Assignment1 Extended

==

Name: Ahmed Nour

Sheet ID: 16

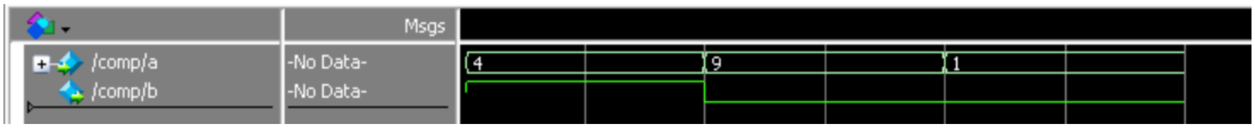
==

Q1)

Code:

```
module comp (a ,b);

input [3:0] a;

output b;


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


endmodule
```

Simulation:

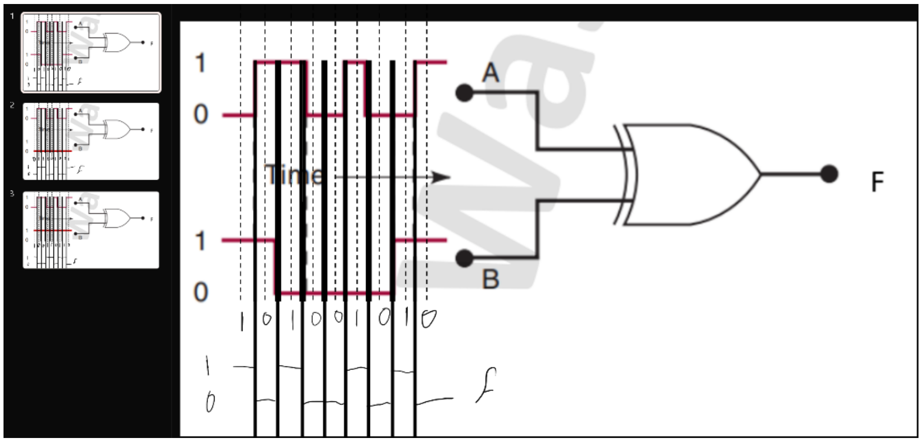


Q2)

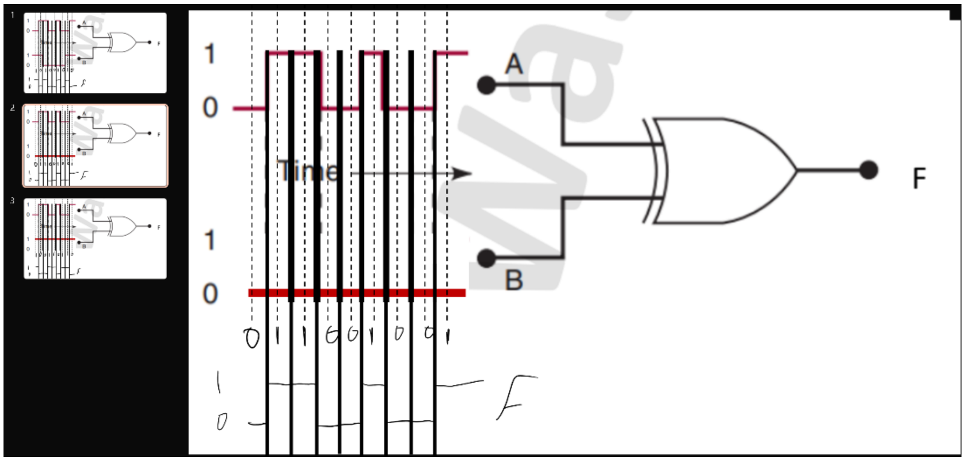
To solve the question, we need to divide the clock into the smallest clock of all of them to make sure that there will not be any clock has 0 and 1 at the same moment.

Note: the bold lines are the dividers, the dashed lines test the value at the clock.

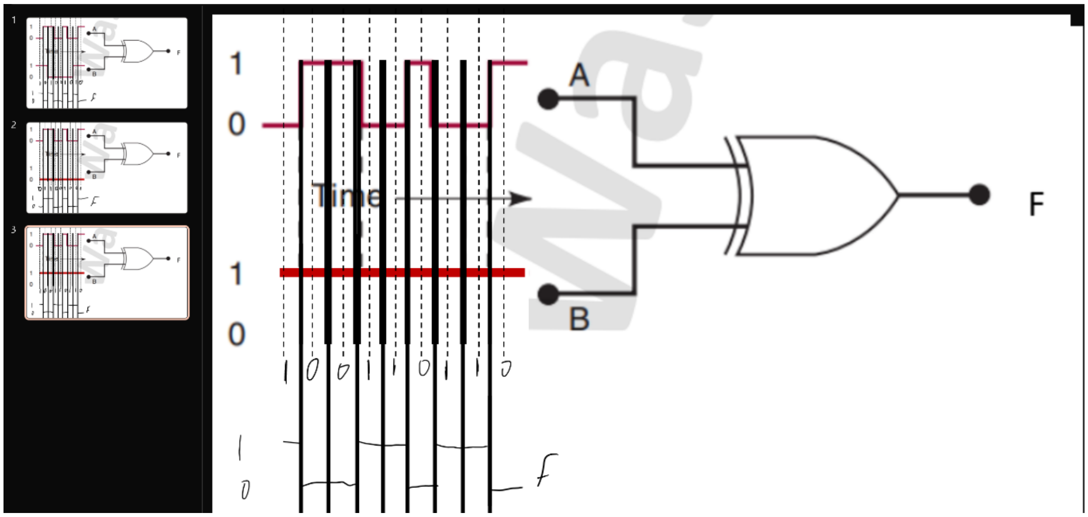
a)



b)



c)

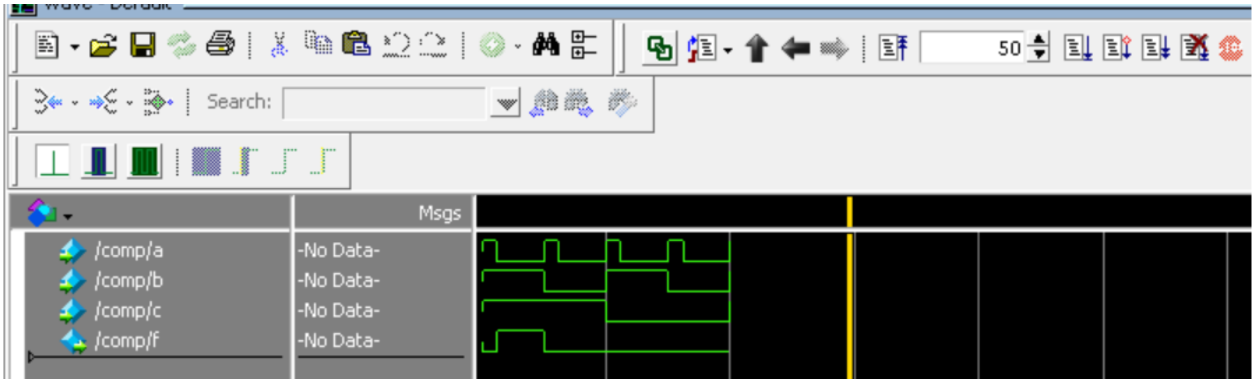


Q3)

Code:

```
module comp (a, b, c, f);

input  a, b, c;

output f;


assign f = ((a ^ b) & (b ~^ c) & c);


endmodule
```

Simulation:



```
/comp/a   -No Data-
/comp/b   -No Data-
/comp/c   -No Data-
/comp/f   -No Data-
```

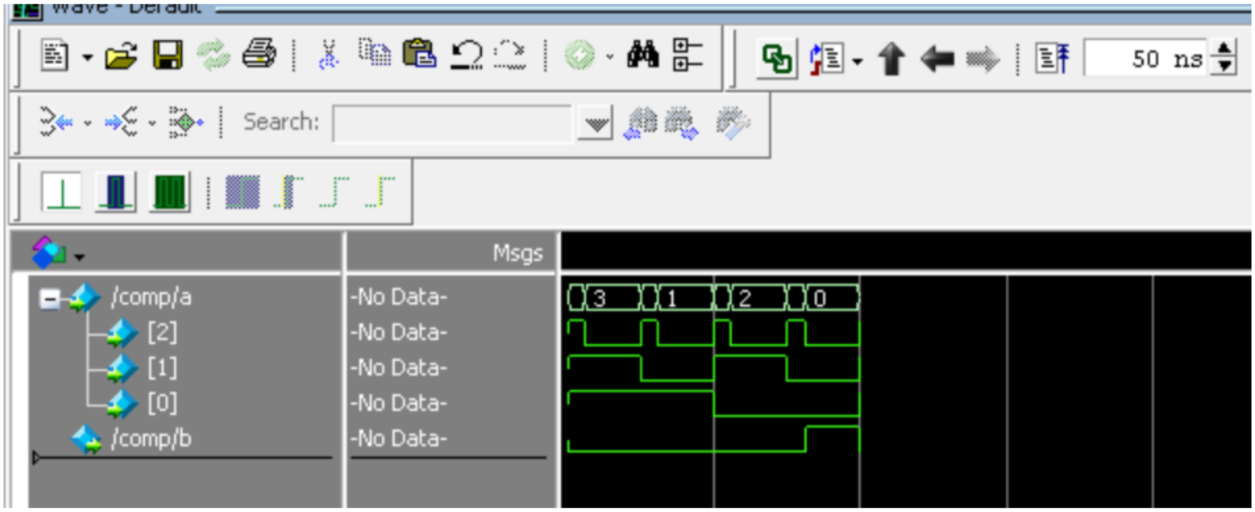
Note: As shown in the simulation the only case that justify f = 1 when a= 0, b = 1 and c = 1.

Q4)

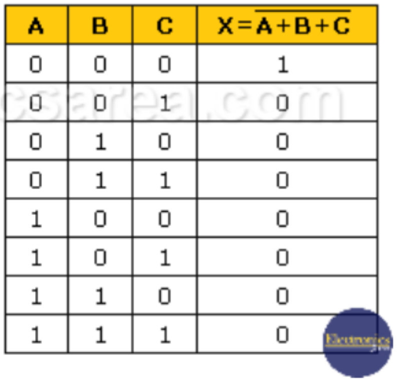
Code:

```
module comp (a ,b);

input [2:0] a;

output b;


assign b = ~|a;


endmodule
```

Simulation:



Note: To get 1 only when all bits are 0s and using one gate we use nor due to truth table and the output of the simulation.



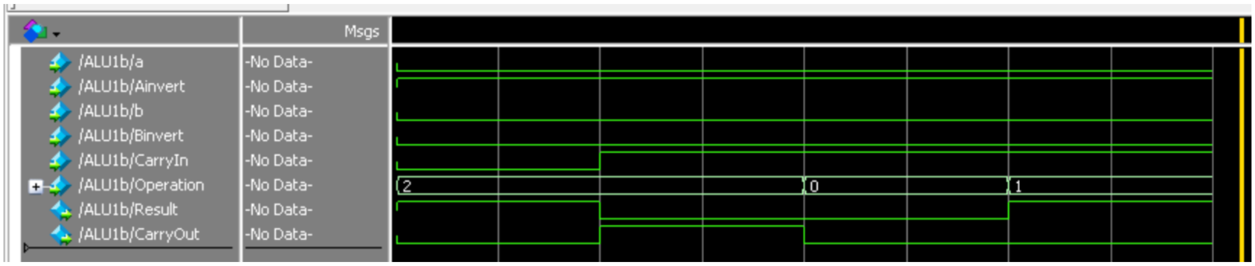
A	B	C	$X=\overline{A+B+C}$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Q5)

Code:

```
module ALU1b (a, b, Ainvert, Binvert, CarryIn, Operation, Result, CarryOut);

input  a, b, Ainvert, Binvert, CarryIn;

input  [1:0] Operation;

output Result, CarryOut;

wire at , bt;

assign at = (Ainvert)? ~a : a;

assign bt = (Binvert)? ~b : b;

assign {CarryOut, Result} = (Operation == 0)? (at & bt) : (Operation == 1)? (at | bt) : (at + bt + CarryIn);


endmodule
```

Simulation:



	Msgs
/ALU1b/a	-No Data-
/ALU1b/Ainvert	-No Data-
/ALU1b/b	-No Data-
/ALU1b/Binvert	-No Data-
/ALU1b/CarryIn	-No Data-
/ALU1b/Operation	-No Data-
/ALU1b/Result	-No Data-
/ALU1b/CarryOut	-No Data-