Designing a Synchronous Decimal Counter using Verilog

(Note: All inputs are asynchronous in nature)

Verilog code

```
module upordown(
   input wire Clk,
   input wire reset,
   input wire UpOrDown,
   input wire load,
   input wire stop,
   input wire start,
   output  [6:0] Count,output buz

);
reg  buz;
reg [6:0] Count = 7'd0;
reg [31:0] clkdiv ;
always@(posedge Clk)
clkdiv=clkdiv + 1;

always @(posedge clkdiv[25])
 begin
   if (stop) begin
      Count <= Count;buz<=1'd0;
   end

         else if (UpOrDown && start) begin // Up mode selected

         if (Count == 7'd99) begin
         Count <= 7'd0;buz<=1'd1;
      end else if (load) begin
         Count <= 7'd90;buz<=1'd0;
      end else if (reset) begin
         Count <= 7'd0;buz<=1'd0;
      end else begin
         Count <= Count + 1;buz<=1'd0;
      end
                  end else if (!UpOrDown && start) begin // Down mode selected
      if (Count == 7'd0) begin
         Count <= 7'd99;buz<=1'd1;
      end else if (load) begin
         Count <= 7'd10;buz<=1'd0;
      end else if (reset) begin
```

```
            Count <= 7'd99;buz<=1'd0;
        end else begin
            Count <= Count - 1;buz<=1'd0;
                                                end
    end
end




endmodule
```

This code module is implementing a simple up/down counter with some additional features.

The inputs to the module are:

- Clk: clock signal
- reset: asynchronous reset signal
- UpOrDown: control signal to select the counting direction (up or down)
- load: signal to load a preset value into the counter
- stop: signal to stop the counter
- start: signal to start the counter

The outputs of the module are:

- Count: a 7-bit output representing the current count value
- buz: a 1-bit output indicating whether or not the buzzer should be turned on

The counter is implemented using a reg type variable Count, which is initialized to 0. The module also uses a reg variable buz to control the buzzer output.

The module uses two always blocks, one triggered by the Clk signal and another by the clkdiv signal. The first ‘always’ block increments or decrements the Count variable depending on the UpOrDown input and the start signal. If the count reaches the maximum value (99), it is reset to 0 and the buzzer output is set to 1. Similarly, if the count reaches the minimum value (0), it is reset to 99 and the buzzer output is set to 1. The load and reset signals are also used to set the Count variable to a preset value. During the up condition load and reset value is 90 and 0 respectively. During the down condition load and reset value is 10 and 99 respectively.

The second ‘always’ block generates a clkdiv signal by incrementing a 32-bit register on every positive edge of the Clk signal(50MHz). This signal is then used to trigger the first ‘always’ block on every 26th positive edge of the Clk signal, effectively slowing down the counter by a factor of 26(1 second).

Test bench

```
module yu;

        // Inputs
        reg Clk;
        reg reset;
        reg UpOrDown;
        reg load;
        reg stop;
        reg start;

        // Outputs
        wire [6:0] Count;
        wire buz;

        // Instantiate the Unit Under Test (UUT)
        upordown uut (
                .Clk(Clk),
                .reset(reset),
                .UpOrDown(UpOrDown),
                .load(load),
                .stop(stop),
                .start(start),
                .Count(Count),
                .buz(buz)
        );
initial Clk = 0;
   always #2 Clk = ~Clk;
        initial begin
                start=1; reset = 0;
      UpOrDown = 0;
      #50;
                 start=0;stop=1;
      #50;
                 start=1;stop=0;
                 #50;
reset=1;
#50
reset=0;
#50;
load=1;
#50;
load=0;
```

```
   #100;
           UpOrDown = 1;
       #50;
       reset = 1;
#50;
reset=0;
#50
load=1;
#50
load=0;

#50;
                     start=0;stop=1;
       #50;
                     start=1;stop=0;


          end
initial begin
$monitor($time," %d %d %d %d %d %d %d",start,UpOrDown,reset,load,stop,buz,Count);
end

endmodule
```

This code module is a testbench for the upordown module.The testbench instantiates the upordown module as uut and provides the necessary inputs to test its functionality.

The testbench initializes the inputs and waits for some time before changing them to test different scenarios. It uses an ‘initial’ block to initialize the inputs and toggle the clock signal every 2 time units. It then sets the start signal to 1 and waits for 50 time units before setting the stop signal to 1 and waiting for another 50 time units. It then sets the start signal back to 1 and waits for 50 time units before setting the stop signal back to 0 and waiting for another 50 time units.

The testbench tests various scenarios by changing the inputs and waiting for some time before changing them again. It uses the $monitor system task to print the values of the inputs and outputs at every time step.

Overall, this testbench tests the functionality of the upordown module by providing different inputs and monitoring the outputs.

UCF File

```
//# Slide Switch
NET "stop" LOC = p22;
NET "start" LOC = p21;
NET "reset" LOC = p17;
NET "load" LOC = p16;

#led
NET "Count[0]" LOC = p33; #LSB
NET "Count[1]" LOC = p32;
NET "Count[2]" LOC = p30;
NET "Count[3]" LOC = p29;
NET "Count[4]" LOC = p27;
NET "Count[5]" LOC = p26;
NET "Count[6]" LOC = p24;

#Slide Switches
NET "UpOrDown" LOC = p22; #LSB

NET "stop" LOC = P21;
NET "start" LOC = P17;
NET "reset" LOC = P16;
NET "load" LOC = P15;

#clk
NET "Clk" LOC = p84;


#buzzer
NET "buz" LOC = p85;
```

This code is a pin assignment file for a digital circuit design that includes a slide switch, LED, buzzer, and clock signal. It assigns specific pins on a physical device to various input and output signals in the digital circuit.

The first section of the code assigns the slide switch signals "stop", "start", "reset", and "load" to pins p22, p21, p17, and p16 respectively. These signals are used as inputs to the digital circuit.

The next section of the code assigns the LED signals "Count[0]" to "Count[6]" to pins p33, p32, p30, p29, p27, p26, and p24 respectively. These signals are used as outputs from the digital circuit.

The next section of the code assigns the slide switch signal "UpOrDown" to pin p22.
The following section of the code assigns the clock signal "Clk" to pin p84.

Finally, the code assigns the buzzer output signal "buz" to pin p85.