Remi de Ferrieres
X664611
CS 120A Section 021

Lab 6 – Timer Design – Laser surgery system

Yann Abou Jaoude
X663581

OVERVIEW:

In the first part of this lab, we worked on the implementation of a flight attendant call system using Verilog. We use the FPGA board to visualize the results. There were two buttons, on for lighting up a led, the other on to turn it off, all this using a clock source (internal).

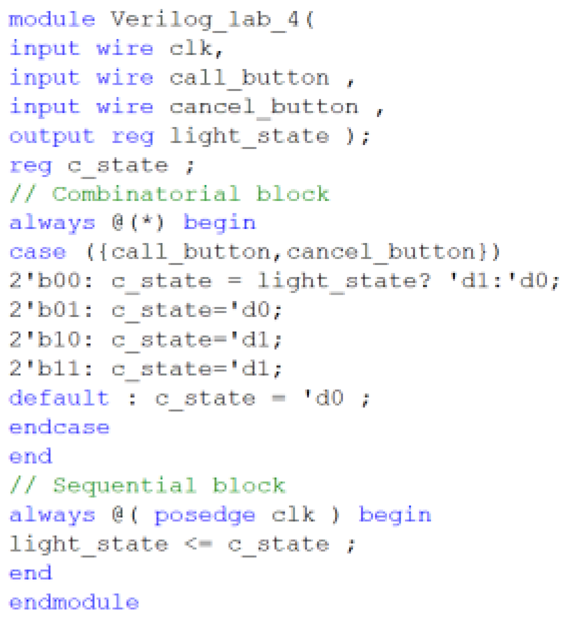
In the second part we worked on the implementation of an FSM developed on Verilog (rising edge detector). Using the FPGA board, we wanted to visualize a clock rising edge and falling edge (led on for rising and off for falling edge). The clock frequency was too important at first (25MHz) so we had to include a bit of code to reduce this frequency to be able to notice the led with our eyes.

In the third part of the lab we worked on LED display time multiplexing. Implementing in Verilog a circuit.

ANALYSIS:

RECORDS:

- Source Code Part1:

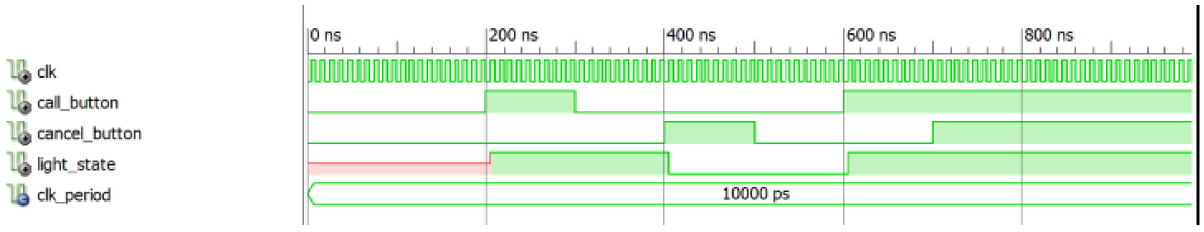


```
module Verilog_lab_4(
input wire clk,
input wire call_button ,
input wire cancel_button ,
output reg light_state );
reg c_state ;
// Combinatorial block
always @(*) begin
case ({call_button,cancel_button})
2'b00: c_state = light_state? 'd1:'d0;
2'b01: c_state='d0;
2'b10: c_state='d1;
2'b11: c_state='d1;
default : c_state = 'd0 ;
endcase
end
// Sequential block
always @( posedge clk ) begin
light_state <= c_state ;
end
endmodule
```

- UCF file Part1:

```
// Inputs
NET "clk" LOC = "B8";
NET "call_button" LOC = "A7";
NET "cancel_button" LOC = "M4";
// Outputs
NET "light_state" LOC = "M11";
```

- Simulation/Wave Form:



- Source code Part2:

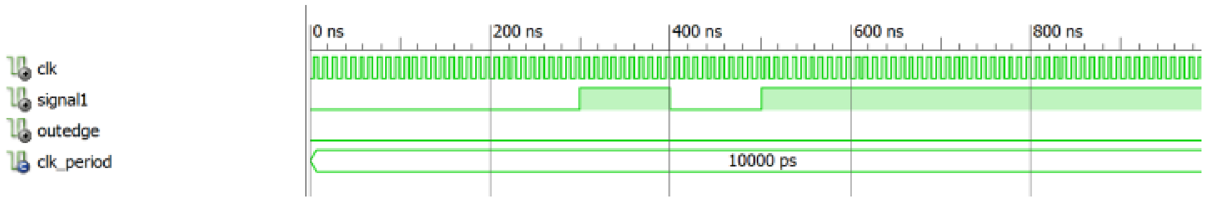
```
`timescale 1ns / 1ps
module Verilog_lab_4_2(
 input  wire clk,
 input  wire signal1,
 output reg outedge
    );

wire slow_clk ;

reg [1:0] c_state ;
reg [1:0] r_state ;

localparam ZERO = 'd0;
localparam CHANGE = 'd1;
localparam ONE = 'd2;
// http://www-inst.eecs.berkeley.edu/~cs150/sp12/agenda/lec/lec17-FSM.pdf
clkdiv c1(clk, slow_clk );

// Comb. logic.

always @(*) begin

  case (r_state)

  ZERO   : begin
          c_state =  signal1 ? CHANGE :  ZERO ;
          outedge = 'd0 ;
          end

  CHANGE  : begin
          c_state =  signal1 ? ONE :  ZERO ;
          outedge = 'd1 ;
          end

  ONE  : begin
          c_state =  signal1 ? ONE :  ZERO ;
          outedge = 'd0 ;
       end

   default : begin
            c_state = ZERO ;
            outedge = 'd0 ;
            end

  endcase

end

//  --------------------------------
// Seq. logic
//  --------------------------------

always @( posedge slow_clk ) begin
   r_state <= c_state ;
end

endmodule
```

```
module clkdiv(clk,clk_out);

  input clk;
  output clk_out;

  reg [15:0] COUNT;

  assign clk_out=COUNT[15];

  always @(posedge clk)
  begin
    COUNT = COUNT + 1;
  end

endmodule
```

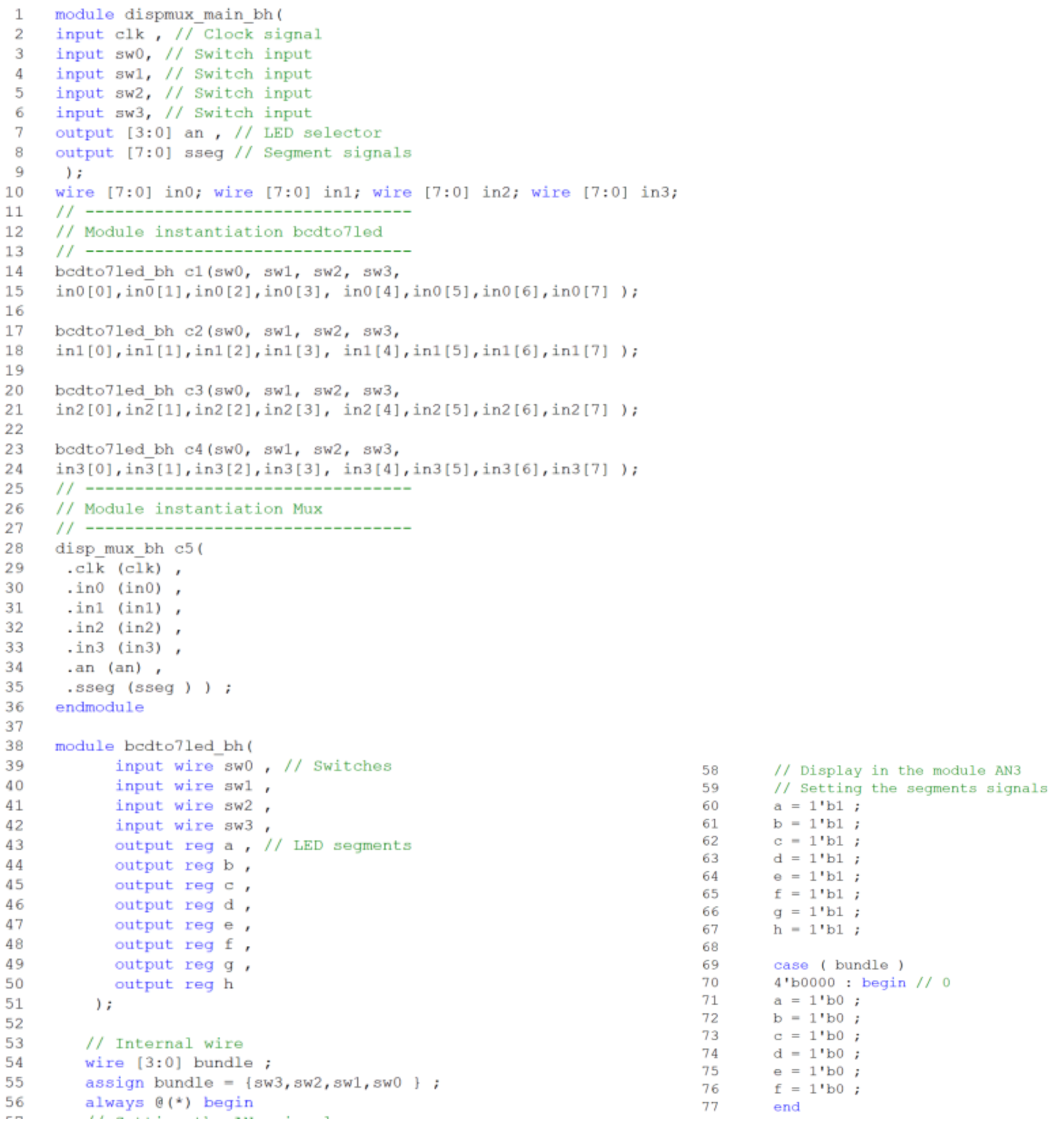
- UCF file Part2:

```
// Inputs
NET "clk" LOC = "B8";
NET "signal1" LOC = "A7";
// Outputs
NET "outedge" LOC = "M5";
```

- Simulation/Wave Form part 2:



- Source code part 3:

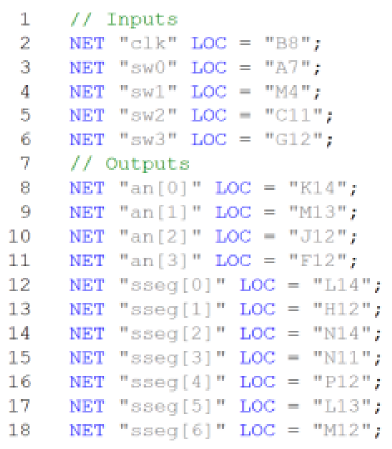


```verilog
module dispmux_main_bh(
input clk , // Clock signal
input sw0, // Switch input
input sw1, // Switch input
input sw2, // Switch input
input sw3, // Switch input
output [3:0] an , // LED selector
output [7:0] sseg // Segment signals
 );
wire [7:0] in0; wire [7:0] in1; wire [7:0] in2; wire [7:0] in3;
// --------------------------------
// Module instantiation bcdto7led
// --------------------------------
bcdto7led_bh c1(sw0, sw1, sw2, sw3,
in0[0],in0[1],in0[2],in0[3], in0[4],in0[5],in0[6],in0[7] );

bcdto7led_bh c2(sw0, sw1, sw2, sw3,
in1[0],in1[1],in1[2],in1[3], in1[4],in1[5],in1[6],in1[7] );

bcdto7led_bh c3(sw0, sw1, sw2, sw3,
in2[0],in2[1],in2[2],in2[3], in2[4],in2[5],in2[6],in2[7] );

bcdto7led_bh c4(sw0, sw1, sw2, sw3,
in3[0],in3[1],in3[2],in3[3], in3[4],in3[5],in3[6],in3[7] );
// --------------------------------
// Module instantiation Mux
// --------------------------------
disp_mux_bh c5(
 .clk (clk) ,
 .in0 (in0) ,
 .in1 (in1) ,
 .in2 (in2) ,
 .in3 (in3) ,
 .an (an) ,
 .sseg (sseg ) ) ;
endmodule

module bcdto7led_bh(
      input wire sw0 , // Switches
      input wire sw1 ,
      input wire sw2 ,
      input wire sw3 ,
      output reg a , // LED segments
      output reg b ,
      output reg c ,
      output reg d ,
      output reg e ,
      output reg f ,
      output reg g ,
      output reg h
    );

   // Internal wire
   wire [3:0] bundle ;
   assign bundle = {sw3,sw2,sw1,sw0 } ;
   always @(*) begin
```

```verilog
// Display in the module AN3
// Setting the segments signals
a = 1'b1 ;
b = 1'b1 ;
c = 1'b1 ;
d = 1'b1 ;
e = 1'b1 ;
f = 1'b1 ;
g = 1'b1 ;
h = 1'b1 ;

case ( bundle )
4'b0000 : begin // 0
a = 1'b0 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b0 ;
e = 1'b0 ;
f = 1'b0 ;
end
```

The code continues with all the cases of each number display..

- UCF file part 3:



```
// Inputs
NET "clk" LOC = "B8";
NET "sw0" LOC = "A7";
NET "sw1" LOC = "M4";
NET "sw2" LOC = "C11";
NET "sw3" LOC = "G12";
// Outputs
NET "an[0]" LOC = "K14";
NET "an[1]" LOC = "M13";
NET "an[2]" LOC = "J12";
NET "an[3]" LOC = "F12";
NET "sseg[0]" LOC = "L14";
NET "sseg[1]" LOC = "H12";
NET "sseg[2]" LOC = "N14";
NET "sseg[3]" LOC = "N11";
NET "sseg[4]" LOC = "P12";
NET "sseg[5]" LOC = "L13";
NET "sseg[6]" LOC = "M12";
```

CONCLUSION:

The purpose of the lab is to introduce us to sequential logic. Although the result is very similar to the previous lab, the processing of information is quite different. In the previous lab we have seen combinatorial logic. All input arrived at the same time. From now on, we have an overview of the sequential logic: data entered at the input of the first flip-flop propagates in the following flip-flops to each signal of the clock.

DISCUSSION:

The system works. We thought we had a problem because the 4 digits were lit. The lab manual seemed to indicate that the result should be a simple hexadecimal counter. We thought only one digit would come on. That said, we did not want to change the piece of code given in manual. After a few attempts, the teacher told us that everything was normal. The only other problem is always to navigate the software. We often have to use the manual of previous labs.

To improve our system, we could:

- Make a 4-digit hexadecimal counter rather than having the same digit on the 4 displays.
- Increase the frequency of the clock, to make our system even more responsive.

QUESTIONS:

1. What will happen if the “clock” signal is of very low frequency (1 Hz)? There will have a big delay between the moment the button is pressed to the result. More precisely, we will have to wait until a rising edge. For a 1hz frequency, we may wait 0 to 1 sec.

2. Design a test-bench and verify the logic performance. We must test successively every input of the truth table. The results are seen on our waveform above.