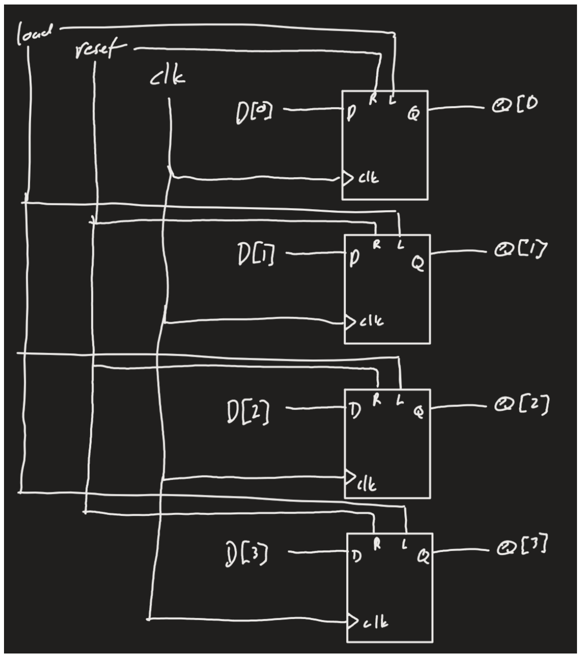
Mark Mitri and Ryan Forelli 9/20/2021
ECE 128 Lab 4

Part 1 Register:

Design:



Verilog Design Code:

```
`timescale 1ns / 1ps
//Register using behavioral modeling

module register(
  input [3:0] D, input clk, reset, load,
  output reg[3:0] Q
  );

  always @ (posedge clk) //Only run on positive clock edge
    begin
      if (reset) Q <= 4'b0000;
      else
        if (load) Q <= D;
        else Q <= Q;
    end
```

```
endmodule
```

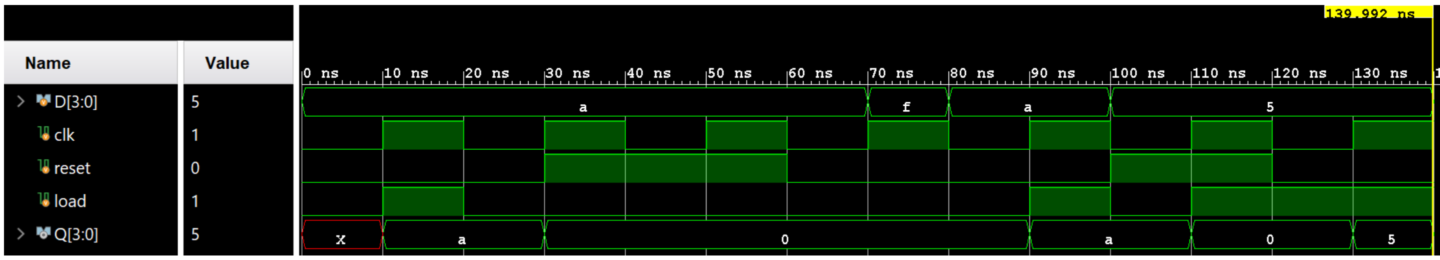
Testbench Code:

```
`timescale 1ns / 1ps

module register_tb;

  reg[3:0] D;
  reg clk, reset, load;
  wire[3:0] Q;

  register reg1(D, clk, reset, load, Q);

  initial begin

    clk=1'b0;D=4'b1010;reset=1'b0;load=1'b0;#10
    clk=1'b1;D=4'b1010;reset=1'b0;load=1'b1;#10
    if(Q == 4'b1010) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;D=4'b1010;reset=1'b0;load=1'b0;#10
    clk=1'b1;D=4'b1010;reset=1'b1;load=1'b0;#10
    if(Q == 4'b0000) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;D=4'b1010;reset=1'b1;load=1'b0;#10
    clk=1'b1;D=4'b1010;reset=1'b1;load=1'b0;#10
    clk=1'b0;D=4'b1010;reset=1'b0;load=1'b0;#10
    clk=1'b1;D=4'b1111;reset=1'b0;load=1'b0;#10
    if(Q != 4'b1111) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;D=4'b1010;reset=1'b0;load=1'b0;#10
    clk=1'b1;D=4'b1010;reset=1'b0;load=1'b1;#10
    clk=1'b0;D=4'b0101;reset=1'b1;load=1'b0;#10
    clk=1'b1;D=4'b0101;reset=1'b1;load=1'b1;#10
    if(Q == 4'b0000) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;D=4'b0101;reset=1'b0;load=1'b1;#10
    clk=1'b1;D=4'b0101;reset=1'b0;load=1'b1;#10
    $finish; //end the simulation
  end
endmodule
```

Testbench Tests:

- Loading test (setting register)
- Reset test
- Change in D without load set high
- Reset with load high

Simulation Waveform:

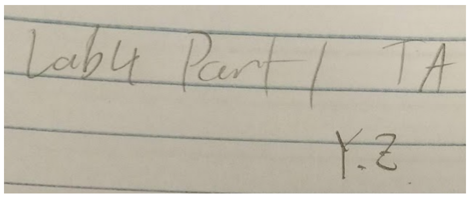


Name	Value
D[3:0]	5
clk	1
reset	0
load	1
Q[3:0]	5

Explanation:

The simulation is expected. On the first positive clock edge, 0xa is stored in the register as load is high on the positive clock edge. When reset goes high at t=30ns, the register is reset to 0 regardless of the input D and load. Q is set to 0xa again when load is set high on the positive clock edge at t=90ns. At t=110ns, Q is set to 0x0 as reset is enabled despite load being high. This behavior is expected.

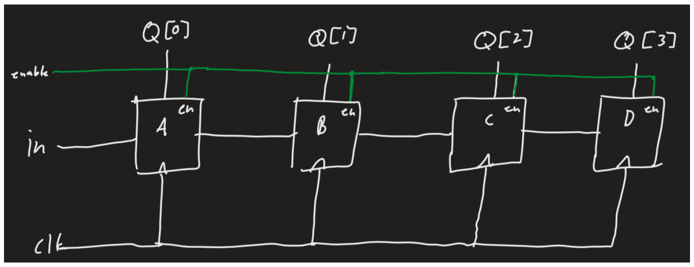
TA's Signature:


Lab4 Part 1 TA

Y.Z

Part 2 Shift Register:

Design:



Verilog Design Code:

```
`timescale 1ns / 1ps
//4-bit Serial in parallel out (SIPO) shift register using behavioral
modeling
module SIPO_shift_register(input clk, input in, shift_enable, output
reg[3:0] Q);
    always @ (posedge clk) begin
        if(shift_enable) begin
            Q[3] <= Q[2];
            Q[2] <= Q[1];
            Q[1] <= Q[0];
            Q[0] <= in;
        end
    end
endmodule
```

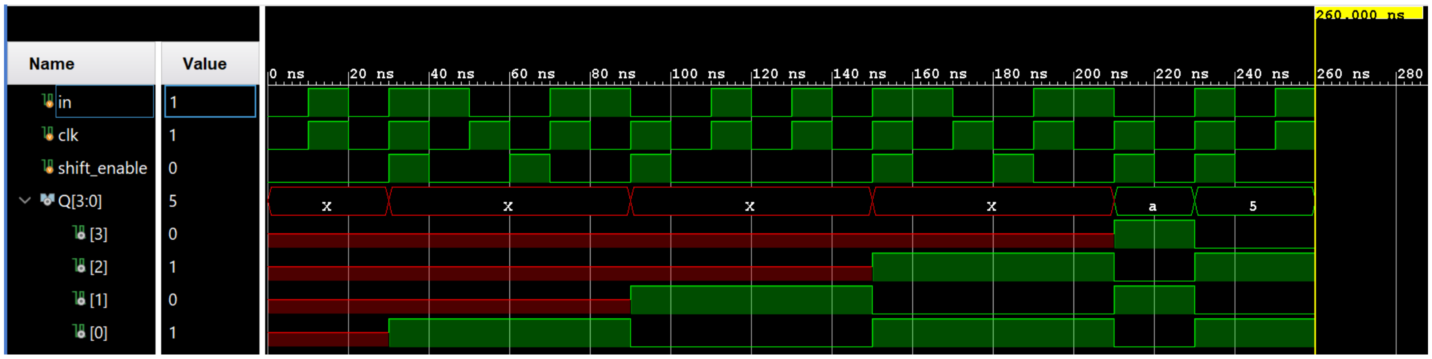
Testbench Code:

```
`timescale 1ns / 1ps

module SIPO_shift_register_tb;

    reg in, clk, shift_enable;
    wire[3:0] Q;

    SIPO_shift_register reg1(clk, in, shift_enable, Q);

    initial begin
        clk=1'b0;in=1'b0;shift_enable=1'b0;#10
        clk=1'b1;in=1'b1;shift_enable=1'b0;#10
        clk=1'b0;in=1'b0;shift_enable=1'b0;#10
        clk=1'b1;in=1'b1;shift_enable=1'b1;#10
        clk=1'b0;in=1'b1;shift_enable=1'b0;#10
        clk=1'b1;in=1'b0;shift_enable=1'b0;#10
        clk=1'b0;in=1'b0;shift_enable=1'b1;#10
        clk=1'b1;in=1'b1;shift_enable=1'b0;#10
        clk=1'b0;in=1'b1;shift_enable=1'b0;#10
        clk=1'b1;in=1'b0;shift_enable=1'b1;#10
        clk=1'b0;in=1'b0;shift_enable=1'b0;#10
        clk=1'b1;in=1'b1;shift_enable=1'b0;#10
        clk=1'b0;in=1'b0;shift_enable=1'b0;#10
        clk=1'b1;in=1'b1;shift_enable=1'b0;#10
        clk=1'b0;in=1'b0;shift_enable=1'b0;#10
        clk=1'b1;in=1'b1;shift_enable=1'b1;#10
        clk=1'b0;in=1'b1;shift_enable=1'b0;#10
        clk=1'b1;in=1'b0;shift_enable=1'b0;#10
        clk=1'b0;in=1'b0;shift_enable=1'b1;#10
        clk=1'b1;in=1'b1;shift_enable=1'b0;#10
        clk=1'b0;in=1'b1;shift_enable=1'b0;#10
        clk=1'b1;in=1'b0;shift_enable=1'b1;#10
        clk=1'b0;in=1'b0;shift_enable=1'b0;#10
        clk=1'b1;in=1'b1;shift_enable=1'b1;#10
        clk=1'b0;in=1'b0;shift_enable=1'b0;#10
        if(Q == 4'b1010) $monitor("PASS at ", $time);
        else $monitor("FAIL at ", $time);
        clk=1'b1;in=1'b1;shift_enable=1'b0;#10

      $finish; //end the simulation
    end
endmodule
```

Testbench Tests:

- Shift_enable set low on high clock edge prevents shift.
- Shift_enable set high on negative clock edge does not cause shift.
- Shift only occurs on positive clock edge and when shift_enable is high.
- Result is Q = 1’b1010 if all tests pass

Simulation Waveform:

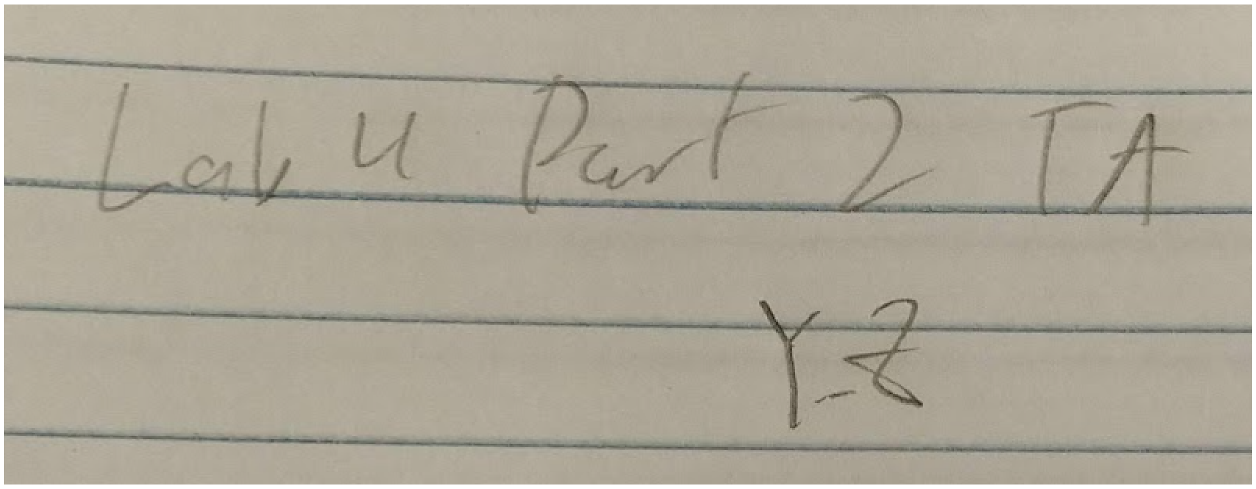


Name	Value
in	1
clk	1
shift_enable	0
Q[3:0]	5
[3]	0
[2]	1
[1]	0
[0]	1

<u>Explanation:</u>

The simulation is expected. A shift only occurs on a positive clock edge and when shift_enable is high. For example, at t=30ns, 1 is added to the shift register. On the next positive clock edge, no value is added as shift_enable is not high. Once the shift register is full, the next bit added to the LSB pushes the MSB out at t=230ns.

<u>TA's Signature:</u>

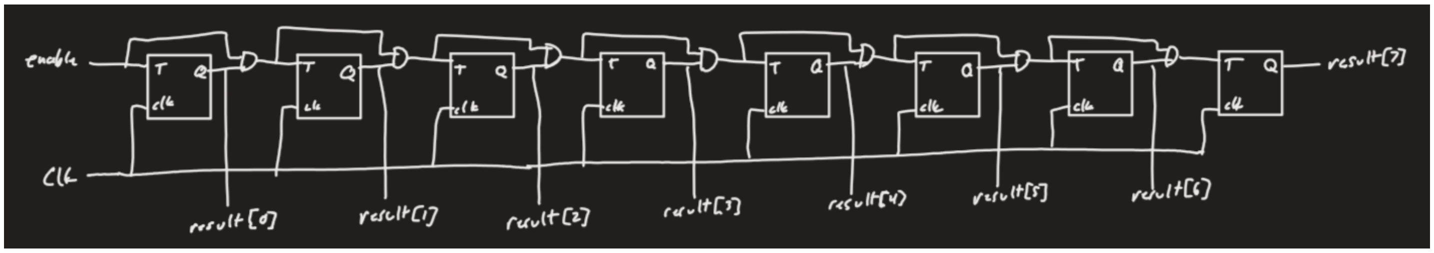


Lab 4 Part 2 TA

[illegible]

Part 3 TFF Counter:

<u>Design:</u>



Verilog Design Code:

```
`timescale 1ns / 1ps
//TFF counter using behavioral and dataflow modeling

module t_flip_flop(
  input enable, clk, clear,
  output reg Q
  );

  always @ (negedge clk) //Only run on negative clock edge
    begin
      if (clear) Q <= 1'b0; //If clear is pulled low, Q is 0
      else
        if (enable) Q <= ~Q;
        else Q <= Q;
    end
endmodule

module tff_counter(input enable, clk, clear, output[7:0] result);

    wire w1, w2, w3, w4, w5, w6, w7;

    t_flip_flop flip1(enable, clk, clear, result[0]);
    assign w1 = result[0] & enable;
    t_flip_flop flip2(w1, clk, clear, result[1]);
    assign w2 = result[1] & w1;
    t_flip_flop flip3(w2, clk, clear, result[2]);
    assign w3 = result[2] & w2;
    t_flip_flop flip4(w3, clk, clear, result[3]);
    assign w4 = result[3] & w3;
    t_flip_flop flip5(w4, clk, clear, result[4]);
    assign w5 = result[4] & w4;
    t_flip_flop flip6(w5, clk, clear, result[5]);
    assign w6 = result[5] & w5;
    t_flip_flop flip7(w6, clk, clear, result[6]);
    assign w7 = result[6] & w6;
    t_flip_flop flip8(w7, clk, clear, result[7]);

endmodule
```

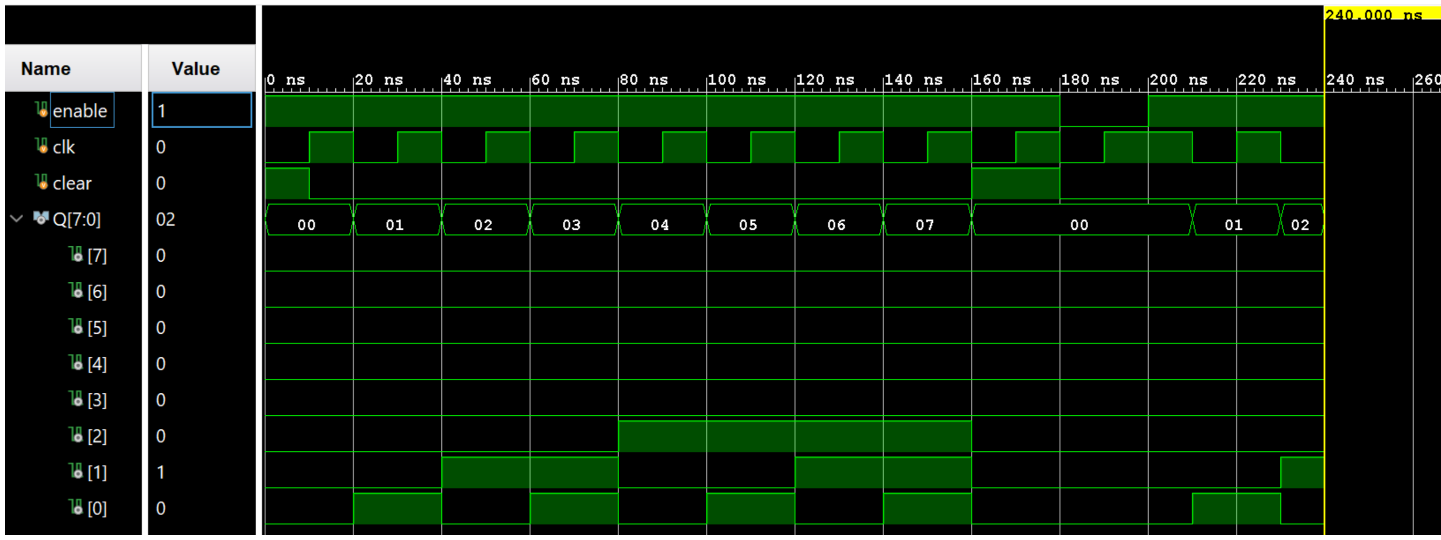
Testbench Code:

```
`timescale 1ns / 1ps

module t_flip_flop_tb;

  reg enable, clk, clear;
  wire[7:0] Q;

  tff_counter counter1(enable, clk, clear, Q);

  initial begin
    clk=1'b0;enable=1'b1;clear=1'b1;#10
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    if(Q == 8'b00000111) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;enable=1'b1;clear=1'b1;#10
    clk=1'b1;enable=1'b1;clear=1'b1;#10
    clk=1'b0;enable=1'b0;clear=1'b0;#10
    clk=1'b1;enable=1'b0;clear=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;#10
    if(Q == 8'b00000001) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b1;enable=1'b1;clear=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;#10

    $finish; //end the simulation
  end
endmodule
```

Testbench Tests:

- Counting up to 0x7 (on negative clock edge) with clear set low
- Clearing counter by setting clear high on a negative clock edge
- Counting up to 0x2 after clearing

<u>Simulation Waveform:</u>



<u>Explanation:</u>

The simulation is expected. The counter counts up on each negative clock edge. When clear is enabled (set low), the counter is cleared on the negative clock edge at t=160ns. When enable is brought low, the counter stops counting and does not increment at t=180ns. When enable is brought high again, the counter continues up to 0x3 at t=250ns.

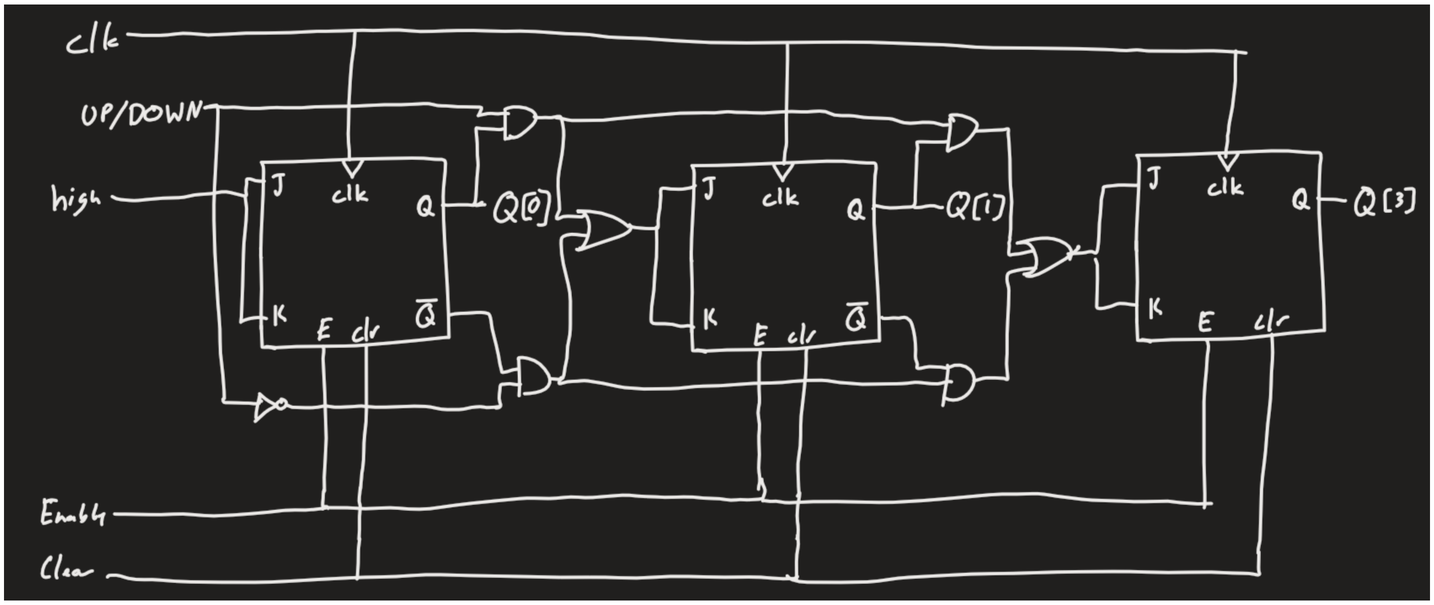
TA's Signature:

Lab 4 Part 3 TA

Y.Z

Part 4 Up-Down Counter

Design:



Verilog Design Code:

```
`timescale 1ns / 1ps

module SevenSeg(
  input[2:0] number,
  output reg[6:0] segments,
  output reg led,
  output reg[7:0] characters //Characters on the FPGA dev board. We
only use character AN0
  );

  always @ (*) //event-sensitivity list
    begin
      led = 1'b0;
      characters = 8'b11111110;
      case (number) //Cases for lighting up each number
        0 : segments = 7'b1000000;
        1 : segments = 7'b1111001;
        2 : segments = 7'b0100100;
        3 : segments = 7'b0110000;
        4 : segments = 7'b0011001;
        5 : segments = 7'b0010010;
        6 : segments = 7'b0000010;
        7 : segments = 7'b1111000;
        default:
          begin
           led = 1'b1;
           segments = 7'b1111111;
            characters = 8'b11111111;
          end
      endcase
    end
endmodule

//Up-down counter using behavioral modeling
module up_down_counter(
  input enable, clk, clear, up, down,
  output reg[2:0] Q,
  output[6:0] segments,
  output led,
  output[7:0] characters //Characters on the FPGA dev board. We only
use character AN0
  );

  SevenSeg seg1(Q, segments, led, characters);
  always @ (posedge clk) //Only run on positive clock edge
    begin
      if (clear) Q <= 3'b000;
      else
        if (enable) begin
            if(up && down)
                Q <= Q;
            else if(up && Q < 3'b111)
                Q <= Q + 3'b001;
            else if(down && Q > 3'b000)
                Q <= Q - 3'b001;
        end
        else Q <= Q;
    end
endmodule
```

<u>Testbench Code:</u>

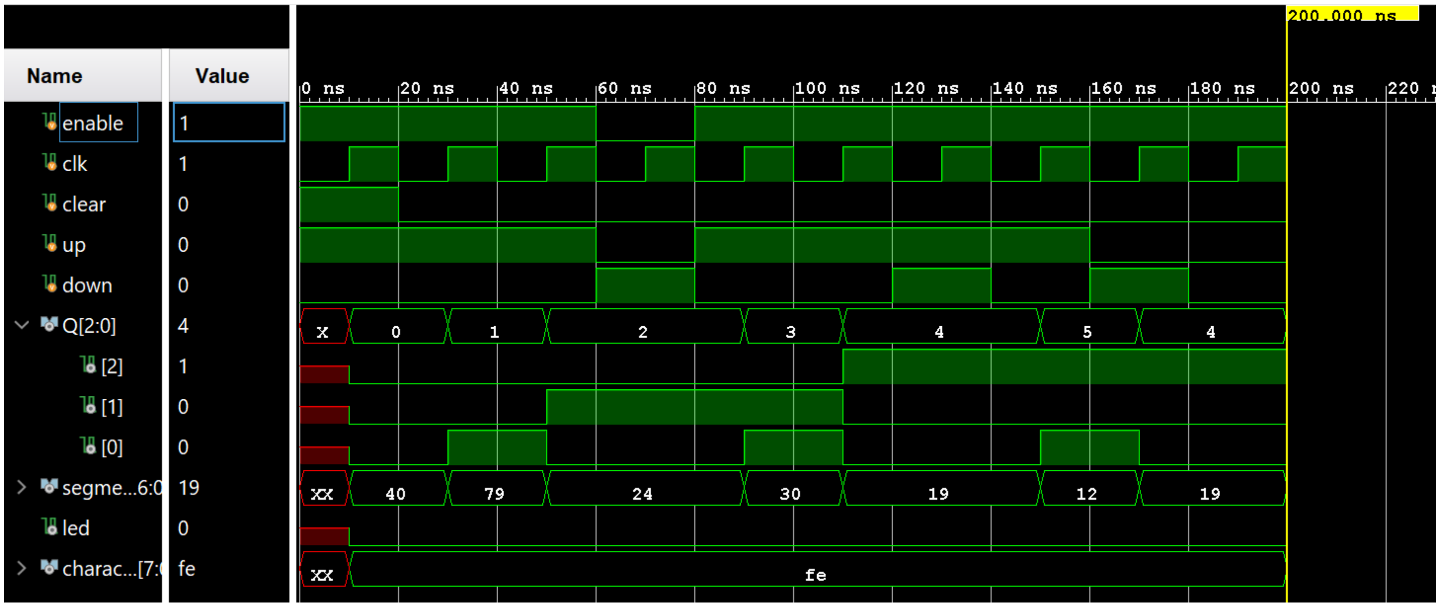
```
`timescale 1ns / 1ps

module up_down_counter_tb;

  reg enable, clk, clear, up, down;
  wire[2:0] Q;
  wire[6:0] segments;
  wire led;
  wire[7:0] characters; //Characters on the FPGA dev board. We only
use character AN0

  up_down_counter counter1(enable, clk, clear, up, down, Q, segments,
led, characters);

  initial begin
    clk=1'b0;enable=1'b1;clear=1'b1;up=1'b1;down=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b1;up=1'b1;down=1'b0;#10
    if(Q == 3'b000) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    if(Q == 3'b010) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;enable=1'b0;clear=1'b0;up=1'b0;down=1'b1;#10
    clk=1'b1;enable=1'b0;clear=1'b0;up=1'b0;down=1'b1;#10
    if(Q == 3'b010) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    clk=1'b0;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    if(Q == 3'b100) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;enable=1'b1;clear=1'b0;up=1'b1;down=1'b1;#10
    clk=1'b1;enable=1'b1;clear=1'b0;up=1'b1;down=1'b1;#10
    if(Q == 3'b100) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;up=1'b1;down=1'b0;#10
    if(Q == 3'b101) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;enable=1'b1;clear=1'b0;up=1'b0;down=1'b1;#10
    clk=1'b1;enable=1'b1;clear=1'b0;up=1'b0;down=1'b1;#10
    if(Q == 3'b100) $monitor("PASS at ", $time);
    else $monitor("FAIL at ", $time);
    clk=1'b0;enable=1'b1;clear=1'b0;up=1'b0;down=1'b0;#10
    clk=1'b1;enable=1'b1;clear=1'b0;up=1'b0;down=1'b0;#10

    $finish; //end the simulation
  end
endmodule
```

Testbench Tests:

- Tested clear
- Tested counting up
- Tested enable

- Tested up and down high at the same time resulting in no change in Q
- Tested counting down

Simulation Waveform:



Explanation:

The simulation is expected. Clear is initially enabled to set Q to 3'b000. After counting for two clock cycles, Q is 3'b010. At t=60ns down is set high but Q is not affected as enable is low. After counting to four, both up and down are set high at t=120ns and Q is not affected as expected. Q is then decremented twice to 3'b100 after down is set high. When neither up or down is high, Q is not affected as expected. Bits corresponding to the segments of the seven segment display are also continuously updated to display the current value of Q.

TA's Signature:

Lab 4 Part 4 TA

[illegible]

References:

- Lecture Slides
- ElectronicsTutorials