Digital Logic Circuit Design
ModelSim-Verilog Practice - 2

Ex.1. Bool's Law

D:/altera/13.0/bool_law.v - Default

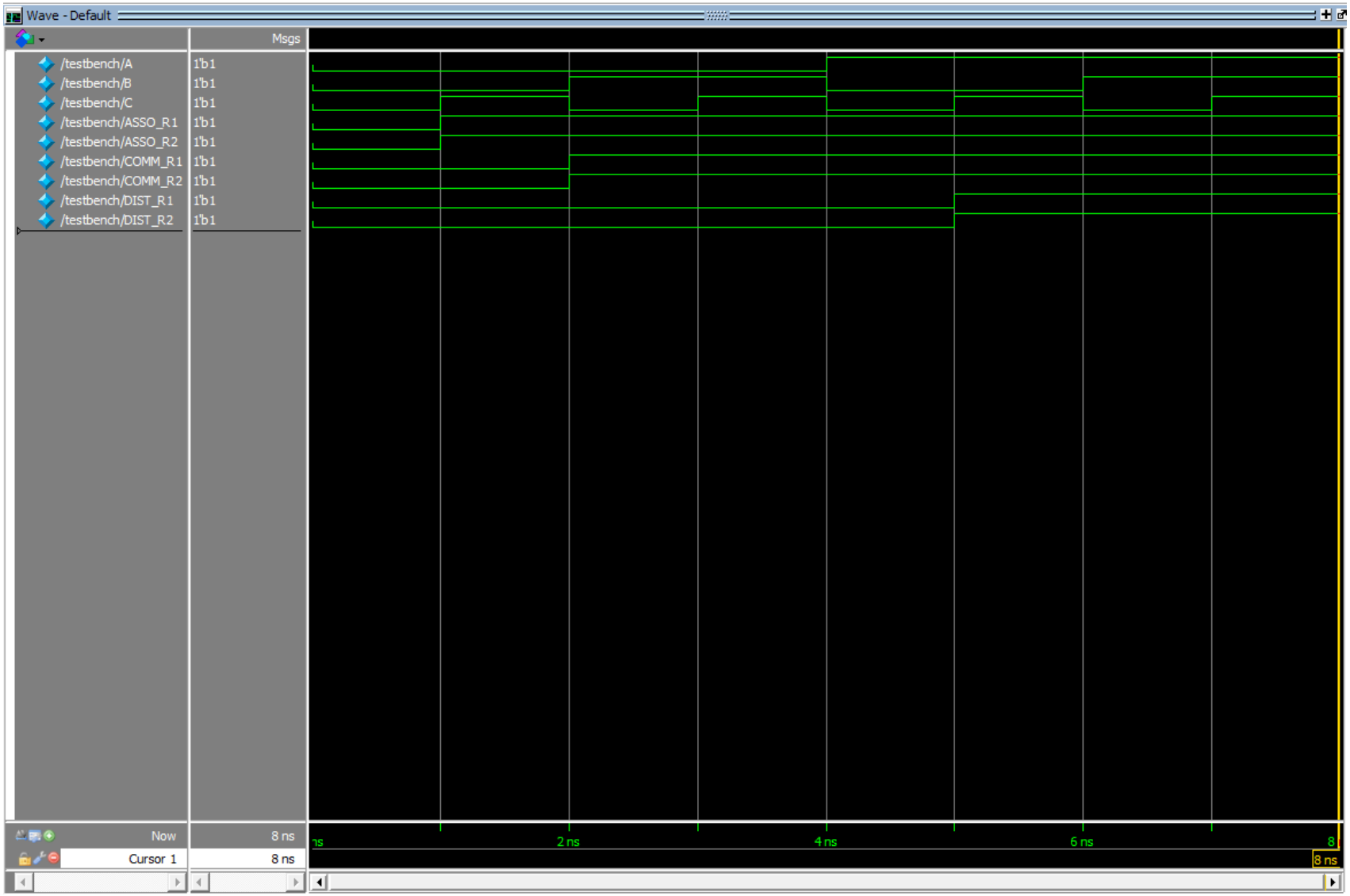
```
Ln#
 1  module BOOL_LAW(A, B, C, COMM_R1, COMM_R2, ASSO_R1, ASSO_R2, DIST_R1, DIST_R2);
 2          input A, B, C;
 3          output COMM_R1, COMM_R2, ASSO_R1, ASSO_R2, DIST_R1, DIST_R2;
 4
 5          // Commutativity
 6          assign COMM_R1 = A | B;
 7          assign COMM_R2 = B | A;
 8
 9          // Associativity
10          assign ASSO_R1 = A | (B | C);
11          assign ASSO_R2 = (A | B) | C;
12
13          // Distributivity
14          assign DIST_R1 = A & (B | C);
15          assign DIST_R2 = (A & B) | (A & C);
16  endmodule
```

Ex.1. Bool's Law

D:/altera/13.0/testbench.v (/testbench) - Default

```
Ln#
1   `timescale lns/lns
2
3   module testbench();
4           reg A,B,C;
5           wire COMM_R1, COMM_R2, ASSO_R1, ASSO_R2, DIST_R1, DIST_R2;
6
7           BOOL_LAW myboollaw(A, B, C, COMM_R1, COMM_R2, ASSO_R1, ASSO_R2, DIST_R1, DIST_R2);
8
9           initial begin
10                  A=0; B=0; C=0; #1
11                  A=0; B=0; C=1; #1
12                  A=0; B=1; C=0; #1
13                  A=0; B=1; C=1; #1
14                  A=1; B=0; C=0; #1
15                  A=1; B=0; C=1; #1
16                  A=1; B=1; C=0; #1
17                  A=1; B=1; C=1;
18          end
19  endmodule
```

Ex.1. Bool's Law



Ex.2. 1-bit Adder

- Ch2 - page 16

$$S = A' B' Cin + A' B Cin' + A B' Cin' + A B Cin$$

$$Cout = A' B Cin + A B' Cin + A B Cin' + A B Cin$$

Ex.2. 1-bit Adder

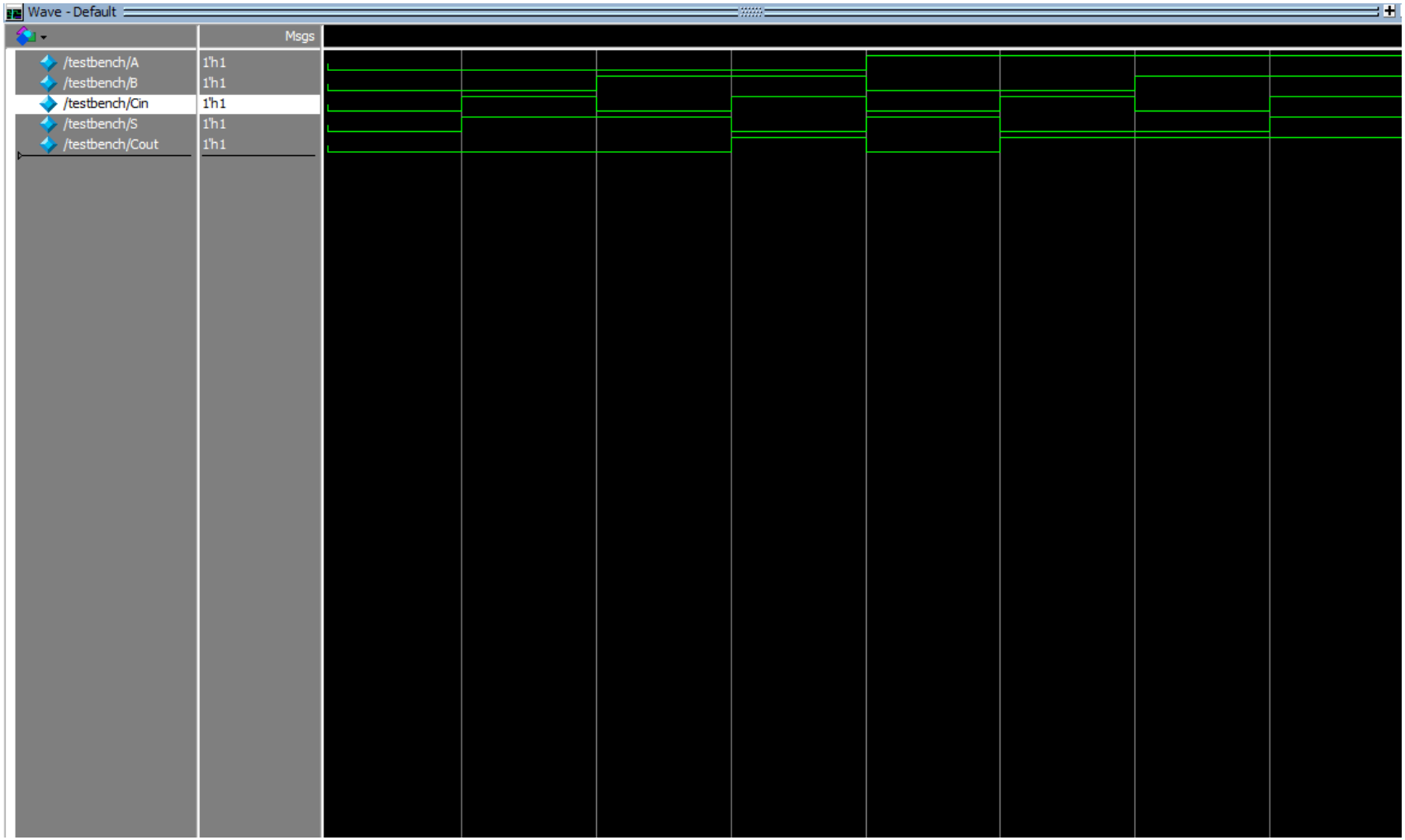
D:/altera/13.0/adder_1bit.v - Default

```
Ln#
1  module adder_1bit(A,B,Cin,S,Cout);
2          input A,B,Cin;
3          output S,Cout;
4
5          assign S = (~A&~B&Cin) | (~A&B&~Cin) | (A&~B&~Cin) | (A&B&Cin);
6          assign Cout = (~A&B&Cin) | (A&~B&Cin) | (A&B&~Cin) | (A&B&Cin);
7  endmodule
```

D:/altera/13.0/testbench.v (/testbench) - Default

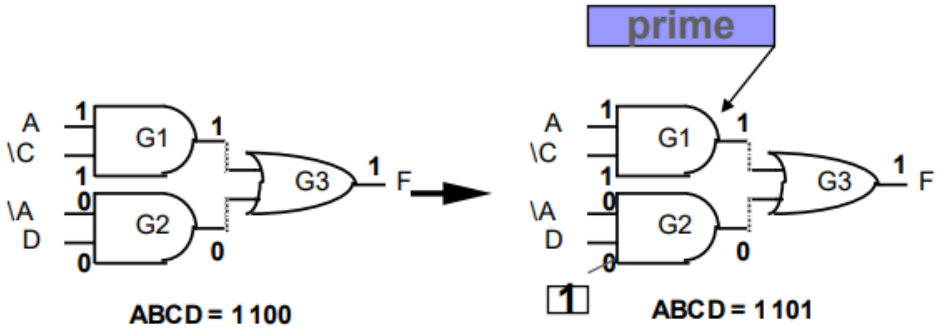
```
Ln#
 1  `timescale 1ns/1ns
 2
 3  module testbench();
 4          reg A,B,Cin;
 5          wire S,Cout;
 6
 7          adder_1bit my_adder(A,B,Cin,S,Cout);
 8
 9          initial begin
10                  A=0; B=0; Cin=0; #1
11                  A=0; B=0; Cin=1; #1
12                  A=0; B=1; Cin=0; #1
13                  A=0; B=1; Cin=1; #1
14                  A=1; B=0; Cin=0; #1
15                  A=1; B=0; Cin=1; #1
16                  A=1; B=1; Cin=0; #1
17                  A=1; B=1; Cin=1;
18          end
19  endmodule
```

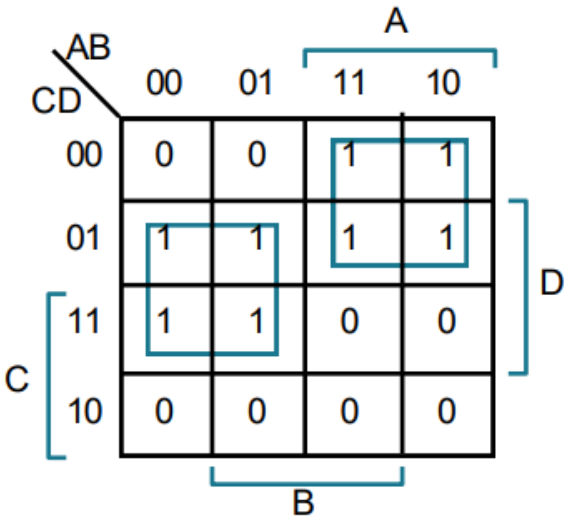
Ex.2. 1-bit Adder

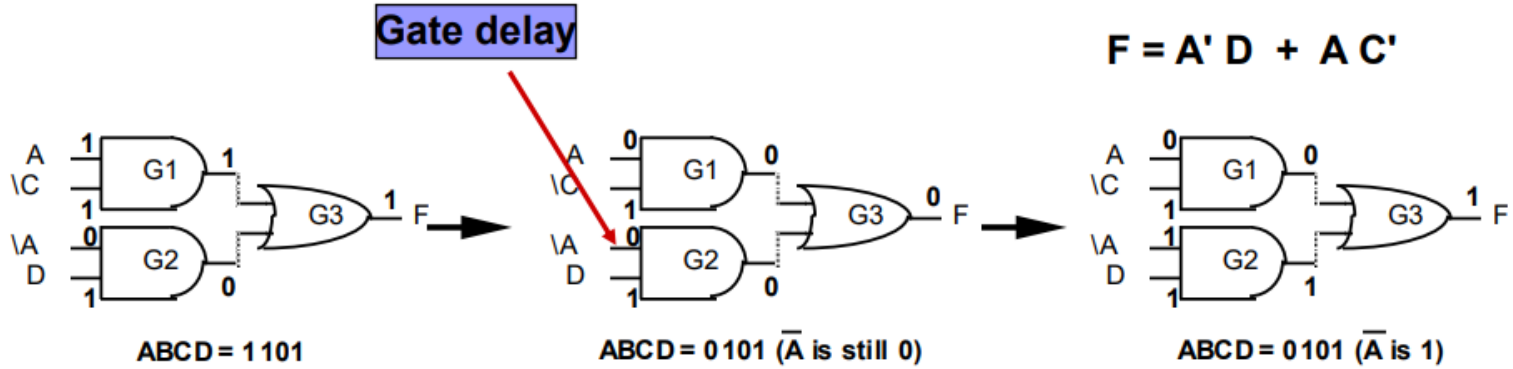


Ex.3. Hazard / Glitches

- Ch3-52







Ex.3. Hazard / Glitches

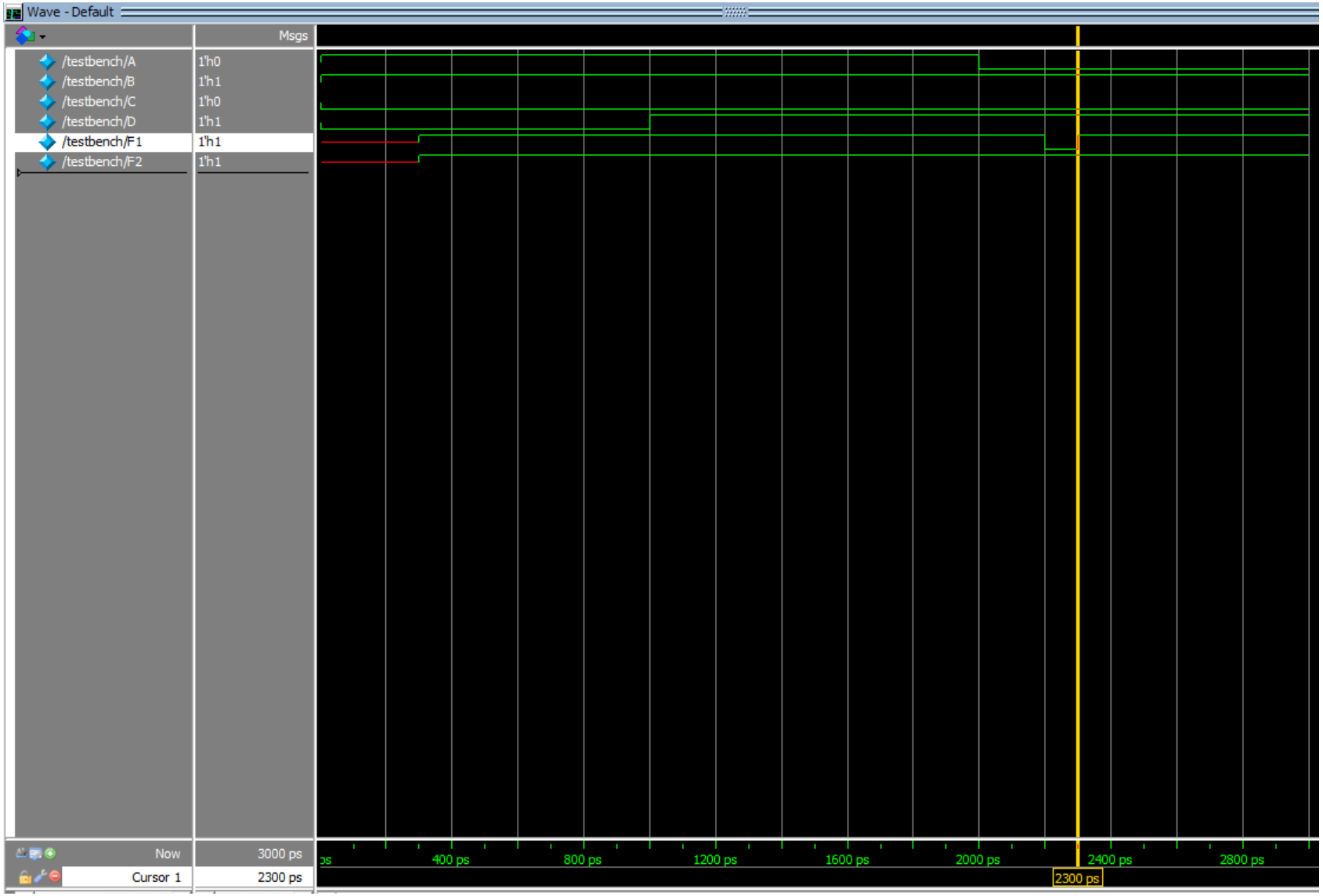
D:/altera/13.0/hazard.v (/testbench/my_hazard) - Default

```
Ln#
 1  `timescale 100ps/100ps
 2
 3  module hazard(A,B,C,D,F1,F2);
 4          input A,B,C,D;
 5          output F1,F2;
 6
 7          // not_gate
 8
 9          // assign F1 = !A&D | A&!C;
10          // assign F2 = !A&D | A&!C | !C&D;
11          wire #1 t1 = !A;
12          wire #1 t2 = t1 & D;
13          wire #1 t3 = !C;
14          wire #1 t4 = A & t3;
15          assign #1 F1 = t2 | t4;
16
17          wire #1 t5 = t3 & D;
18          assign #1 F2 = t2 | t4 | t5;
19
20  endmodule
```

D:/altera/13.0/testbench.v (/testbench) - Default

```
Ln#
 1  `timescale 1ns/1ns
 2
 3  module testbench();
 4          reg A,B,C,D;
 5          wire F1,F2;
 6
 7          hazard my_hazard(A,B,C,D,F1,F2);
 8
 9          initial begin
10                  A=1; B=1; C=0; D=0; #1
11                  A=1; B=1; C=0; D=1; #1
12                  A=0; B=1; C=0; D=1;
13          end
14  endmodule
```

Ex.3. Hazard / Glitches



Q & A

email : dawnshin2000@naver.com