

يوسف أحمد محمد ابراهيم

Assignment 1

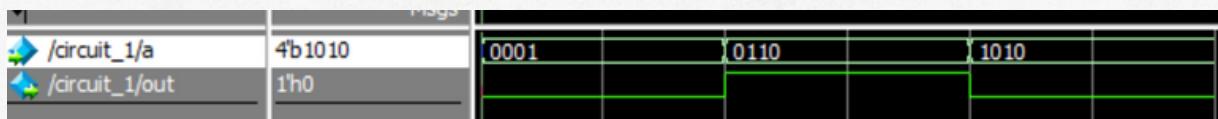
Extra

Circuit_1 :

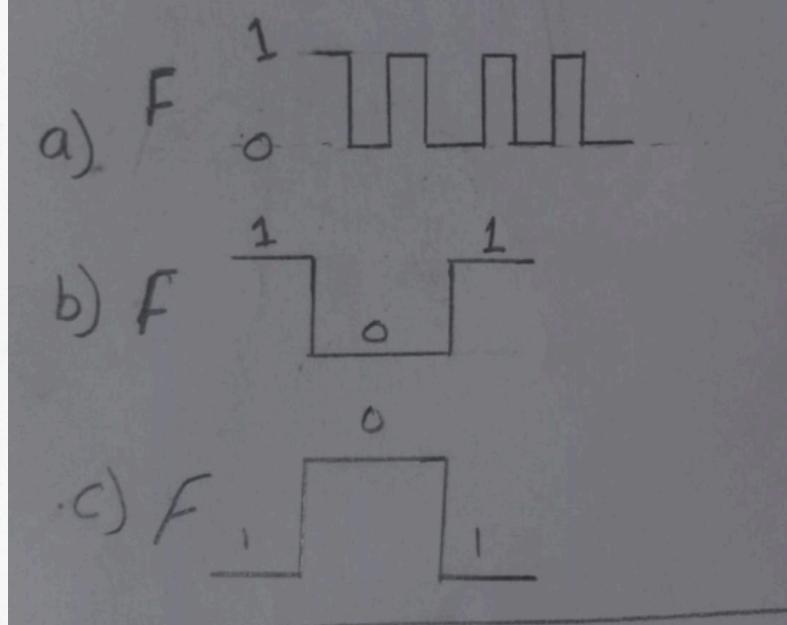
- module:

```
1 module circuit_1(input [3:0]a, output out);
2 |   assign out = (a < 4'b1000 && a > 4'b0010) ? (1'b1) : (0);
3 endmodule
```

- Output:



2.

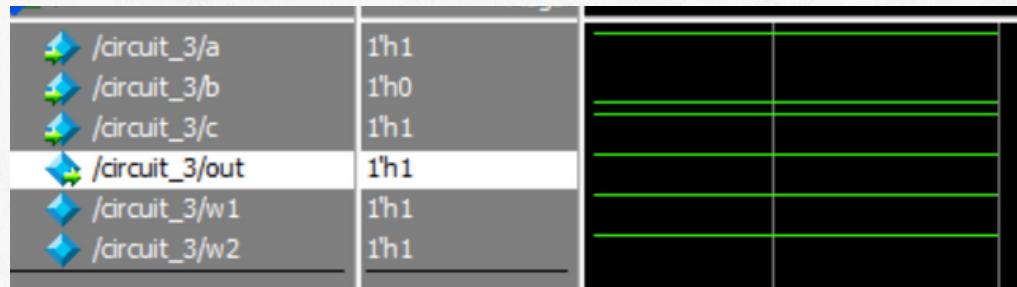


Circuit_3:

- module

```
1  module circuit_3(input a, b, c, output out);
2      xor (w1, a, b);
3      xnor (w2, c, b);
4      and (out, c, w1, w2);
5  endmodule
```

- waveform:

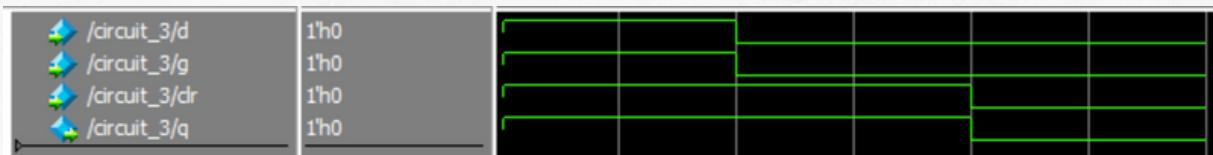


Circuit_3:

- code

```
1  module circuit_3(input d, g, clr, output q);
2  always @(*) begin
3      if(~clr)
4          q = 0;
5      else if(g)
6          q = d;
7  end
8  endmodule
```

- output:



$$\begin{aligned} & (a\bar{b} + \bar{a}b) \cdot (\bar{b}\bar{c} + \bar{b}c) \dots c \\ &= a\bar{b}c + \bar{a}bc \end{aligned}$$

if out = 1 at

$a = 1$
$b = 0$
$c = 1$

4. Using nand gate

Circuit_5:

- code

```
1 module circuit_5(input a, b, ainv, binv, [1:0]op, carryin, output reg result);
2
3     reg wirea, wireb;
4
5
6     always @(*) begin
7         if(ainv)
8             wirea = ~a;
9         else
10            wire a = a;
11         if(binv)
12             wireb = ~b;
13         else
14            wire b = b;
15
16         case (op)
17             2'b0 : result = wirea & wireb;
18             2'b1 : result = wirea | wireb;
19             default : result = wirea + wireb + carryin;
20         endcase
21     end
22 endmodule
```

- output:

	Msgs
/circuit_5/a	1'h1
/circuit_5/b	1'h0
/circuit_5/ainv	1'h0
/circuit_5/binv	1'h0
+ /circuit_5/op	2'h0
+ /circuit_5/carryin	2'h0
/circuit_5/result	1'h0
/circuit_5/wirea	1'h1
/circuit_5/wireb	1'h0