net (fo=68, routed)         1.554     9.650    CDIV/count0[0]_i_1__1_n_0
  SLICE_X53Y20          LUT5 (Prop_lut5_I0_O)       0.124     9.774 r  CDIV/count3[0]_i_1__1/O
                        net (fo=32, routed)         1.111    10.885    CDIV/clear
  SLICE_X54Y24          FDRE                                       r  CDIV/count3_reg[20]/R
----------------------------------------------------------------------   -------------------

                        (clock sys_clk_pin rise edge)
                                                   10.000    10.000 r
  W5                                                0.000    10.000 r  clk (IN)
                        net (fo=0)                  0.000    10.000    clk
  W5                    IBUF (Prop_ibuf_I_O)        1.388    11.388 r  clk_IBUF_inst/O
                        net (fo=1, routed)          1.862    13.250    clk_IBUF
  BUFGCTRL_X0Y0         BUFG (Prop_bufg_I_O)        0.091    13.341 r  clk_IBUF_BUFG_inst/O
                        net (fo=390, routed)        1.435    14.776    CDIV/clk_IBUF_BUFG
  SLICE_X54Y24          FDRE                                       r  CDIV/count3_reg[20]/C
                        clock pessimism             0.274    15.050
                        clock uncertainty          -0.035    15.015
  SLICE_X54Y24          FDRE (Setup_fdre_C_R)      -0.524    14.491    CDIV/count3_reg[20]
----------------------------------------------------------------------
                        required time                        14.491
                        arrival time                        -10.885
----------------------------------------------------------------------
                        slack                                 3.606
```

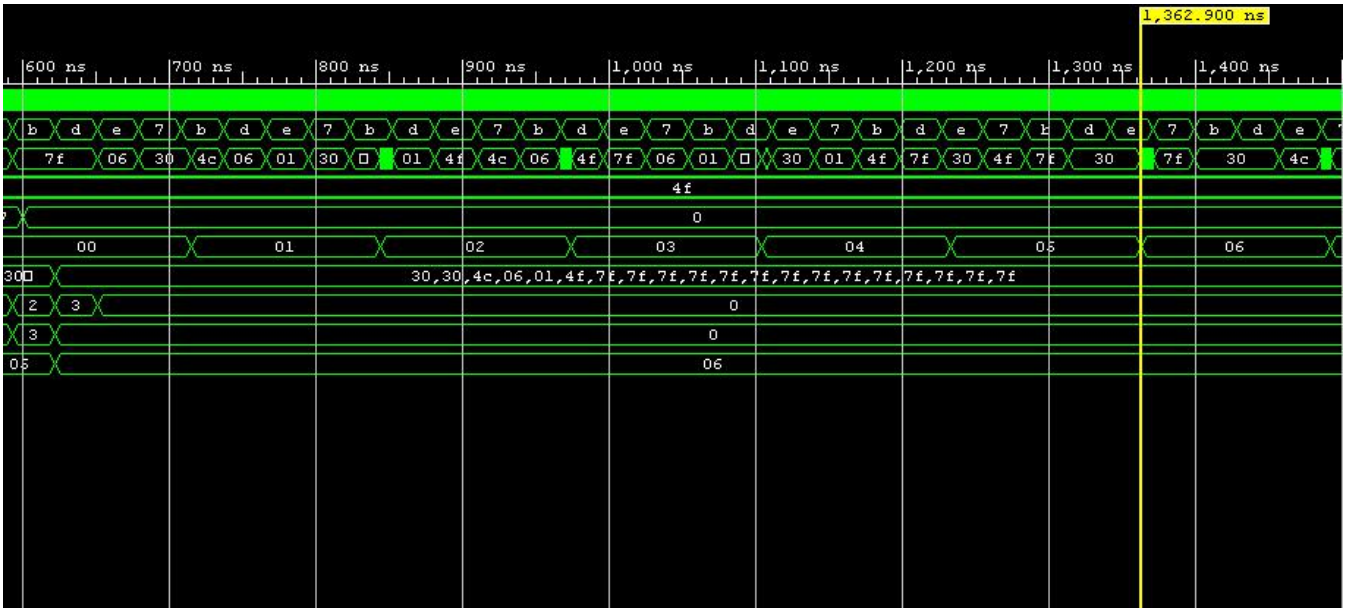
E) Schematic



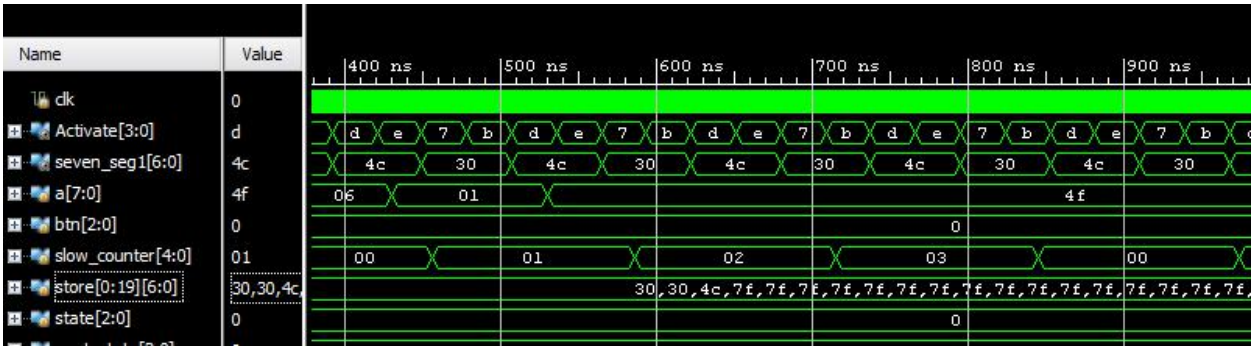
F) Waveform analysis



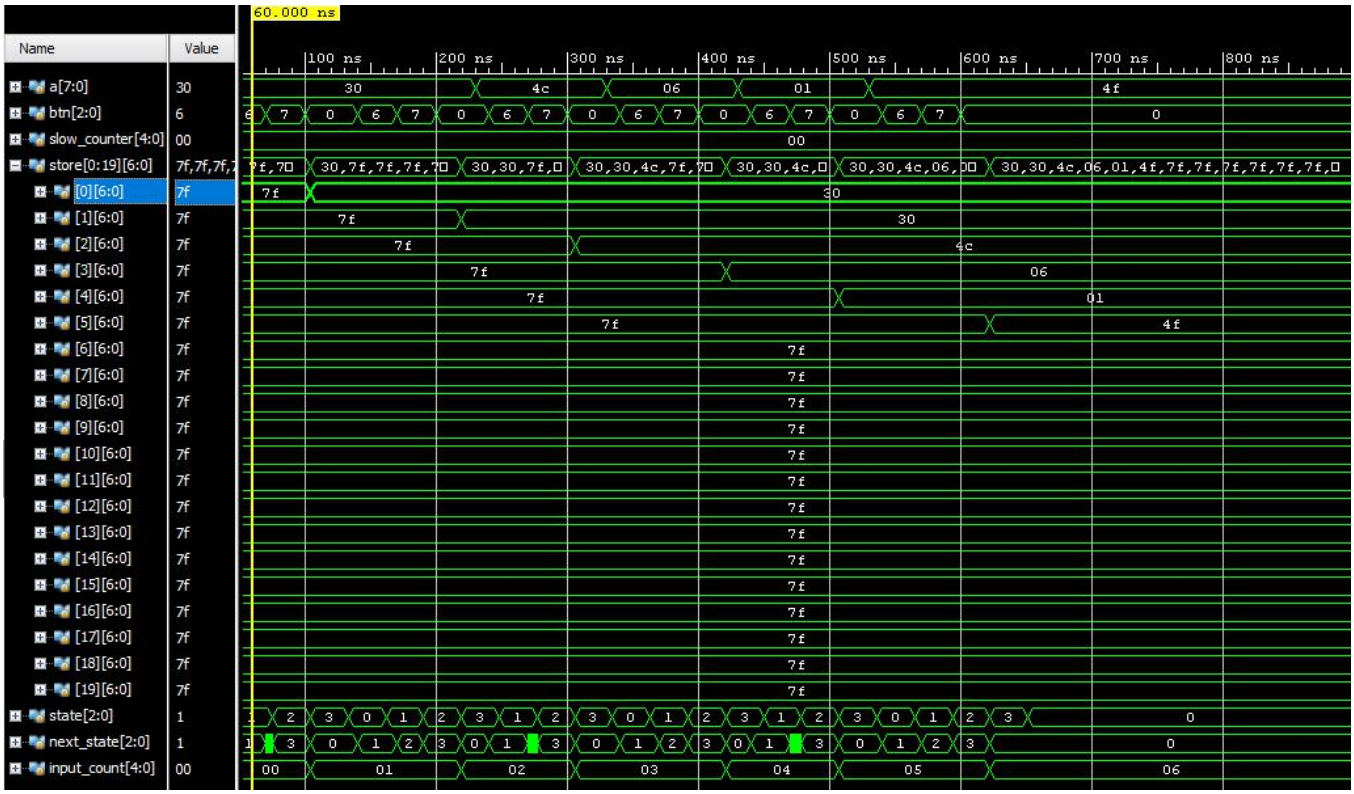
From the last waveform we can see that, after each S2 state, the value of a will be put in the “store” register. In my simulation, “EE4301” corresponding to hexadecimal number 30,30,4x,06,01,4f has been stored in the register in order. Next in the S0, we can see that for the enable number e,d,b,7 (the MSB to LSB), the moving text function is achieved.




Next it is the simulation for storing less than 3 character. In my simulation, I store the character “EE4”, which is corresponding to 30,30,4c. The display and moving text is shown as followed:




The storing process are shown in the next page.


60.000 ns
Name
Value
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
a[7:0]
30
btn[2:0]
6
slow_counter[4:0]
00
store[0:19][6:0]
7f,7f,7f,7
[0][6:0]
7f
[1][6:0]
7f
[2][6:0]
7f
[3][6:0]
7f
[4][6:0]
7f
[5][6:0]
7f
[6][6:0]
7f
[7][6:0]
7f
[8][6:0]
7f
[9][6:0]
7f
[10][6:0]
7f
[11][6:0]
7f
[12][6:0]
7f
[13][6:0]
7f
[14][6:0]
7f
[15][6:0]
7f
[16][6:0]
7f
[17][6:0]
7f
[18][6:0]
7f
[19][6:0]
7f
state[2:0]
1
next_state[2:0]
1
input_count[4:0]
00