Assignment_1_Extended

(under supervision of Eng : Kareem Waseem)

Q1]

this circuit take 4-bits input A and check if

2<A<8 therefore ,the output(out) will be active high (1)

else the output will be equal to zero.

```
module question1_ext(A , out);

input [3:0] A ;

output out ;

assign out = ((A>4'b0010)&&(A<4'b1000))?1:0 ;

endmodule
```

Figure (1) : Verilog code

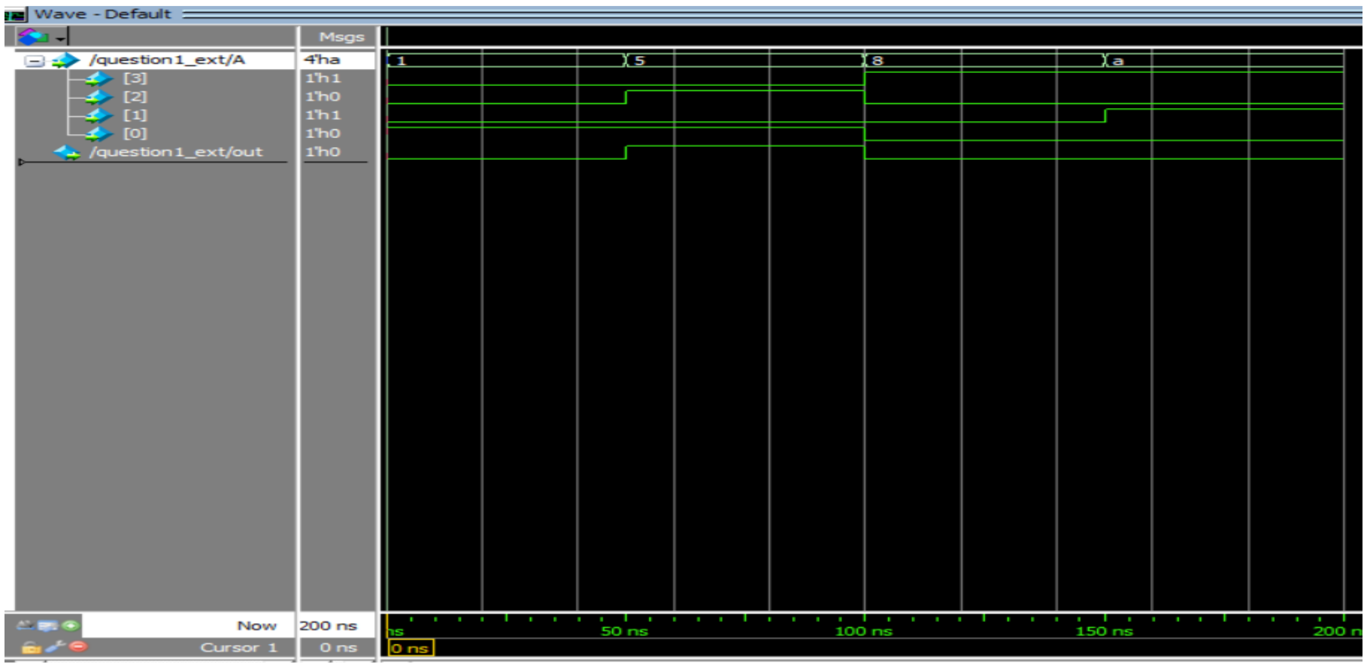

Figure (2) : waveform

clock cycle	input	output
First cycle	0001 (1)	0
Second cycle	0101 (5)	1
Third cycle	1000 (8)	0
Fourth cycle	1010 (10)	0

This table is obtained from the simulated waveform as shown in figure (2)

Q2] part a)

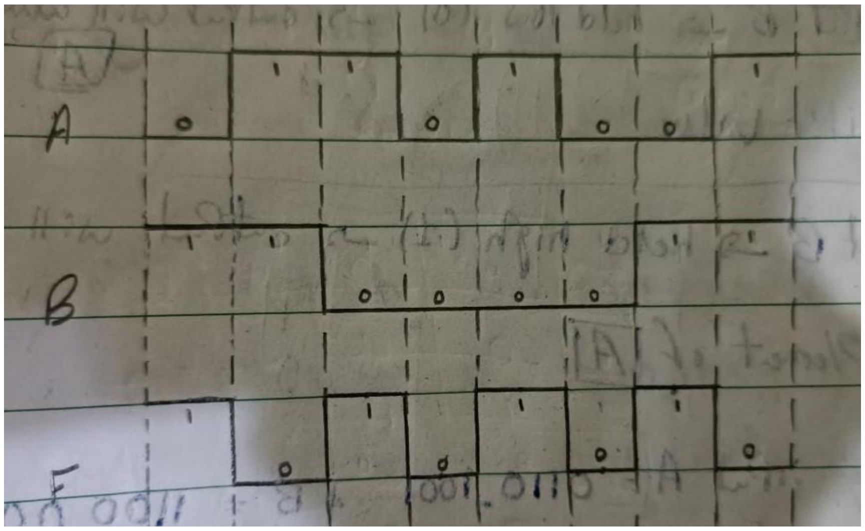

Figure (4) : logic diagram for part (a)

```
// this code for part (a)
module question2_ext(A,B,F);
input A,B ;
output F ;
assign F=A^B;
endmodule
```

Figure (3) : verilog code

- *Note : in part (a) we assume input A = 0110_1001 & input B = 1100_0011*

Part b)

if the input B held low (0) $\Rightarrow$ output F will always be equal to **input A** regardless it's value.

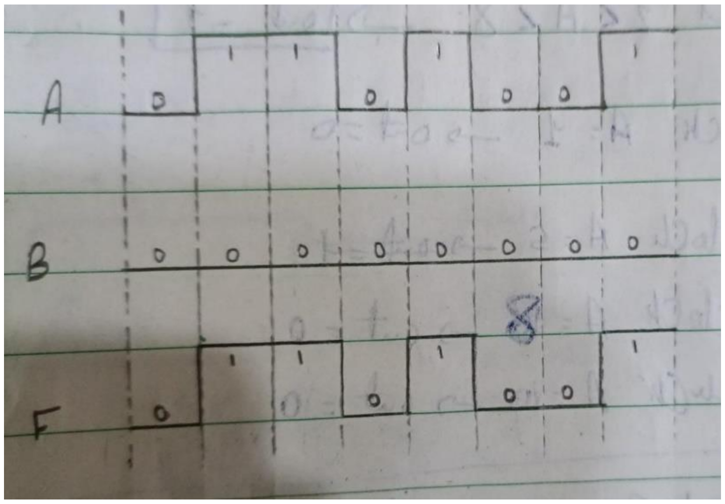

Figure (5) : logic diagram for part (b)

part c)

if the input B held high (1) $\Rightarrow$ Output F will be equal to **complement of A**

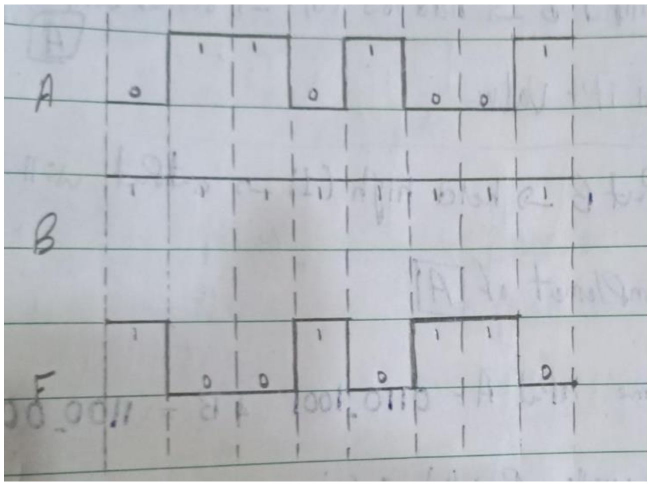

Figure (6) : logic diagram for part (c)

Q3] After implementing the truth table for the logic diagram

We found that the input condition that make the output F

is active high (1) → **Input condition : A=0 , B=1 , C=1**

- where the input A is the MSB and C is the LSB

```
module question3_ext(A,B,C,F);

input A,B,C;

output F;

wire w1,w2;

assign w1 = A^B ;

assign w2 = ~(B^C) ;

assign F = w1&w2&C ;

endmodule
```

Figure (7) : verilog code

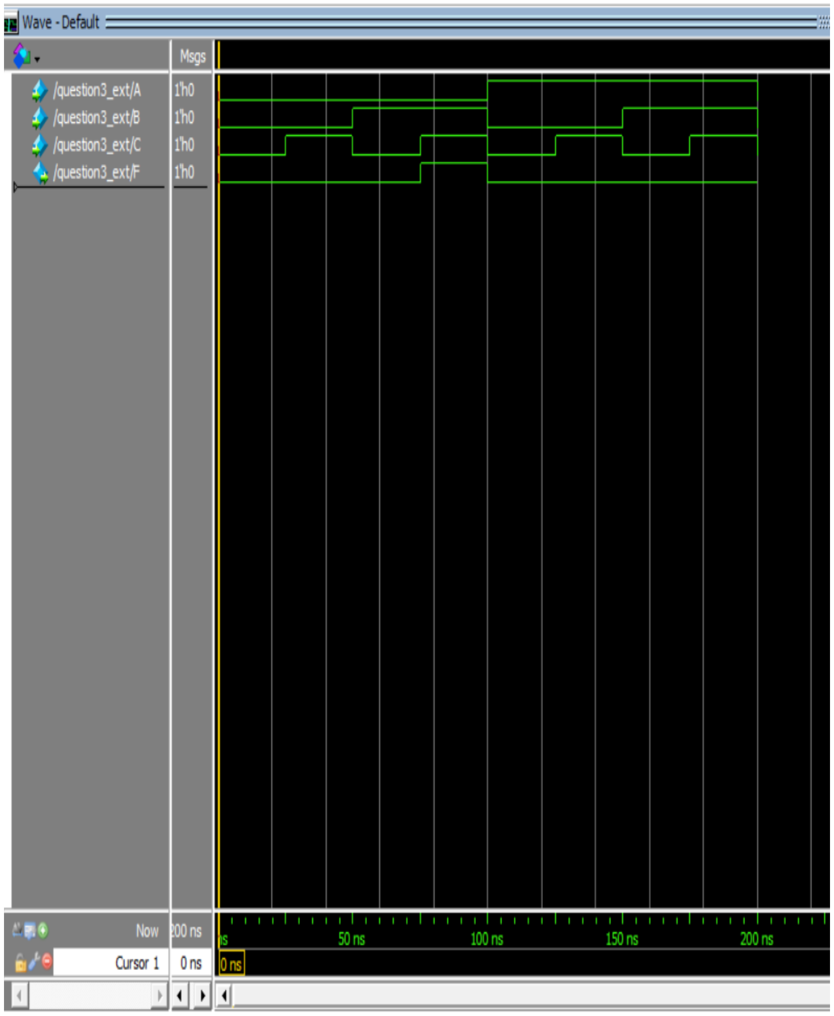

Figure (8) : waveform

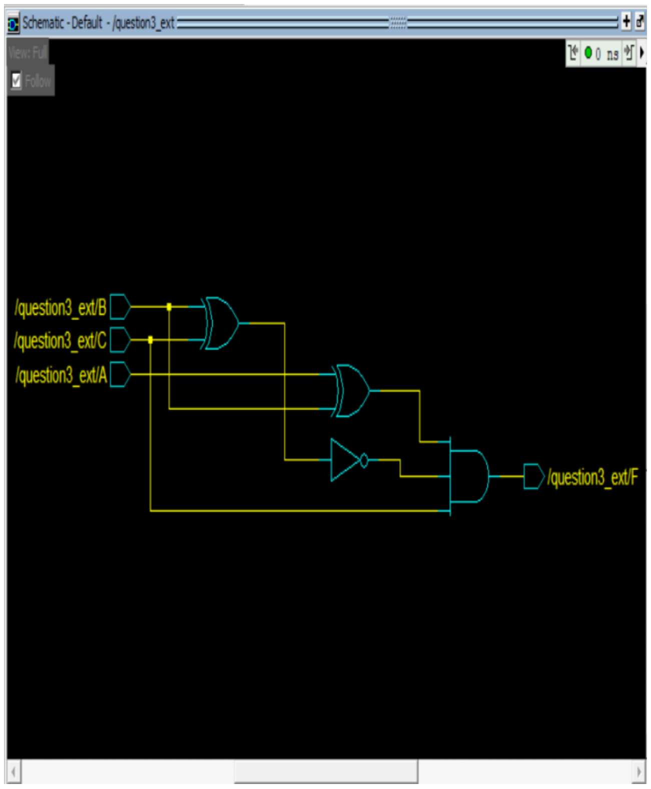

Figure (9) : schematic

-Figure (8&9) shows the waveform and the schematic of this circuit to check if the circuit operate well or not.

Q4] This circuit act as prime number detector where if the input is prime the output will be active high (1) else the output will be low (0) .

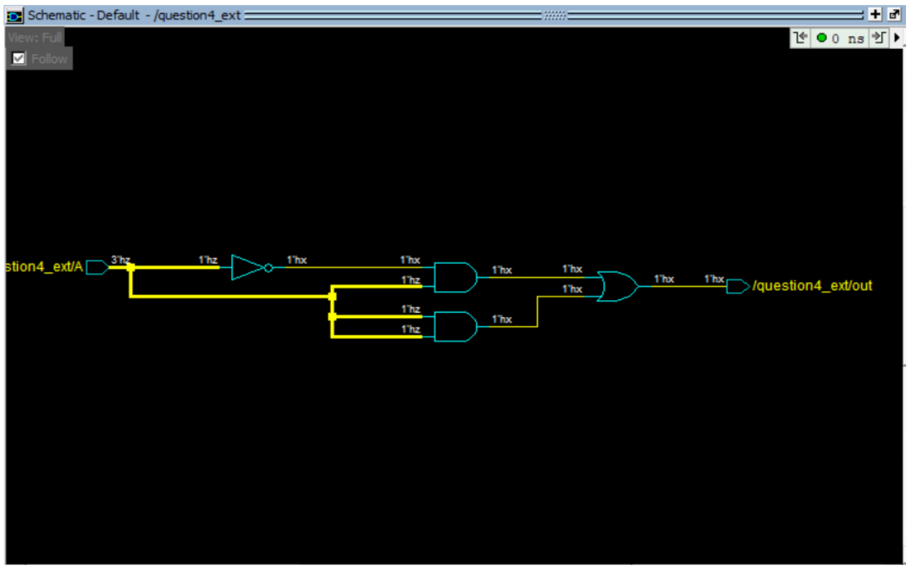

Figure (11) : schematic

```
module question4_ext(A,out);

input [2:0] A;

output out;

wire w1,w2 ,w3 ;

assign w1=~A[2];

assign w2=A[1]&w1;

assign w3=A[0]&A[2];

assign out=w2|w3;

endmodule
```

Figure (10) : Verilog code

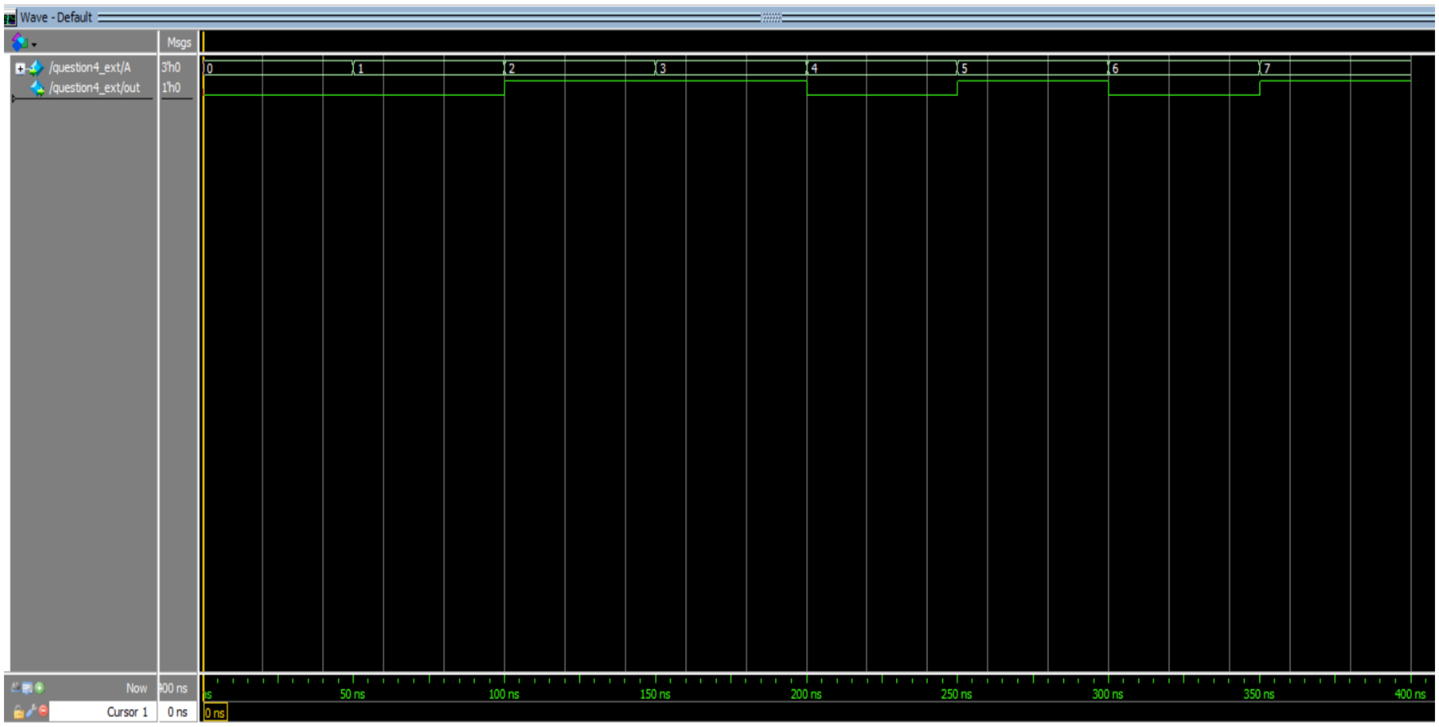

Figure (12) : waveform

Q5] This circuit take two "4-bit" inputs (A,B) and produce two "4-bit" outputs (F,K) Where the role of this circuit is to compare the inputs and assign the largest value to F and the smallest value to K

```
module question5_ext(A,B,F,K);

input [3:0] A,B ;

output [3:0] F,K ;

assign F =(A>B)?A:B;

assign K =(A>B)?B:A;

endmodule
```

Figure (13) : verilog code

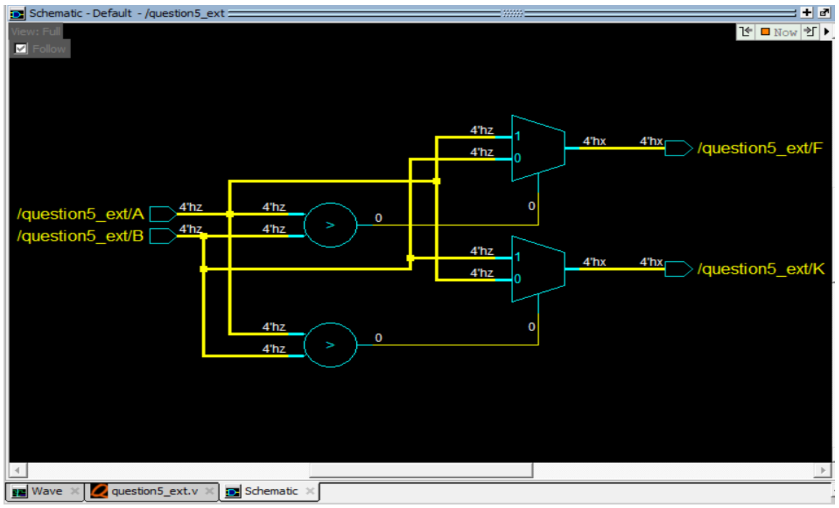

Figure (14) : schematic

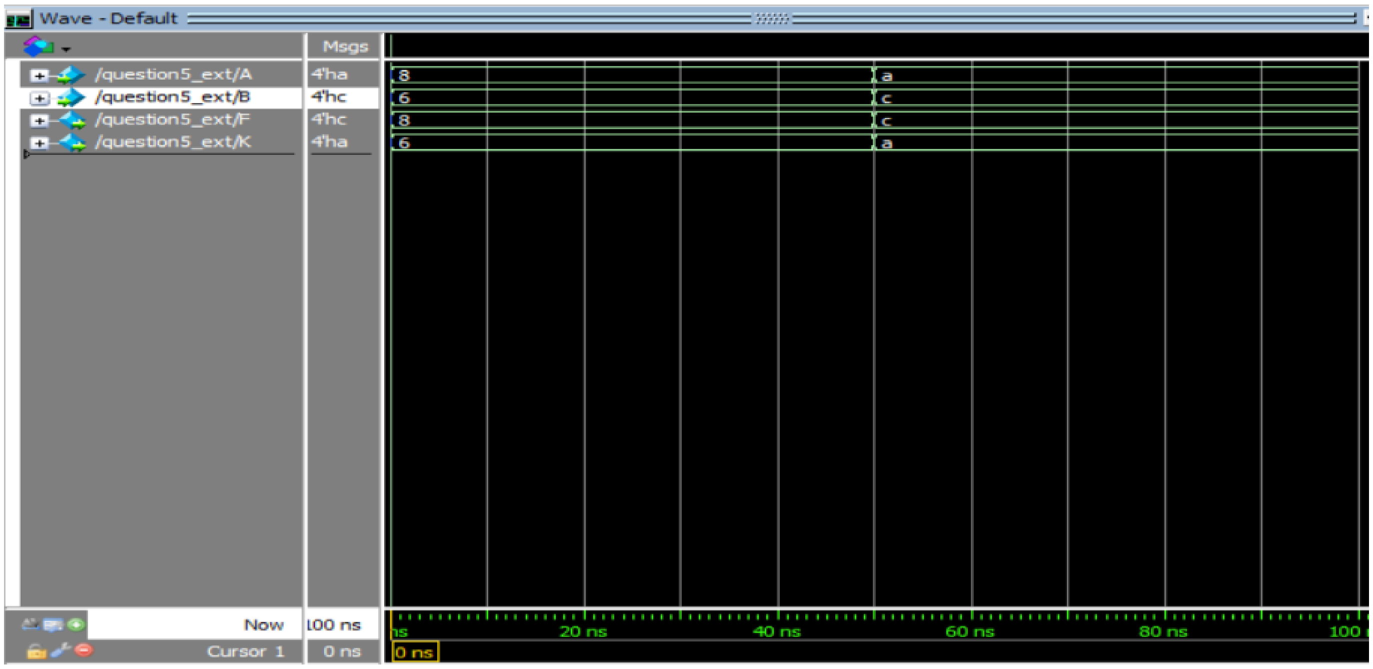

Figure (15) : waveform

Clock cycle	A	B	F	K
First cycle	1000 (8)	0110 (6)	1000	0110
Second cycle	1010 (10)	1100 (12)	1100	1010

This table explain the above waveform that shown in figure (15)

Q6] this circuits takes two 4-bit inputs (Ta,Tb) where the Tb measure the outgoing air from the air conditioning while Ta measures the incoming air temp to the air conditioning

- **therefore the Tb must be less than or equal to Ta to make the**
- **air conditioning perform well**

```
//this design circuit known as check
module question6_ext (Ta,Tb,on);
input [3:0] Ta,Tb ;
output on ;
assign on = (Tb<=Ta)?1:0;
endmodule
```

Figure (16) : Verilog code

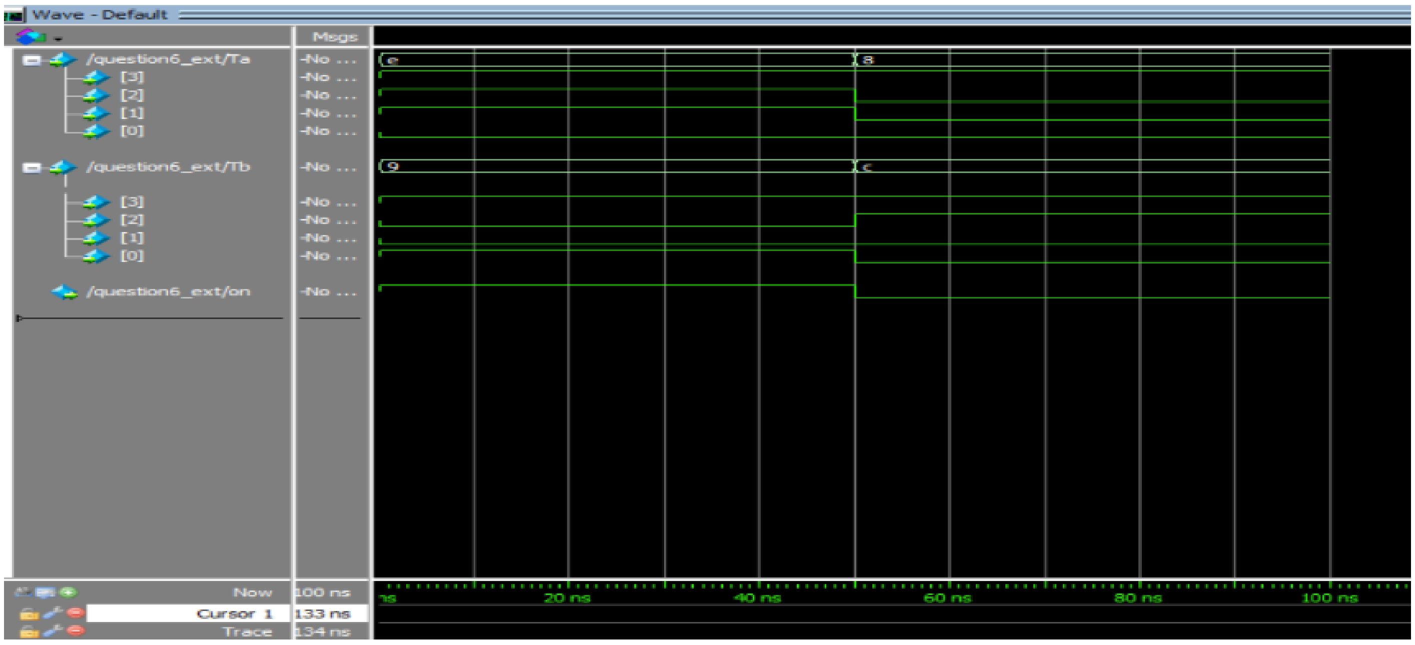

Figure (17) : waveform

Clock cycle	input (Ta)	input (Tb)	Output (on)
First cycle	1110	1001	1
Second cycle	1000	1100	0

This table explain the waveform shown above in figure (17)

Q7]

```
// this code describe simple ALU block for 1-bit inputs
module question7_ext(A,B,Ainvert,Binvert,Cin,operation,result,Cout);   //Cin --> carry in  & Cout --> carry out
input A,B,Ainvert,Binvert,Cin;                      // Ainvert & Binvert are two selection lines for different MUXs
input [1:0] operation;
output result,Cout;
wire w1,w2,w3,w4,w5,w6,w7 ;              // these wires represent the internal signals of the ALU
assign w1=~A;
assign w2=~B;
assign w3 = (Ainvert==0)?A:w1;
assign w4 = (Binvert==0)?B:w2;
assign w5=w3&w4;
assign w6=w3|w4;
assign w7=w3^w4^Cin;                         //this is the design of the full adder where w7 act as sum
assign Cout=(w3&w4)|(Cin&(w3^w4));
assign result =(operation==2'b00)?w5:(operation==2'b01)?w6:w7 ;
endmodule
```

Figure (18) : verilog code

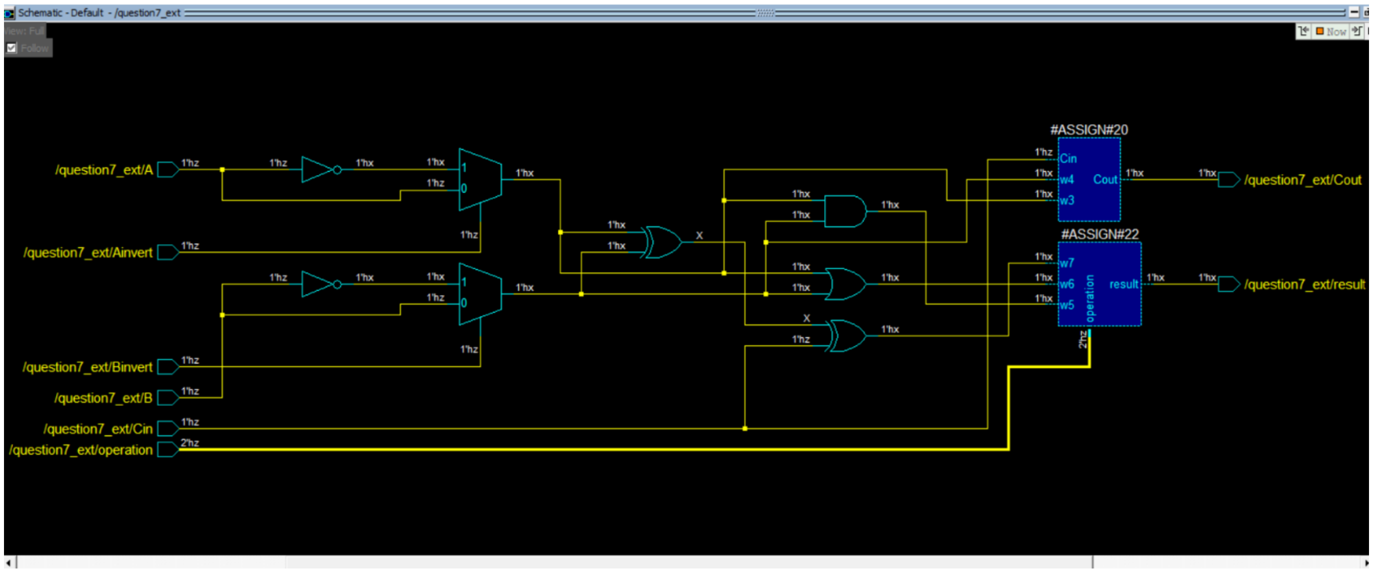

Figure (19) : schematic design for the ALU given