CSE460 - LAB 02 & 03

Name: F.M. Ashfaq

ID: 20101327

Section: 05

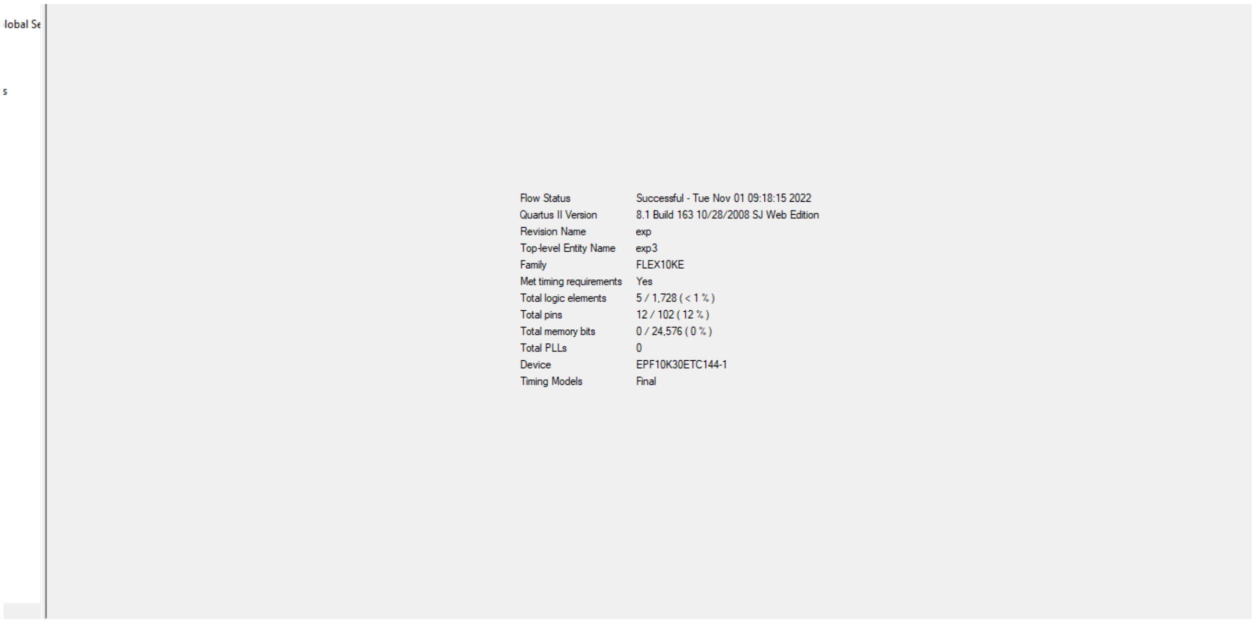
LAB 2

Problem statement: Design an 8 to 1 MUX using case statements in Verilog HDL.

Code:

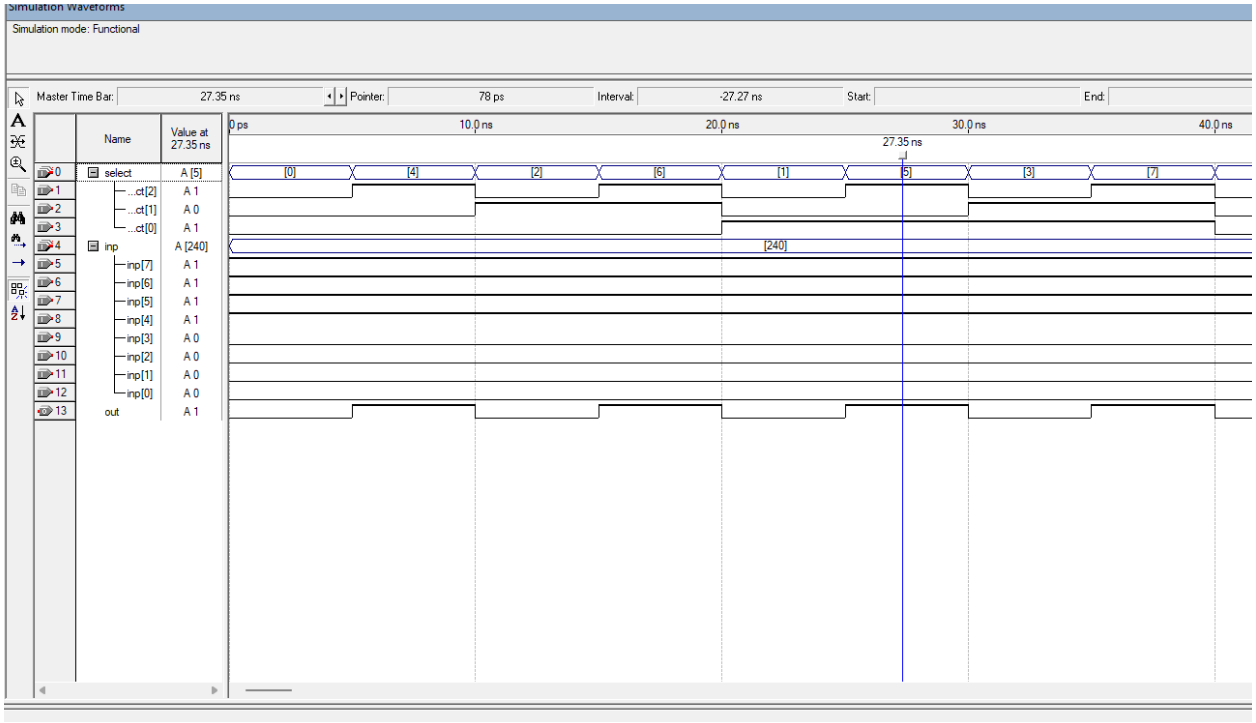
```
module exp3(out, select, inp);

        input[7:0] inp;
        input[2:0] select;
        output reg out;
        always@(*)
        begin
                case(select)
                        3'b000 : out = inp[0];
                        3'b001 : out = inp[1];
                        3'b010 : out = inp[2];
                        3'b011 : out = inp[3];
                        3'b100 : out = inp[4];
                        3'b101 : out = inp[5];
                        3'b110 : out = inp[6];
                        3'b111 : out = inp[7];
                endcase
        end
Endmodule
```

Compilation report:



Flow Status	Successful - Tue Nov 01 09:18:15 2022
Quartus II Version	8.1 Build 163 10/28/2008 SJ Web Edition
Revision Name	exp
Top-level Entity Name	exp3
Family	FLEX10KE
Met timing requirements	Yes
Total logic elements	5 / 1,728 (< 1 %)
Total pins	12 / 102 (12 %)
Total memory bits	0 / 24,576 (0 %)
Total PLLs	0
Device	EPF10K30ETC144-1
Timing Models	Final

Simulation report:



Simulation Waveforms

Simulation mode: Functional

Master Time Bar: 27.35 ns Pointer: 78 ps Interval: -27.27 ns Start: End:

	Name	Value at 27.35 ns
0	select	A [5]
1	...ct[2]	A 1
2	...ct[1]	A 0
3	...ct[0]	A 1
4	inp	A [240]
5	inp[7]	A 1
6	inp[6]	A 1
7	inp[5]	A 1
8	inp[4]	A 1
9	inp[3]	A 0
10	inp[2]	A 0
11	inp[1]	A 0
12	inp[0]	A 0
13	out	A 1

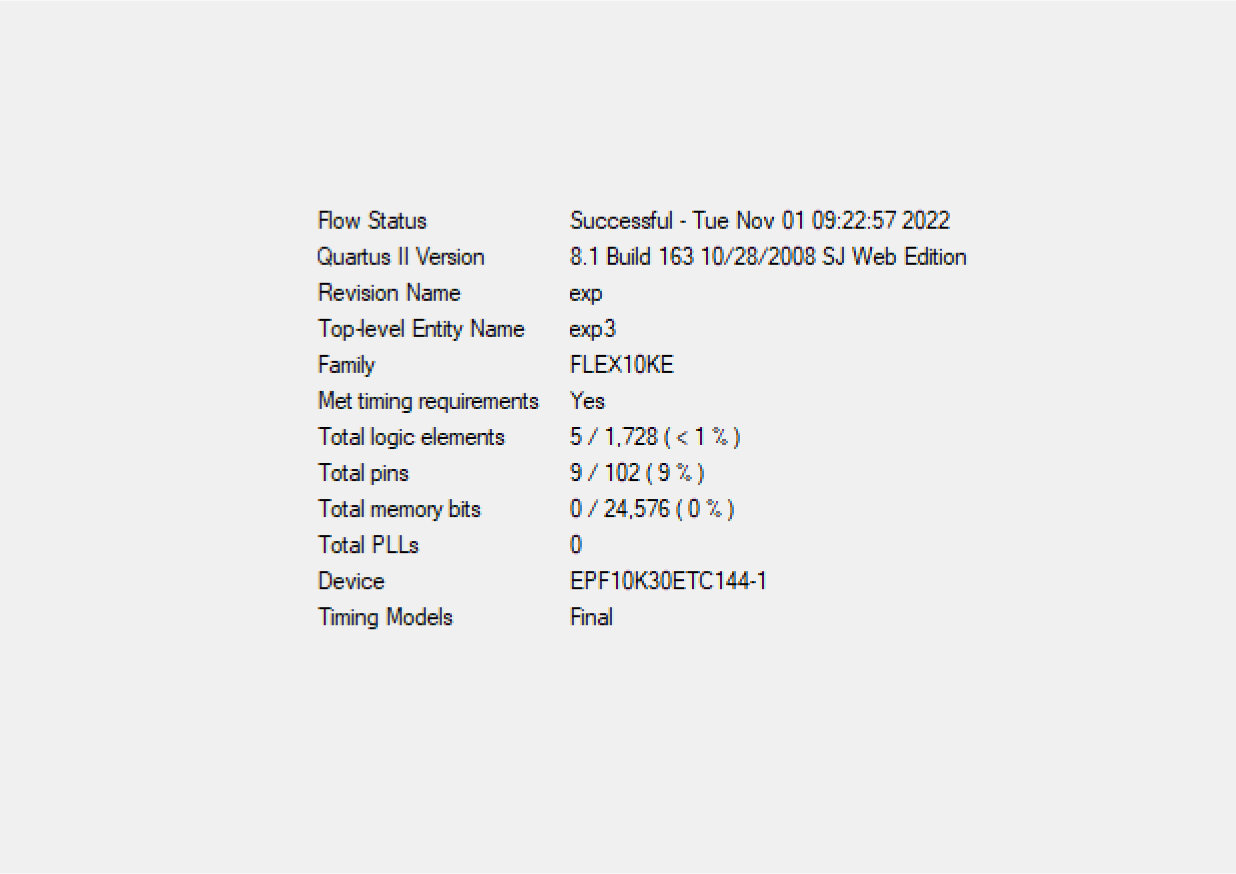
Discussion: We have created an 8 to 1 mux here. As a buffer, Mux chooses the bits as necessary. So, the selection pins are required. With the use of the selector bit, we may choose an input, and the output will include that particular input. For instance, if the selection pin returns a 3, we choose the input-3 pin. We will likewise see high on the output pin if input-3 is in the high condition.

LAB 3

Code:

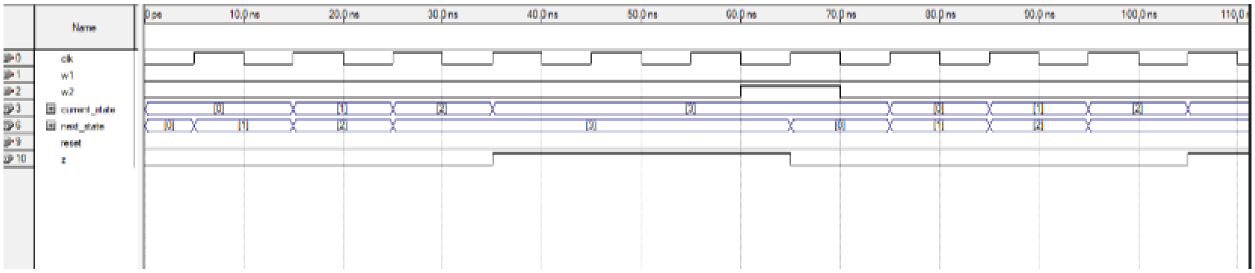
```
module exp3(clk, reset, w1, w2, z, current_state, next_state);
   input clk, reset, w1, w2;
   output reg z;
   output reg [1:0] current_state, next_state;
   parameter [1:0] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;
   always @(posedge clk, posedge reset)
   begin
      if(reset == 1)
      begin
         current_state = A;
         next_state = A;
         z = 0;
      end
      else
      begin
         current_state = next_state;
         case(current_state)
            A:    if(w1==w2)
               begin
                  next_state = B;
                  z = 0;
               end
               else
               begin
                  next_state = A;
                  z = 0;
               end
            B:    if(w1==w2)
               begin
                  next_state = C;
                  z = 0;
               end
               else
               begin
                  next_state = A;
                  z = 0;
               end
            C:    if(w1==w2)
               begin
            next_state = D;
            z = 0;
          end
          else
          begin
            next_state = A;
            z = 0;
          end
        D:    if(w1==w2)
          begin
            next_state = D;
            z = 1;
          end
          else
          begin
            next_state = A;
            z = 0;
          end
      endcase
    end
  end
endmodule
```

Compilation report:



Flow Status	Successful - Tue Nov 01 09:22:57 2022
Quartus II Version	8.1 Build 163 10/28/2008 SJ Web Edition
Revision Name	exp
Top-level Entity Name	exp3
Family	FLEX10KE
Met timing requirements	Yes
Total logic elements	5 / 1,728 (< 1 %)
Total pins	9 / 102 (9 %)
Total memory bits	0 / 24,576 (0 %)
Total PLLs	0
Device	EPF10K30ETC144-1
Timing Models	Final

Diagram:



Discussion: Now, we will construct a mealy type sequential circuit that receives two inputs and returns a high value when the same input is received for four consecutive clock cycles. We have four states here, and when a match is discovered, we go on to the next state. If we reach the fourth state and still earn a match, it will be our fourth consecutive contest. As a result, we self-loop to the fourth state and set the output to high. It will remain high as long as we continue to receive the same feedback. If we detect a non-matching input in any of the states, we return to the initial state and reset the process.