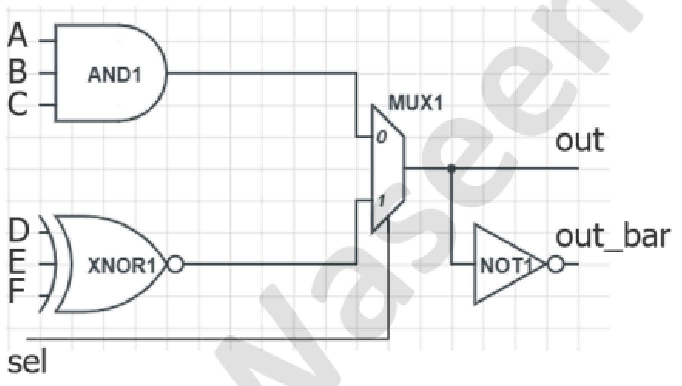
abdalrahman taha Assignment_1

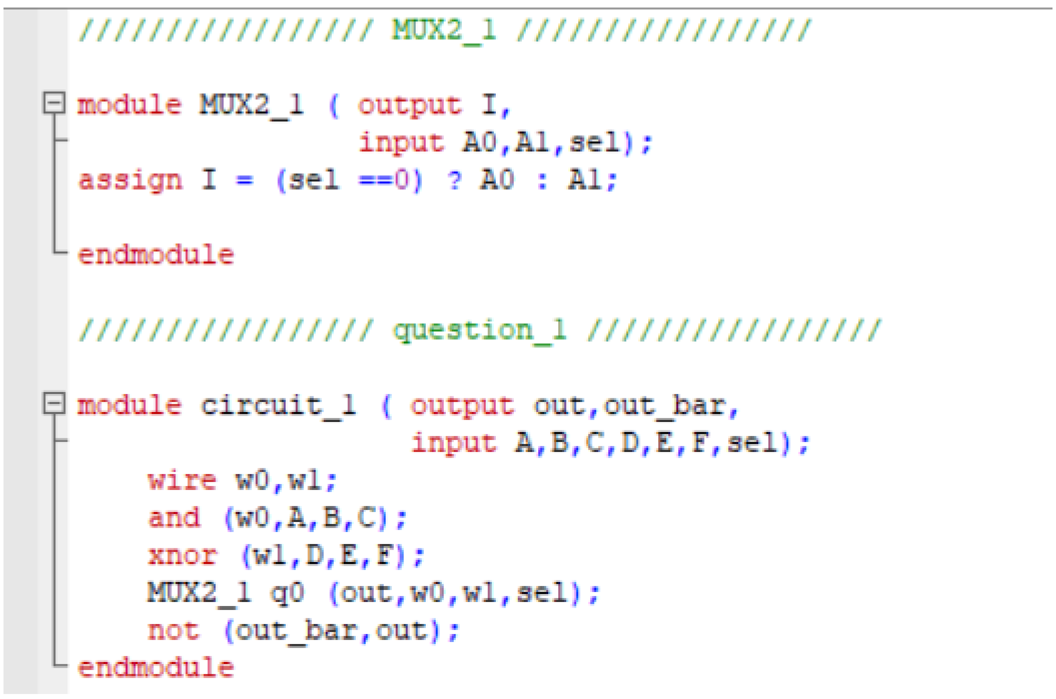
Q1

Design the following circuits with Verilog using assign statements

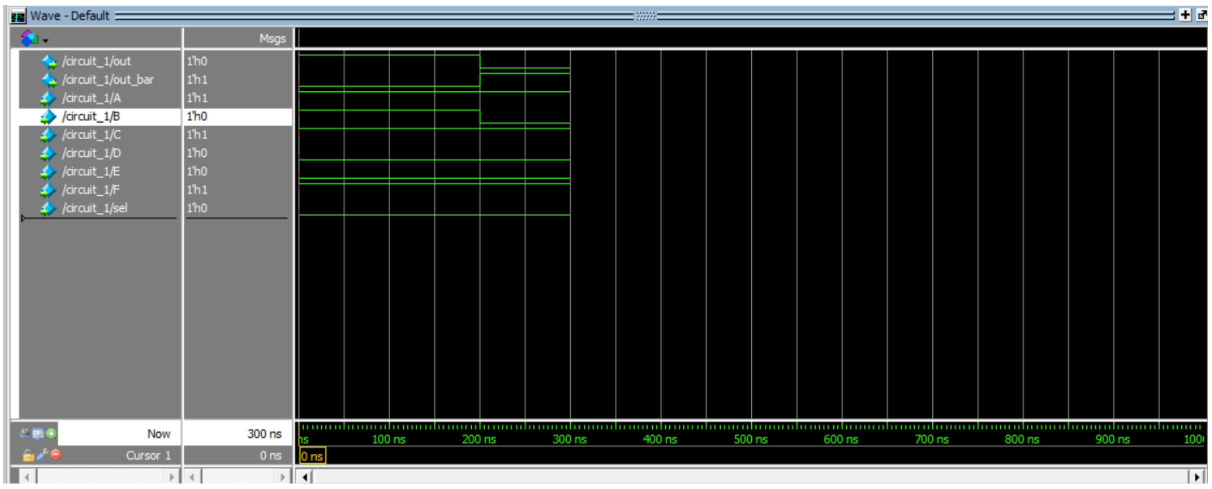
1)

- The design has 7 inputs and 2 outputs
- Use assign statements to design the following





```
////////////////// MUX2_1 //////////////////

module MUX2_1 ( output I,
                input A0,A1,sel);
assign I = (sel ==0) ? A0 : A1;

endmodule

////////////////// question_1 //////////////////

module circuit_1 ( output out,out_bar,
                   input A,B,C,D,E,F,sel);
    wire w0,w1;
    and (w0,A,B,C);
    xnor (w1,D,E,F);
    MUX2_1 q0 (out,w0,w1,sel);
    not (out_bar,out);
endmodule
```

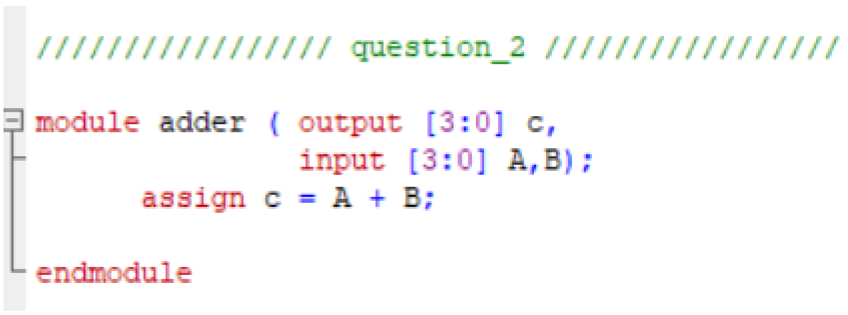


```
Wave - Default
                        Msgs
/circuit_1/out          1'h0
/circuit_1/out_bar      1'h1
/circuit_1/A            1'h1
/circuit_1/B            1'h0
/circuit_1/C            1'h1
/circuit_1/D            1'h0
/circuit_1/E            1'h0
/circuit_1/F            1'h1
/circuit_1/sel          1'h0

Now        300 ns
Cursor 1   0 ns
```

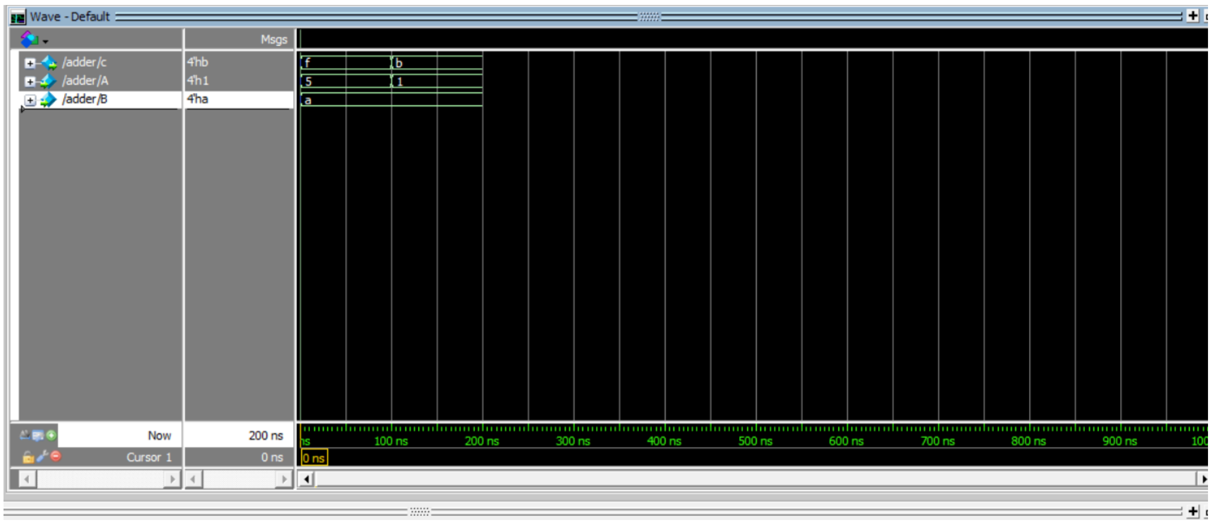
Q2

2) Implement 4-bit adder using addition operator and assign statement

- The design takes 2 inputs (**A, B**) and the summation is assigned to output (**C**) ignoring the carry



```
///////////////// question_2 /////////////////

module adder ( output [3:0] c,
               input [3:0] A,B);
      assign c = A + B;

endmodule
```



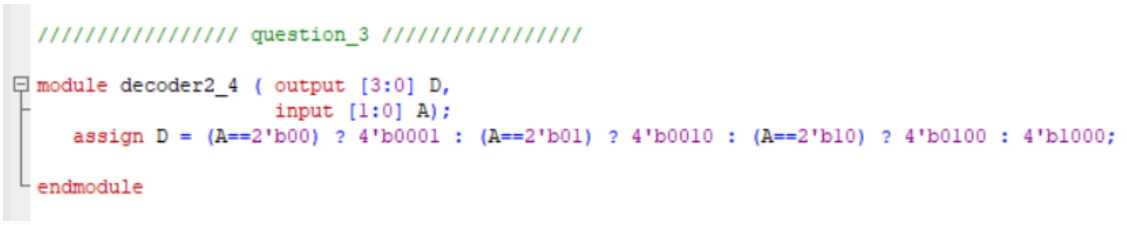
```
Wave - Default
                 Msgs
/adder/c         4'hb
/adder/A         4'h1
/adder/B         4'ha

Now        200 ns
Cursor 1   0 ns
```

Q3

3) Implement 2-to-4 Decoder using conditional operator (A logic decoder has n input lines and 2^n output lines. Each output line corresponds to a unique combination of the input values.)

- The design has input **A** (2 bits) and output **D** (4 bits)
- you can use the following format for the conditional operator.
- assign <output_signal> = <condition1> ? <value1> : <condition2> ? <value2> : <default_value>);

A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

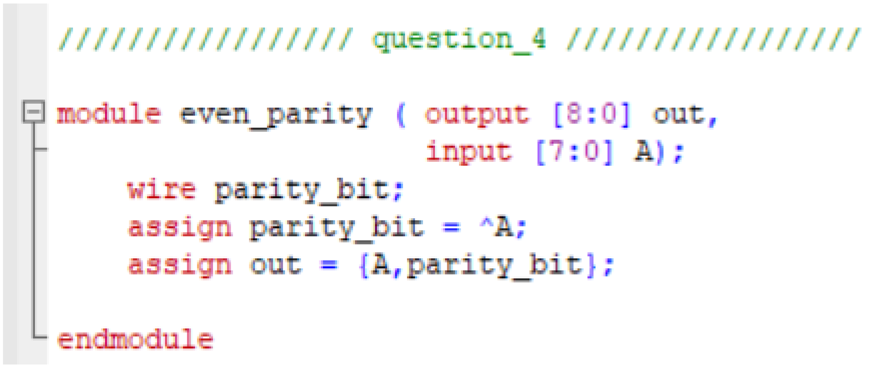


```
////////////////// question_3 /////////////////

module decoder2_4 ( output [3:0] D,
                    input  [1:0] A);
   assign D = (A==2'b00) ? 4'b0001 : (A==2'b01) ? 4'b0010 : (A==2'b10) ? 4'b0100 : 4'b1000;

endmodule
```

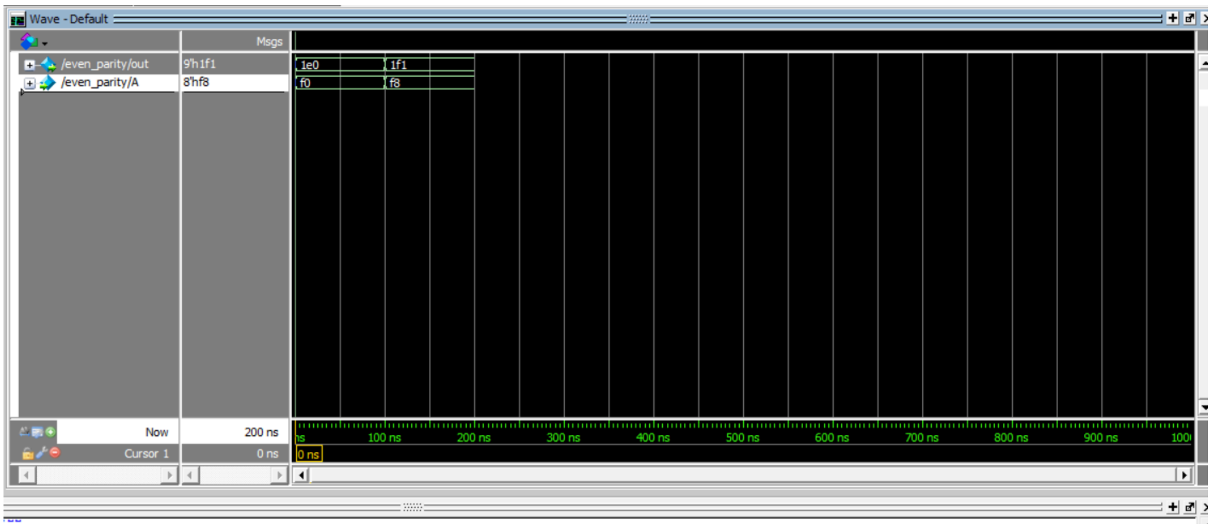
Q4

4) Implement an even parity generator module using assign statement. In case you don't know what a parity bit is, please check this link. The design input is a bus where a reduction operator will be used to generate the even parity bit.

- The design has 1 input **A** (8 bits) and 1 output **out_with_parity** (9 bit) where the parity bit calculated will be inserted in the least significant bit of the output bus and the remaining bits will be the input A (Hint: use concatentation).



```
////////////////// question_4 //////////////////

module even_parity ( output [8:0] out,
                     input  [7:0] A);
    wire parity_bit;
    assign parity_bit = ^A;
    assign out = {A,parity_bit};

endmodule
```



Wave - Default

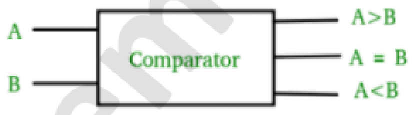
	Msgs		
/even_parity/out	9'h1f1	1e0	1f1
/even_parity/A	8'hf8	f0	f8

Now 200 ns
Cursor 1 0 ns

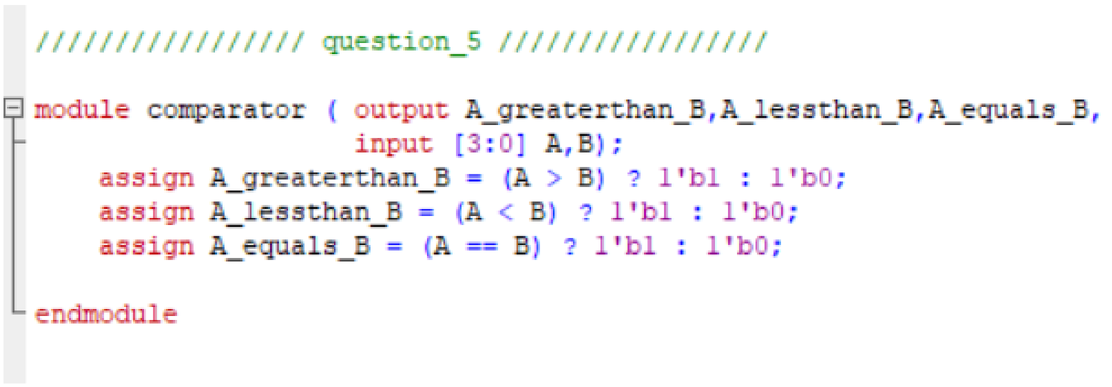
Q5

5) Implement a comparator that compares 2 inputs (**A, B**) and has 3 outputs using conditional operator.

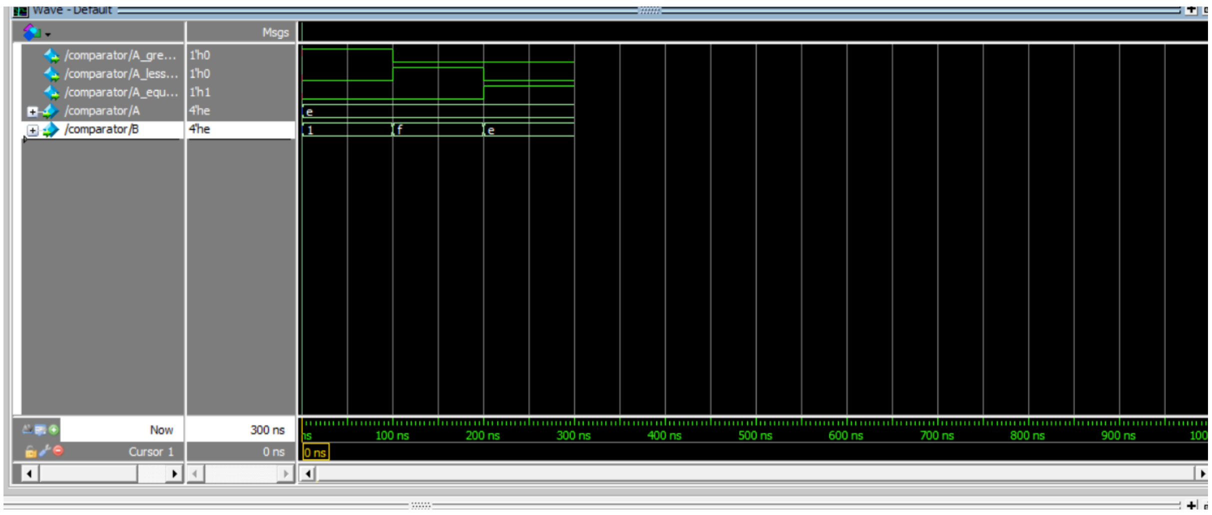
- The first output **A_greaterthan_B** is high only when A is greater than B
- The second output **A_equals_B** is high only when A equals B
- The third output **A_lessthan_B is** high only when A is less than B



Inputs A and B are 4-bit bus while the 3 outputs are single bits.



```
///////////////// question_5 /////////////////

module comparator ( output A_greaterthan_B,A_lessthan_B,A_equals_B,
                    input [3:0] A,B);
    assign A_greaterthan_B = (A > B) ? 1'b1 : 1'b0;
    assign A_lessthan_B = (A < B) ? 1'b1 : 1'b0;
    assign A_equals_B = (A == B) ? 1'b1 : 1'b0;

endmodule
```



```
Wave - Default
                          Msgs
/comparator/A_gre...      1'h0
/comparator/A_less...     1'h0
/comparator/A_equ...      1'h1
/comparator/A             4'he      e
/comparator/B             4'he      1        f        e

Now                     300 ns
Cursor 1                  0 ns
```