Solutions PDF Generated from: solutions-openai-generated/lab-practical-exam/lab-practical-exam-solutions-set-01.json

Question A

Implement Verilog code for a counter that can count up or down or pause, based on button inputs from the users. The counter can also be enabled or disabled. The current value of the counter will be displayed on four 7- segment displays at the same time. One switch will determine if the counter is enabled or disabled. If the counter is disabled by the switch, the counter will be reset to zero. If the counter is enabled by the switch, 3 buttons, pressed individually, will determine if the counter counts up, counts down, or pauses. 7-segnment display code is provided, also code for a clock divider is provided. All you have to do is to write Verilog code to monitor input buttons and a switch, manage the counter.

```
‘module count_down_up (
  // Seven segment output
  output [6:0] seg_o,

  // Clock signal
  input clk,

  // Reset signal
  input rst,

  // enable, up, down and pause signal
  input [3:0] enable_signal
);

  reg count_enable;

  // Output 4 Digit counter
  reg [3:0] count_o;

  // Clock divider
  reg [14:0] clock_div;

  // Internal FSM signal
  reg [2:0] state, next_state;

  // Initialize
  initial begin
    // Set enable signal to 0
    count_enable = 0;

    // Initialize counter output
    count_o = 4’b0000;

    // Set clock division register
```

```
  clock_div = 15'b000000000000000;

  // Initialize FSM state
  state = 3'b000;
  next_state = 3'b000;
end

// Clock divider
always @ (posedge clk)
  if (rst)
    clock_div <= 15'b000000000000000;
  else
    clock_div <= clock_div + 1;

// Debounce logic
always @ (clock_div)
  if (clock_div == 15'b111100000000000 )
    begin
      // Set enable signal based on input
      count_enable <= enable_signal[0];

      // Check if the counter is enabled
      if (enable_signal[0] == 1)
        begin
          // Determine which action is taken, e.g. down, up, pause
          case (enable_signal[1:3])
            3'b001 : next_state = 3'b001; // Count up
            3'b010 : next_state = 3'b010; // Count down
            3'b011 : next_state = 3'b011; // Pause
          endcase
        end
      // If the counter is disabled, reset it
      else
        next_state = 3'b000;
    end

// FSM logic to update the count
always @ (posedge clk or posedge rst)
  if (rst)
    // Reset the state
    state <= 3'b000;
  else
    // Update state based on next_state
    state <= next_state;

// Counter
always @ (posedge clk or posedge rst)
  // Reset counter if counter is disabled
  if (rst | !count_enable)
    count_o <= 4'b0000;
  // Increment counter
  else if (state == 3'b001)
    begin
      // Limit the maximum count
      if (count_o == 4'b1001)
        count_o <= 4'b0000;
      else
        count_o <= count_o + 4'b0001;
    end
```

```
  // Decrement counter
  else if (state == 3'b010)
    begin
      // Limit the minimum count
      if (count_o == 4'b0000)
        count_o <= 4'b1001;
      else
        count_o <= count_o - 4'b0001;
    end

// 7-segment Logic
// Provide the driving signals for 7-segment display
assign seg_o = 7_seg(count_o);

// Definition of 7-segment display
function [6:0] 7_seg (input [3:0] count_i);
  case (count_i)
    4'b0000: begin 7_seg = 7'b1000000; end
    4'b0001: begin 7_seg = 7'b1111001; end
    4'b0010: begin 7_seg = 7'b0100100; end
    4'b0011: begin 7_seg = 7'b0110000; end
    4'b0100: begin 7_seg = 7'b0011001; end
    4'b0101: begin 7_seg = 7'b0010010; end
    4'b0110: begin 7_seg = 7'b0000010; end
    4'b0111: begin 7_seg = 7'b
```

Execution Time

0:00:37.320648

OpenAI Parameters

Model: text-davinci-003, Max. Tokens: 1024, Temperature: 1, N: 1