Digital Logic Laboratory

Report On LAB #5 - Timer

Ana Maghradze

18th December 2021

Contents

1 Description of The Assignment

For this assignment we have to implement timer and use Seven Segment Display to show the timer values on that. Pressing on KEY[1] should should pause/continue timer, pressing on KEY[0] should reset timer. In order for Seven Segment Display segments not to be burnt, we need to use level converters and don't directly connect GPIO pins to the display pins.

2 Description of The Solution

I have three modules in my code. In de0_nano_soc_baseline module, I handle KEY inputs (pause/continue/reset) and send data on GPIO_1 pins: the first 7 GPIO[7:0]) pins are used for display segments, the last 4 pins (GPIO[35:32]) are used for determining digit position.

In digitPositionDecoder module, I have counter for 4 digit positions and use clock enable because the digits on seven segment display are displayed one at a time so I need clock fast enough to display all of them almost at the same time to see the counter value correctly. Inside the clock, I take counter value and determine, firstly, if it is a digit of units, digit of hundreds or digit of thousands, then on what position the digit goes.

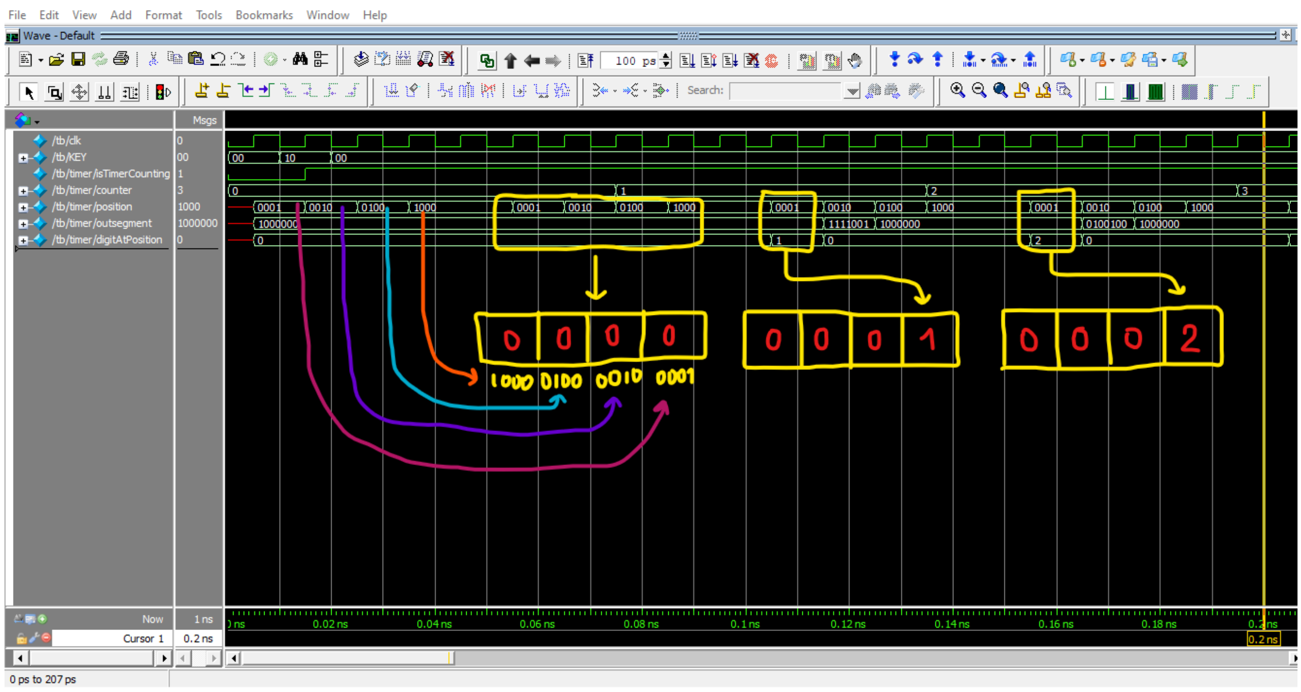
In decoder module, I take digit from digitPositionDecoder and decode - send 0s (ON) and 1s (OFF) on the first 7 pins of GPIO_7 in order for corresponding segments to light up on seven segment display.

I fixed it for seven segment display that on the reset not to be turned off and instead, to display zeros.

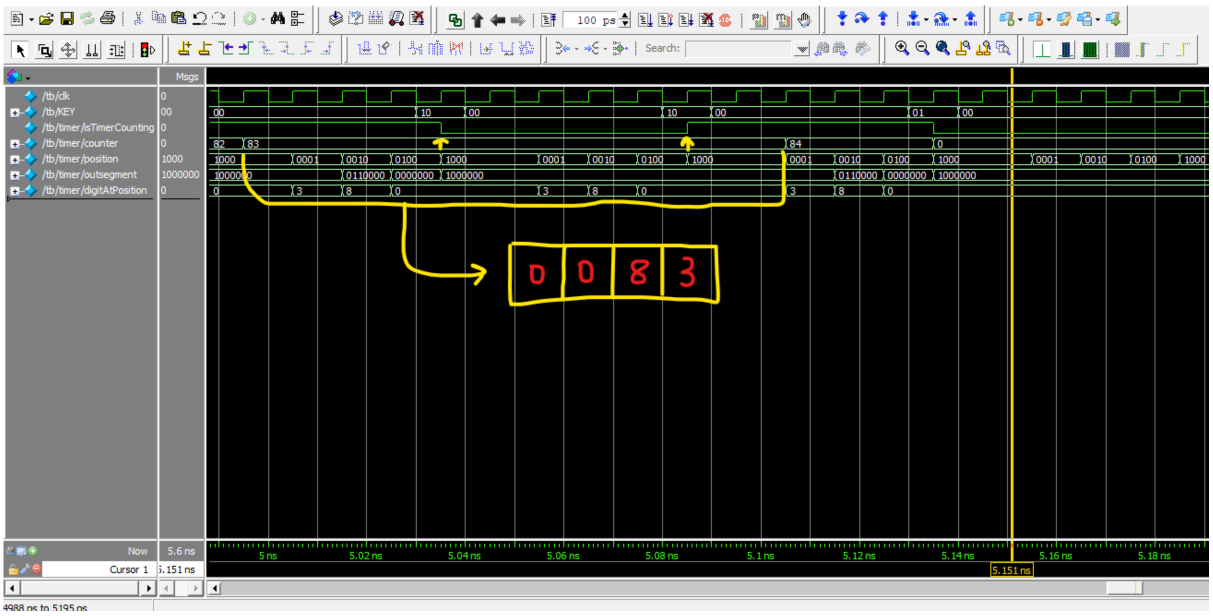
I worked on this lab with Giorgi Zirakashili and Khatia Nikuradze.

3 ModelSim Simulation

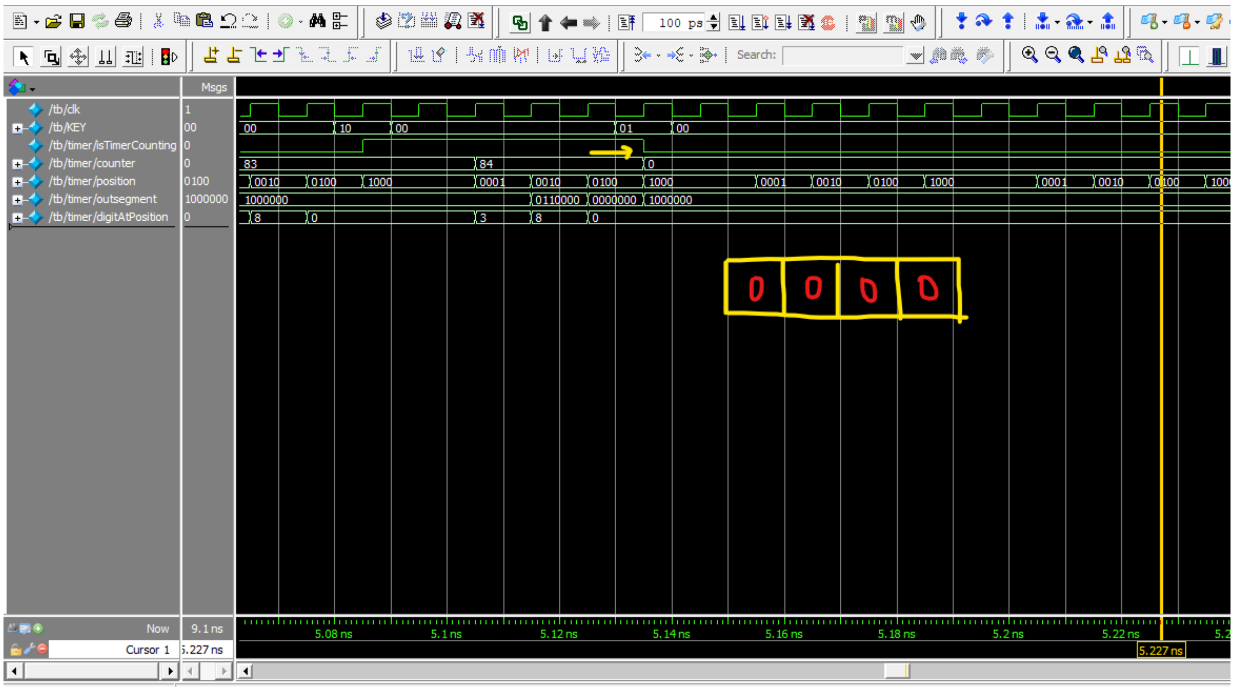
3.1 Start Timer



3.2 Pause/Continue Timer



3.3 Reset Timer



4 Appendix

4.1 de0_nano_soc_baseline

```
module de0_nano_soc_baseline(
  input CLOCK_50,
  input [1:0] KEY,
  inout [35:0] GPIO_1 // comment this for modelsim
);

reg [1:0] prevKey = 2'b11;

localparam CLK_HZ = 50000000;
integer clock_counter = 0;
reg isTimerCounting = 0;

integer counter = 0;

wire [3:0] digitAtPosition;
wire [6:0] outsegment;
wire [3:0] position;

digitPositionDecoder d2(CLOCK_50, counter, digitAtPosition, position);
decoder d1(CLOCK_50, isTimerCounting, digitAtPosition, outsegment);

assign GPIO_1[6:0] = outsegment[6:0];  // comment this for modelsim
assign GPIO_1[35:32] = position;      // comment this for modelsim

always @(posedge CLOCK_50)
begin
  if((clock_counter == CLK_HZ) && isTimerCounting)
  begin
    counter <= counter + 1;
    clock_counter <= 0;
  end
  else if(isTimerCounting) clock_counter <= clock_counter + 1;

  if(KEY[0] && prevKey[0] == 0)
  begin
    isTimerCounting <= 0;
    counter <= 0;
  end
  if(KEY[1] && ~prevKey[1]) isTimerCounting <= ~isTimerCounting;
  prevKey <= KEY;
end

endmodule
```

4.2 decoder

```
module decoder(
  input clk,
  input isCounting,
  input [3:0] digit,
  output reg [6:0] outsegment
);

always @(posedge clk)
begin
  if(isCounting)
  begin
    case(digit)
      0: outsegment <= 7'b1000000;
      1: outsegment <= 7'b1111001;
      2: outsegment <= 7'b0100100;
      3: outsegment <= 7'b0110000;
      4: outsegment <= 7'b0011001;
      5: outsegment <= 7'b0010010;
      6: outsegment <= 7'b0000010;
      7: outsegment <= 7'b1111000;
      8: outsegment <= 7'b0000000;
      9: outsegment <= 7'b0010000;
    endcase
  end
  else outsegment <= 7'b1000000;

end

endmodule
```

4.3 digitPositionDecoder

```
module digitPositionDecoder(
  input clk,
  input [13:0] in,
  output reg [3:0] digitAtPosition,
  output reg [3:0] position
);

localparam CLK_HZ = 50000000;
integer clk_counter = 0;

integer count = 0;

always @(posedge clk)
begin
  if(clk_counter == CLK_HZ / 1000)
  begin
    case(count)
      0:
      begin
        digitAtPosition <= in % 10;
        position <= 4'b0001;
      end
      1:
      begin
        digitAtPosition <= ((in % 100) - (in % 10)) / 10;
        position <= 4'b0010;
      end
      2:
      begin
        digitAtPosition <= ((in % 1000) - (in % 100)) / 100;
        position <= 4'b0100;
      end
      3:
      begin
        digitAtPosition <= ((in % 10000) - (in % 1000)) / 1000;
        position <= 4'b1000;
      end
    endcase

    count <= count == 4 ? 0 : count + 1;
    clk_counter <= 0;
  end
  else clk_counter <= clk_counter + 1;
end

endmodule
```

4.4 testbench

```
module tb;

reg clk = 0;
reg [1:0] KEY = 0;

always #5 clk <= ~clk;

de0_nano_soc_baseline timer(clk, KEY);

initial
begin
  // start counting
  #10 KEY[1] = 1;
  #10 KEY[1] = 0;

  #5000

  // pause counting
  #10 KEY[1] = 1;
  #10 KEY[1] = 0;

  #30

  // resume counting
  #10 KEY[1] = 1;
  #10 KEY[1] = 0;

  #30

  //  reset
  #10 KEY[0] = 1;
  #10 KEY[0] = 0;

end

endmodule
```