Digital Logic Laboratory

Lab #1 – Report

Ana Maghradze

16th October 2021

Contents

1 Description of The Assignment

In this assignment, we implement press multiplier on FPGA. We should have 4 different stages, use Key[0] to go from one stage to another and Key[1] to change the values of press_counter_A and press_counter_B – the numbers that we need to multiply.

When Key[0] is pressed, program goes to the first stage and LED[0] is ON. We should be able to increment press_counter_A by pressing key[1] at first stage. If Key[0] is pressed again, program goes to the second stage, LED[0] and LED[1] are ON and we should be able to increment press_counter_B by pressing key[1]. If Key[0] is pressed again, program goes to the third stage and the result of the multiplication of press_counter_A and press_counter_B should appear on LEDs. If Key[0] is pressed again, the program returns to the initial stage.

2 Description of The Solution

In the code, together with the inputs - CLOCK_50 and KEY, I had to use 4 registers: stage, press_counter_A, press_counter_B, and prevKey.

2-bit register – prevKey holds the previous value of the KEY. I need this register to detect the change from previous value to the new value of KEY and, by this way, avoid the incorrect output caused by pressing on the button as its value remains the same for greater time than we need it to be. So, at the end of the always block, I write the KEY value into the prevKey.

On the positive edge of the clock, I check the values of KEY[0] and prevKey[0] together and KEY[1] and prevKey[1] together.

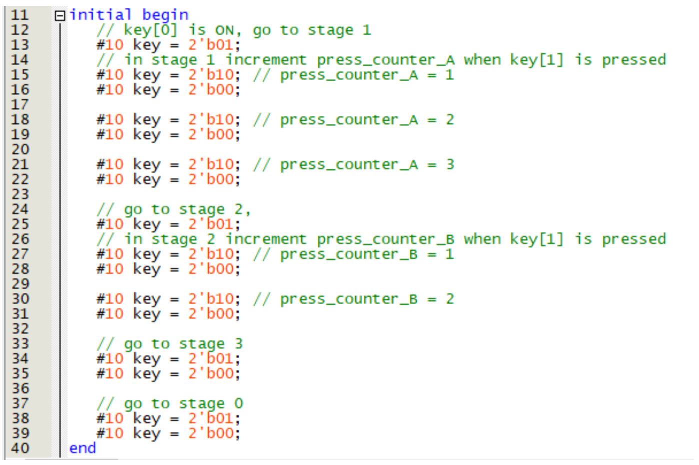
If KEY[0] is pressed, stage is changed. From the 3rd state everything goes to the 0th stage again.

We need KEY[1] only in 2nd and 3rd stages – if it's pressed, on the first stage press_counter_A is incremented, on the second stage press_counter_B is incremented.

If stage changed from 3 to 0, press_counter_A and press_counter_B values become 0s.

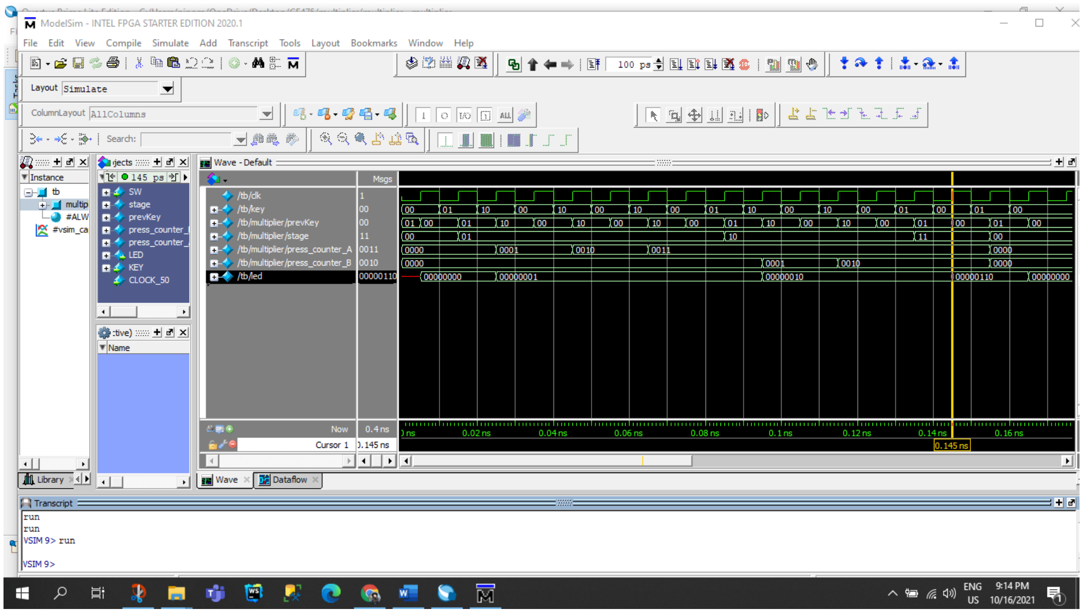
LED value on 0th, 1st or 2nd stage is the value of the stage itself, on the 3rd stage its value becomes the multiplication result.

3 ModelSim Simulation



```verilog
initial begin
    // key[0] is ON, go to stage 1
    #10 key = 2'b01;
    // in stage 1 increment press_counter_A when key[1] is pressed
    #10 key = 2'b10; // press_counter_A = 1
    #10 key = 2'b00;

    #10 key = 2'b10; // press_counter_A = 2
    #10 key = 2'b00;

    #10 key = 2'b10; // press_counter_A = 3
    #10 key = 2'b00;

    // go to stage 2,
    #10 key = 2'b01;
    // in stage 2 increment press_counter_B when key[1] is pressed
    #10 key = 2'b10; // press_counter_B = 1
    #10 key = 2'b00;

    #10 key = 2'b10; // press_counter_B = 2
    #10 key = 2'b00;

    // go to stage 3
    #10 key = 2'b01;
    #10 key = 2'b00;

    // go to stage 0
    #10 key = 2'b01;
    #10 key = 2'b00;
end
```

Figure 1: testbench



```
ModelSim - INTEL FPGA STARTER EDITION 2020.1

Wave - Default                          Msgs
/tb/clk                                 1
/tb/key                                 00
/tb/multiplier/prevKey                  00
/tb/multiplier/stage                    11
/tb/multiplier/press_counter_A          0011
/tb/multiplier/press_counter_B          0010
/tb/led                                 00000110

Now       0.4 ns
Cursor 1  0.145 ns

Transcript
run
run
VSIM 9> run

VSIM 9>
```

4 Appendix

```
module de0_nano_soc_baseline(
  input CLOCK_50,
  input [1:0] KEY,
  input [3:0] SW,
  output reg [7:0] LED
);
  reg [1:0] stage = 0;
  reg [1:0] prevKey = 1;
   reg [3:0] press_counter_A = 0;
   reg [3:0] press_counter_B = 0;

  always @(posedge CLOCK_50)
  begin
    if(KEY[0] == 1 && prevKey[0] == 0)
    begin
      press_counter_A <= stage == 3 ? 0 : press_counter_A;
      press_counter_B <= stage == 3 ? 0 : press_counter_B;

      stage <= stage == 3 ? 0 : stage + 1;
    end

    if(KEY[1] == 1 && prevKey[1] == 0)
    begin
      press_counter_A <= stage == 1 ? press_counter_A + 1 : press_counter_A;
      press_counter_B <= stage == 2 ? press_counter_B + 1 : press_counter_B;
    end

    LED[7:0] <= stage == 3 ? press_counter_A * press_counter_B : stage;
    prevKey[1:0] <= KEY[1:0];

  end
endmodule
```

TESTBENCH

```
module tb;
reg clk = 0;
reg [1:0] key = 0;
reg [3:0] sw = 0;
wire [7:0] led;

de0_nano_soc_baseline multiplier(clk, key, sw, led);

always #5 clk <= ~clk;

initial begin
  // key[0] is ON, go to stage 1
  #10 key = 2'b01;
  // in stage 1 increment press_counter_A when key[1] is pressed
  #10 key = 2'b10; // press_counter_A = 1
  #10 key = 2'b00;

  #10 key = 2'b10; // press_counter_A = 2
  #10 key = 2'b00;

  #10 key = 2'b10; // press_counter_A = 3
  #10 key = 2'b00;

  // go to stage 2,
  #10 key = 2'b01;
  // in stage 2 increment press_counter_B when key[1] is pressed
  #10 key = 2'b10; // press_counter_B = 1
  #10 key = 2'b00;

  #10 key = 2'b10; // press_counter_B = 2
  #10 key = 2'b00;

  // go to stage 3
  #10 key = 2'b01;
  #10 key = 2'b00;

  // go to stage 0
  #10 key = 2'b01;
  #10 key = 2'b00;
end

endmodule
```