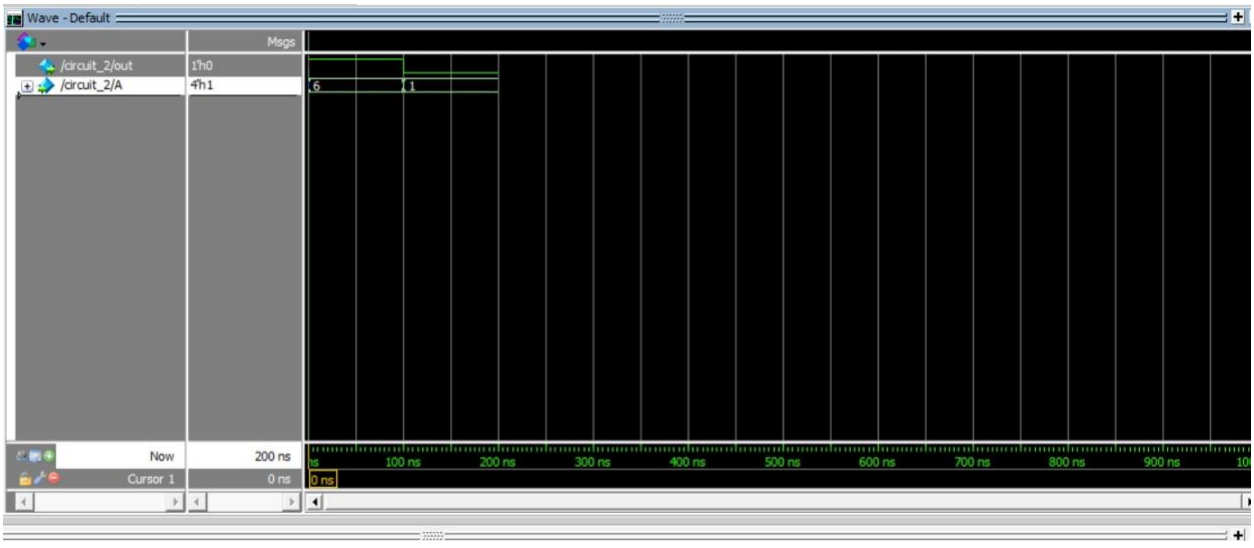
Q1

1) A four-bit binary number is represented as A3A2A1A0, where A3, A2, A1, and A0 represent the individual bits and A0 is equal to the LSB. Design a logic circuit using Verilog that will produce a HIGH output whenever the binary number is greater than 0010 and less than 1000.

- The design takes 1 input **A** (4-bits) and output **out** (1-bit)

```
////////////////////// question_1 ////////////////////////

module circuit_2 ( output out,
                   input [3:0] A);
    assign out = (A > 4'b0010 && A < 4'b1000) ? 1'b1 : 1'b0;

endmodule
```

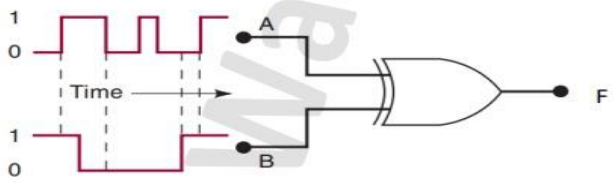


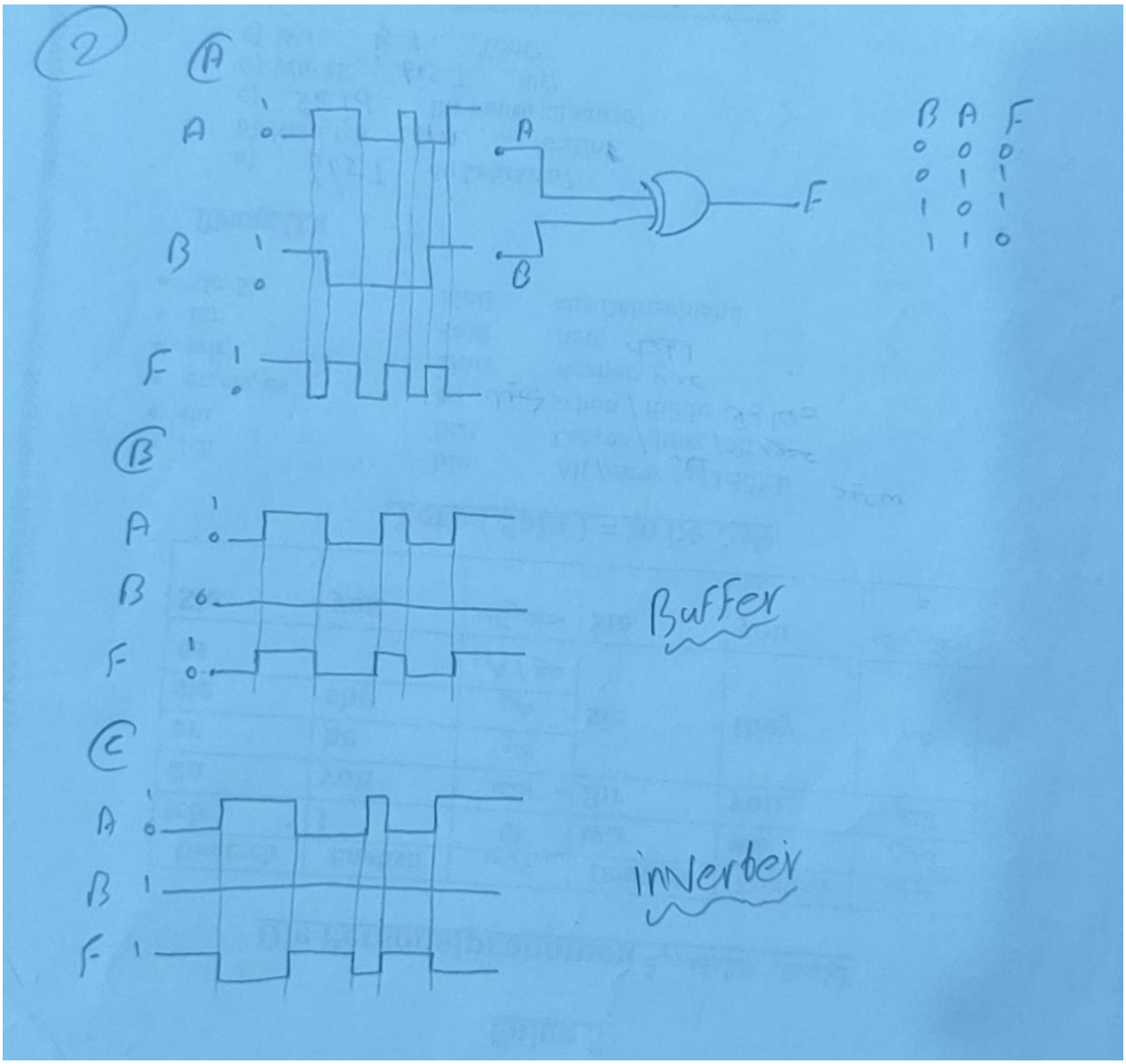
Q2

2) (a) Draw with your pen on a piece of paper the output waveform F for the circuit of Figure below.

(b) Repeat with the B input held LOW.

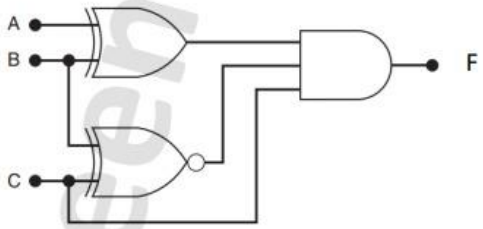
(c) Repeat with B held HIGH.



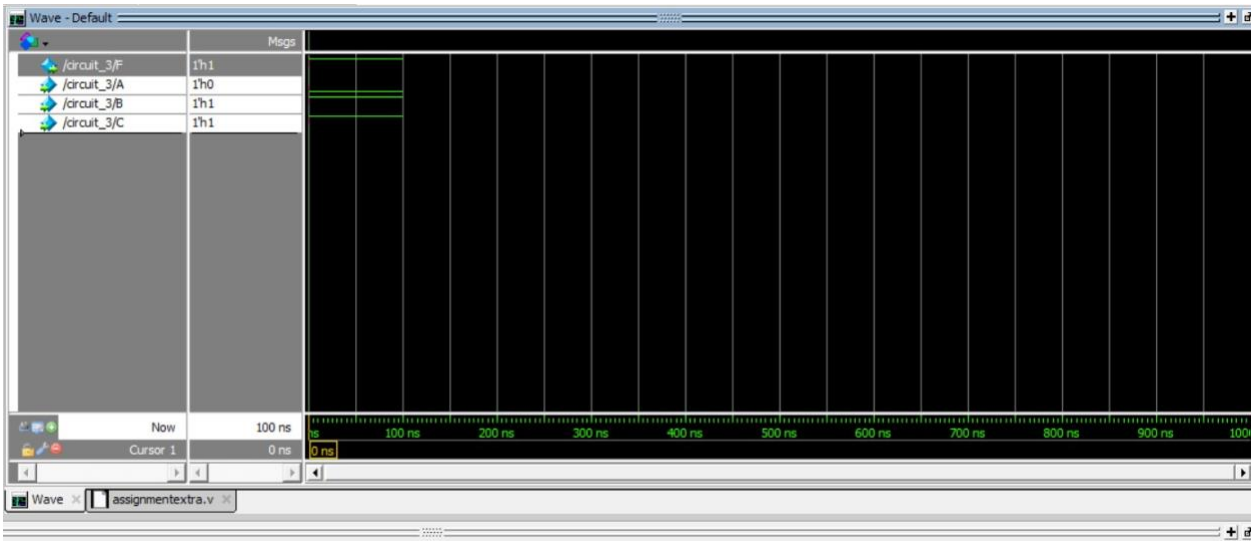


Q3

3) Design the following circuit using Verilog and determine the input conditions needed to produce F = 1



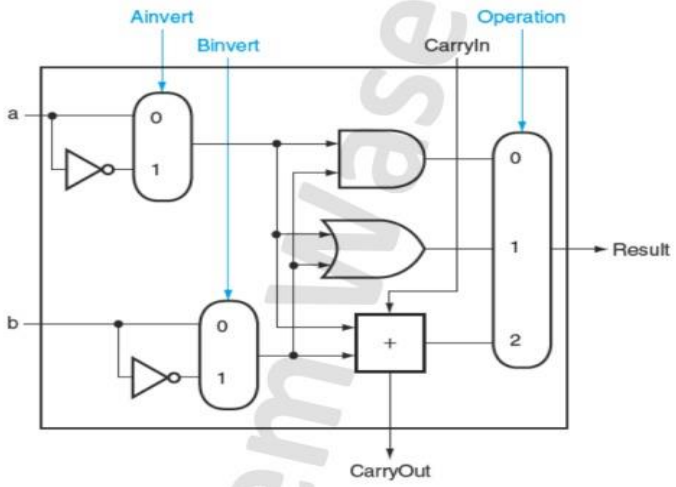
```
//////////////////// question_3 ////////////////////////

module circuit_3 ( output F,
                   input A,B,C);

    assign F = (A^B)&(B~^C)&(C);

    /* F=1 when A=0 B=1 C=1 */

endmodule
```



Q4

4) Implement the following 1-bit ALU. If you are unfamiliar with the concept of an ALU, you can find more information by clicking here. Use conditional operator for the multiplexers. For the 3-to-1 Mux, you can use the following format for the conditional operator.

assign <output_signal> = <condition1> ? <value1> : <condition2> ? <value2> : <default_value>);



Port Name	Type	Size	Description
A	Input	1 bit	Input a
B			Input b
Ainvert			Select signal for the multiplexer to select a or a complement
Binvert			Select signal for the multiplexer to select b or b complement
CarryIn			Carry in
Operation		2 bits	Select signal for the multiplexer to drive the Result output
CarryOut	Output	1 bit	Carry out
Result		1 bit	Output of the multiplexer

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

module ALU ( output reg result,carry_out,
             input A,B,Ainvert,Binvert,carry_in,
             input [1:0] operation);
      wire w0,w1;
      MUX2_1 e (w0,A,~A,Ainvert);
      MUX2_1 f (w1,B,~B,Binvert);
      always @(*) begin
        case (operation)
            2'b00: result = w0&w1;
            2'b01: result = w0|w1;
            2'b10: {carry_out,result} = w0+w1+carry_in;
            default: result = 0;
        endcase
      end

endmodule
```