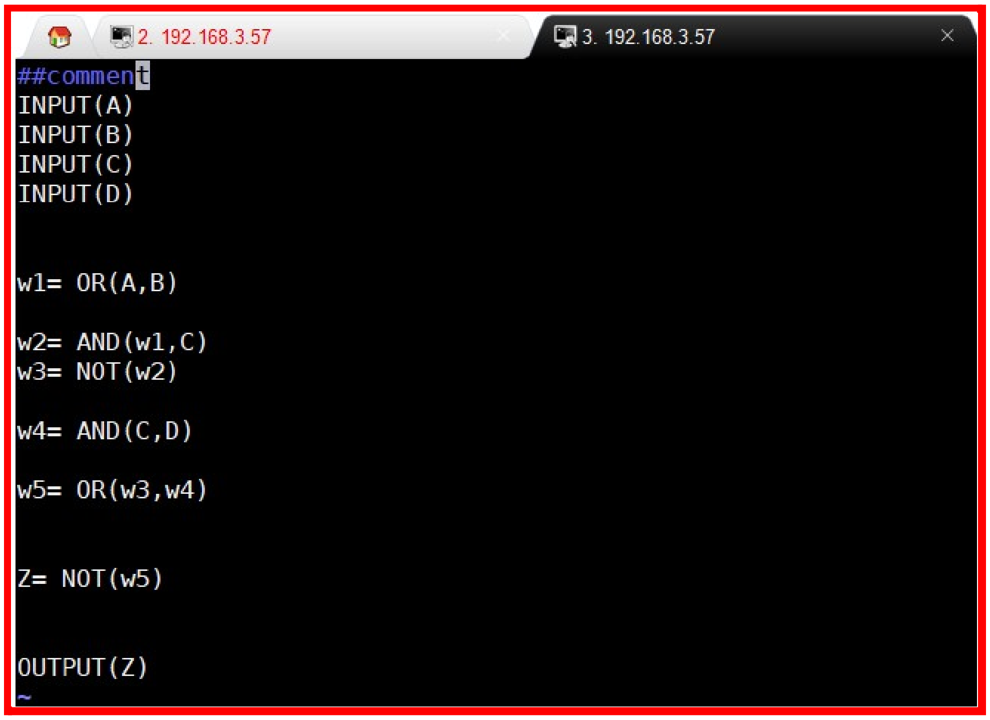
ASSIGNMENT 7

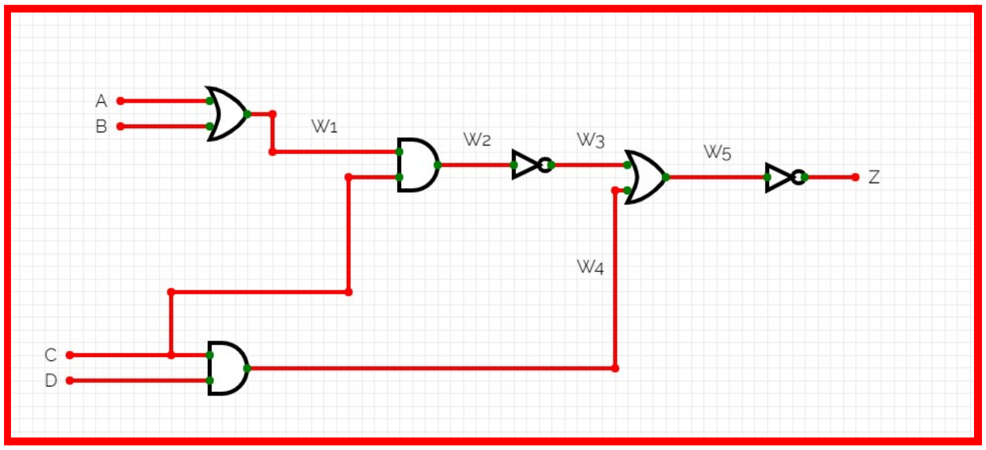
Samaksh Gupta
2019200

Bench File

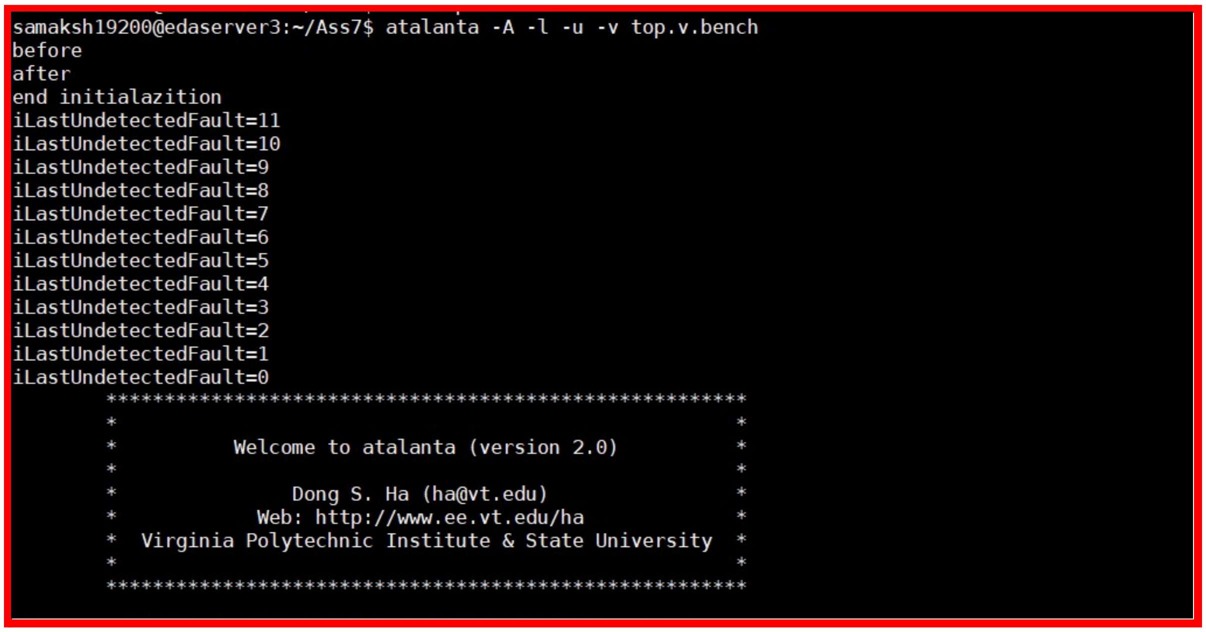


```
##comment
INPUT(A)
INPUT(B)
INPUT(C)
INPUT(D)


w1= OR(A,B)

w2= AND(w1,C)
w3= NOT(w2)

w4= AND(C,D)

w5= OR(w3,w4)


Z= NOT(w5)


OUTPUT(Z)
~
```

Circuit Schematic

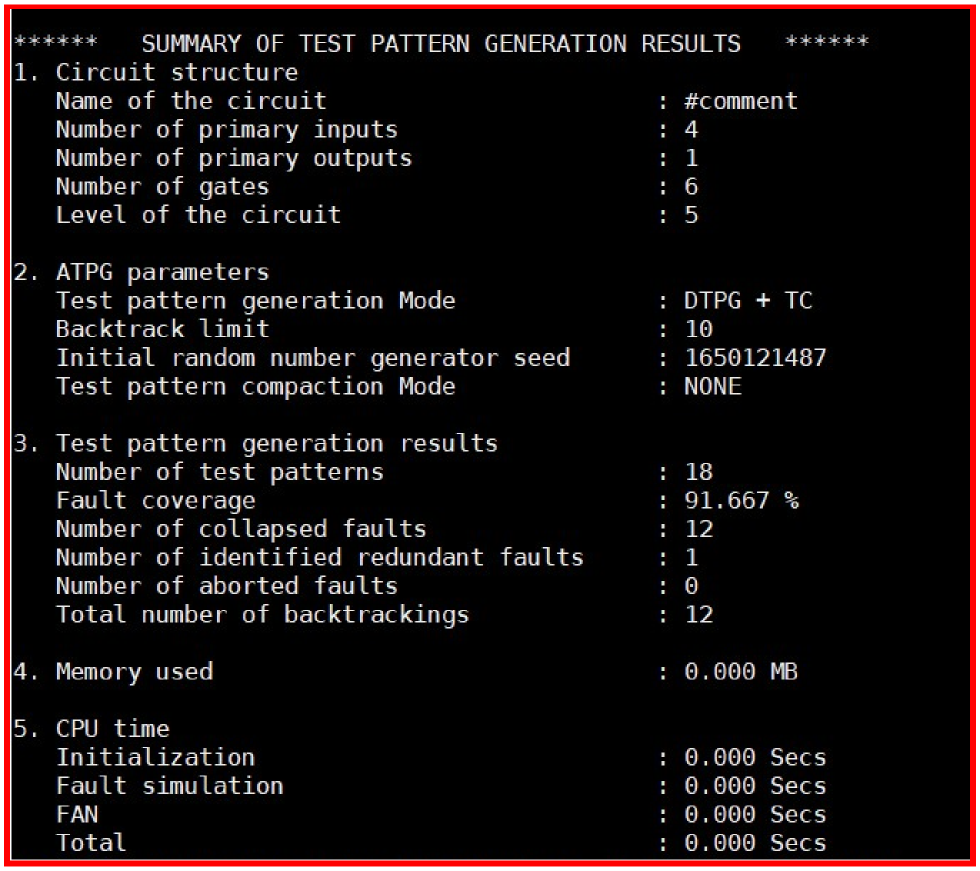


2. Running Atalanta



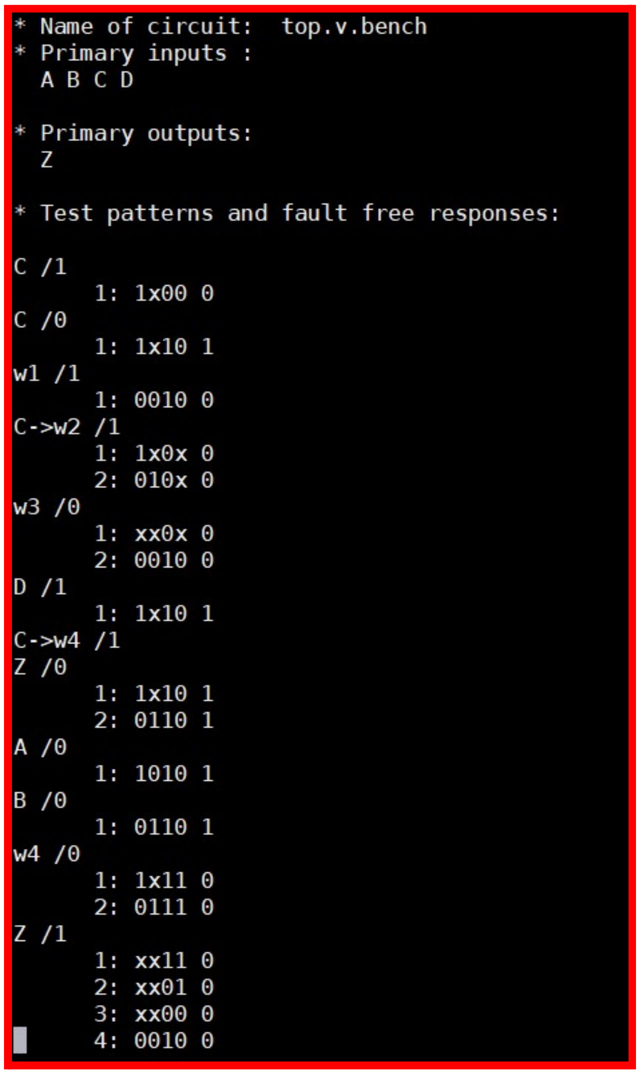
```
samaksh19200@edaserver3:~/Ass7$ atalanta -A -l -u -v top.v.bench
before
after
end initialazition
iLastUndetectedFault=11
iLastUndetectedFault=10
iLastUndetectedFault=9
iLastUndetectedFault=8
iLastUndetectedFault=7
iLastUndetectedFault=6
iLastUndetectedFault=5
iLastUndetectedFault=4
iLastUndetectedFault=3
iLastUndetectedFault=2
iLastUndetectedFault=1
iLastUndetectedFault=0
        *******************************************************
        *                                                     *
        *          Welcome to atalanta (version 2.0)          *
        *                                                     *
        *               Dong S. Ha (ha@vt.edu)                *
        *            Web: http://www.ee.vt.edu/ha             *
        *  Virginia Polytechnic Institute & State University  *
        *                                                     *
        *******************************************************
```

3. Analysis From Atalanta



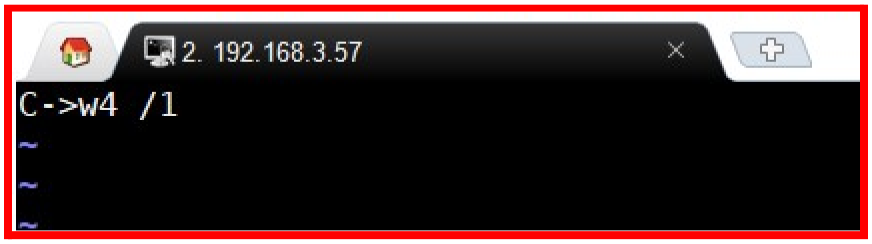
```
******   SUMMARY OF TEST PATTERN GENERATION RESULTS   ******
1. Circuit structure
   Name of the circuit                       : #comment
   Number of primary inputs                  : 4
   Number of primary outputs                 : 1
   Number of gates                           : 6
   Level of the circuit                      : 5

2. ATPG parameters
   Test pattern generation Mode              : DTPG + TC
   Backtrack limit                           : 10
   Initial random number generator seed      : 1650121487
   Test pattern compaction Mode              : NONE

3. Test pattern generation results
   Number of test patterns                   : 18
   Fault coverage                            : 91.667 %
   Number of collapsed faults                : 12
   Number of identified redundant faults     : 1
   Number of aborted faults                  : 0
   Total number of backtrackings             : 12

4. Memory used                               : 0.000 MB

5. CPU time
   Initialization                            : 0.000 Secs
   Fault simulation                          : 0.000 Secs
   FAN                                       : 0.000 Secs
   Total                                     : 0.000 Secs
```

Faults Reported



```
* Name of circuit:  top.v.bench
* Primary inputs :
  A B C D

* Primary outputs:
  Z

* Test patterns and fault free responses:

C /1
      1: 1x00 0
C /0
      1: 1x10 1
w1 /1
      1: 0010 0
C->w2 /1
      1: 1x0x 0
      2: 010x 0
w3 /0
      1: xx0x 0
      2: 0010 0
D /1
      1: 1x10 1
C->w4 /1
Z /0
      1: 1x10 1
      2: 0110 1
A /0
      1: 1010 1
B /0
      1: 0110 1
w4 /0
      1: 1x11 0
      2: 0111 0
Z /1
      1: xx11 0
      2: xx01 0
      3: xx00 0
      4: 0010 0
```

Redundant Fault



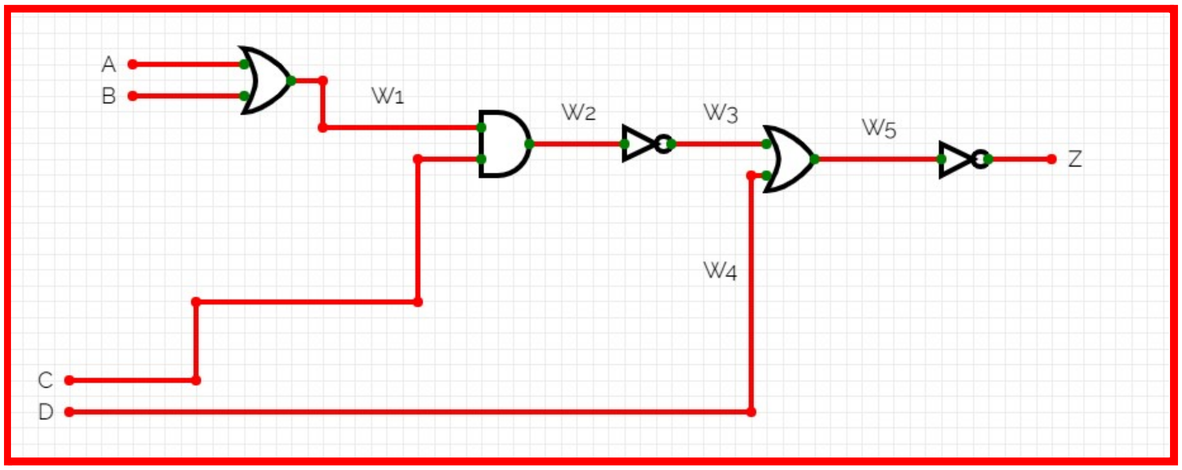
```
C->w4 /1
~
~
~
```

Fault Sites and Type	Test Pattern (Tool)	Test Pattern (Manual)
A/0	1010	1010
B/0	0110	0110
C/0	1x10	1x10
C/1	1x00	1x00
D/1	1x10	1x10
w1/1	0010	0010
w3/0	xx0x 0010	xx0x 0010
w4/0	1x11 0111	1x11 0111
Z/1	xx11 xx01 xx00 0010	xx11 xx01 xx00 0010
C→w2/1	1x0x 010x	1x0x 010x
C→w4/0	1x11 0111	1x11 0111
C→w4/1	Redundant	Redundant

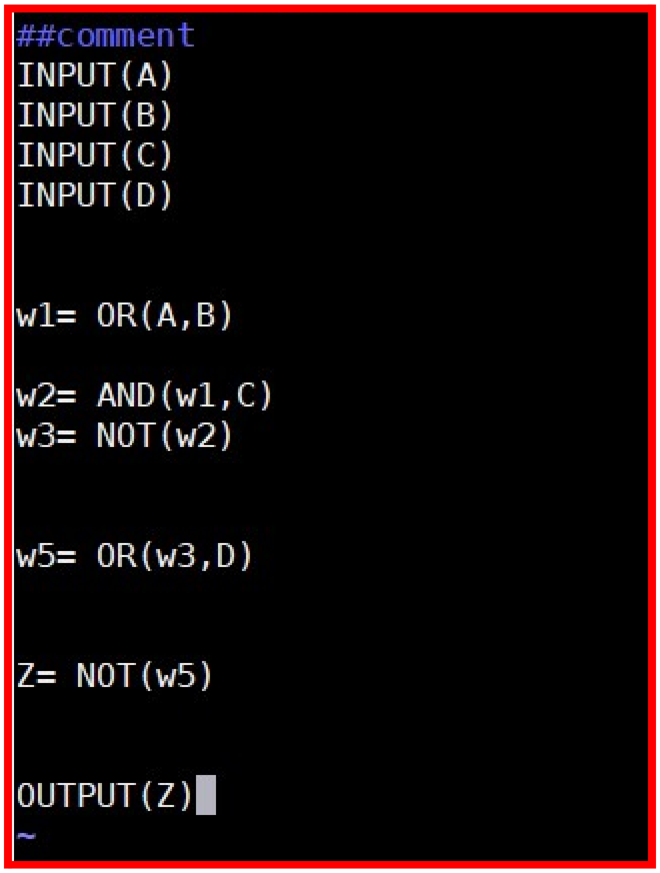
X→ Don't Care : Manual Calculations Match the Tool's.

The redundant fault is on the input of the 'AND' gate. It is a SA/1 fault.
1 is the non-controlling input of the 'AND' gate; as a result, whatever 'D' is, it would be propagated to the input of the 'OR' gate. Hence, if C→W4/1 is a redundant fault, we can remove the 'AND' gate altogether and have 'D' be directly given as the input to the 'OR' gate.

New Schematic

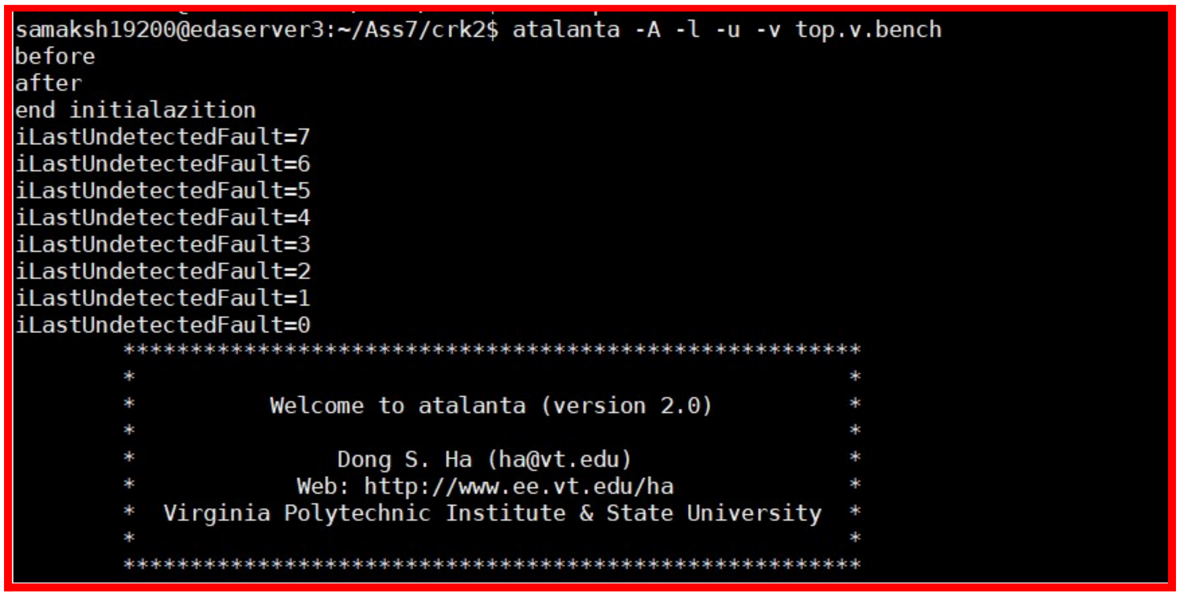


New Bench File

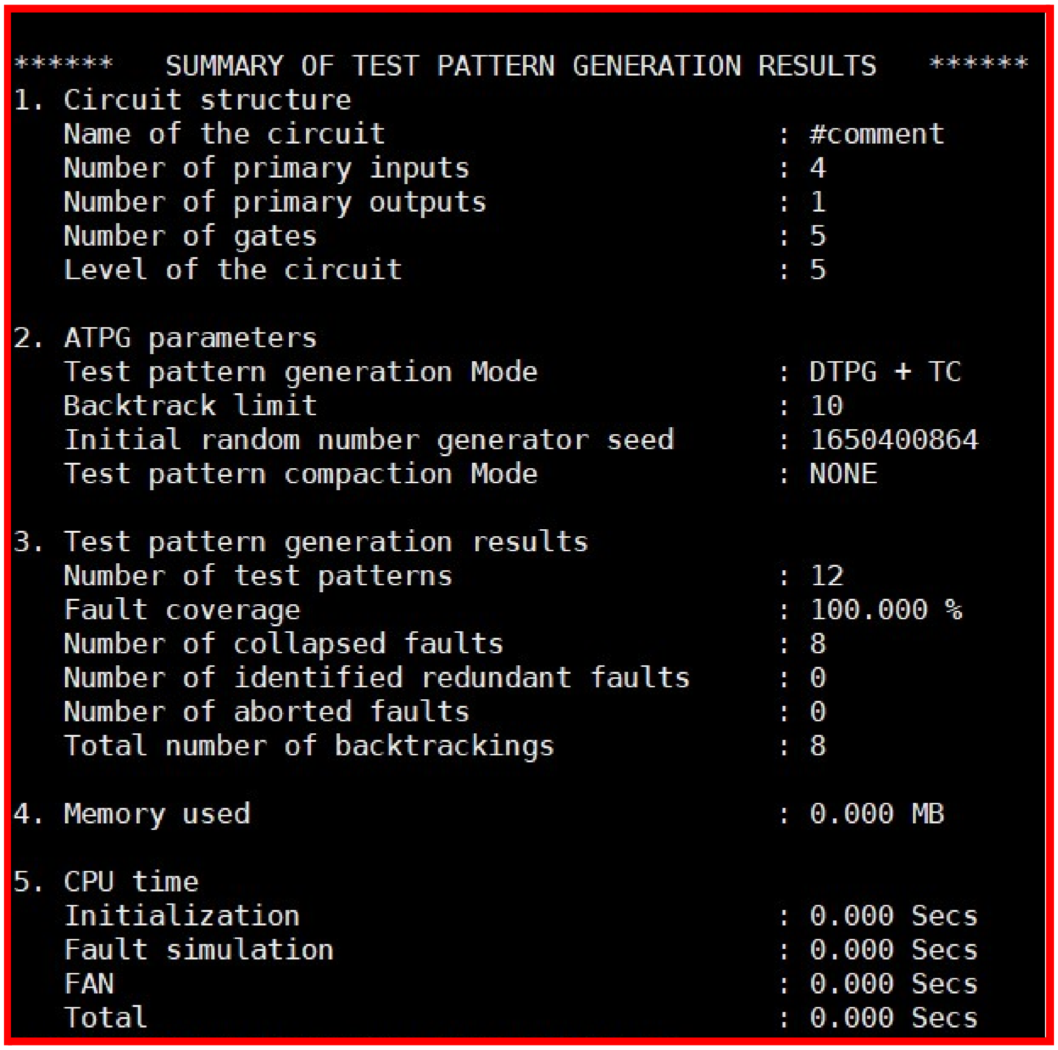


```
##comment
INPUT(A)
INPUT(B)
INPUT(C)
INPUT(D)


w1= OR(A,B)

w2= AND(w1,C)
w3= NOT(w2)


w5= OR(w3,D)


Z= NOT(w5)


OUTPUT(Z)
~
```

New Analysis from Atalanta

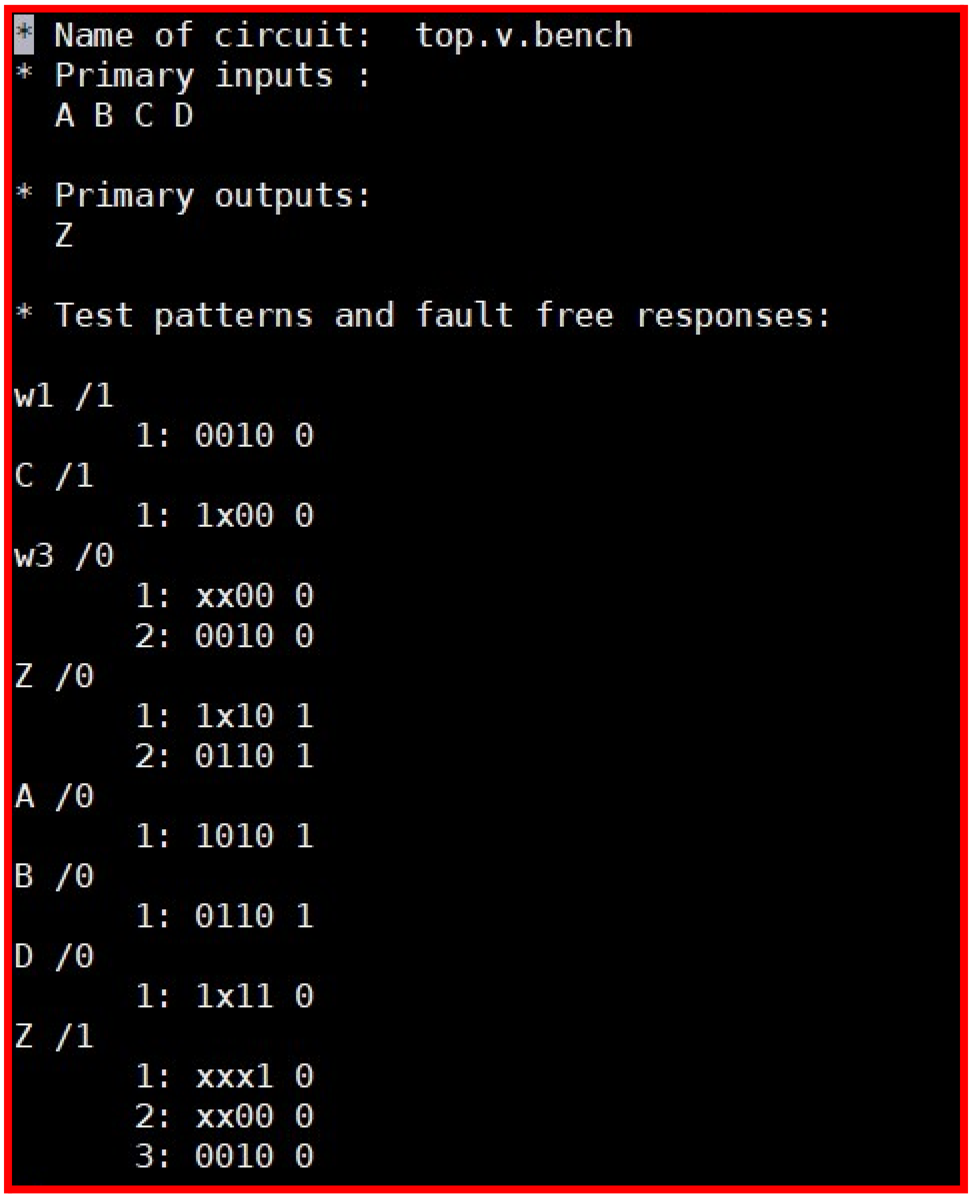


```
samaksh19200@edaserver3:~/Ass7/crk2$ atalanta -A -l -u -v top.v.bench
before
after
end initialazition
iLastUndetectedFault=7
iLastUndetectedFault=6
iLastUndetectedFault=5
iLastUndetectedFault=4
iLastUndetectedFault=3
iLastUndetectedFault=2
iLastUndetectedFault=1
iLastUndetectedFault=0
        *******************************************************
        *                                                     *
        *           Welcome to atalanta (version 2.0)         *
        *                                                     *
        *                Dong S. Ha (ha@vt.edu)               *
        *              Web: http://www.ee.vt.edu/ha           *
        *  Virginia Polytechnic Institute & State University  *
        *                                                     *
        *******************************************************
```



```
******   SUMMARY OF TEST PATTERN GENERATION RESULTS   ******
1. Circuit structure
   Name of the circuit                          : #comment
   Number of primary inputs                     : 4
   Number of primary outputs                    : 1
   Number of gates                              : 5
   Level of the circuit                         : 5

2. ATPG parameters
   Test pattern generation Mode                 : DTPG + TC
   Backtrack limit                              : 10
   Initial random number generator seed         : 1650400864
   Test pattern compaction Mode                 : NONE

3. Test pattern generation results
   Number of test patterns                      : 12
   Fault coverage                               : 100.000 %
   Number of collapsed faults                   : 8
   Number of identified redundant faults        : 0
   Number of aborted faults                     : 0
   Total number of backtrackings                : 8

4. Memory used                                  : 0.000 MB

5. CPU time
   Initialization                               : 0.000 Secs
   Fault simulation                             : 0.000 Secs
   FAN                                          : 0.000 Secs
   Total                                        : 0.000 Secs
```

New Faults Reported



```
* Name of circuit:  top.v.bench
* Primary inputs :
  A B C D

* Primary outputs:
  Z

* Test patterns and fault free responses:

w1 /1
      1: 0010 0
C /1
      1: 1x00 0
w3 /0
      1: xx00 0
      2: 0010 0
Z /0
      1: 1x10 1
      2: 0110 1
A /0
      1: 1010 1
B /0
      1: 0110 1
D /0
      1: 1x11 0
Z /1
      1: xxx1 0
      2: xx00 0
      3: 0010 0
```

No Redundant Faults Anymore

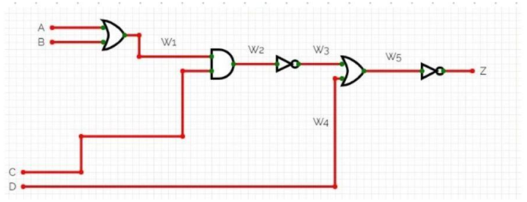
The number of fault sites reduced too since we have removed a whole gate and hence possible fault sites in itself.

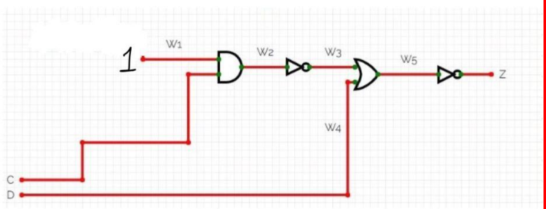
Fault Sites and Type	Test Pattern (Tool)	Test Pattern (Manual)
A/0	1010	1010
B/0	0110	0110
C/1	1x00	1x00
D/0	1x11	1x11
w1/1	0010	0010
w3/0	xx00 0010	xx00 0010
Z/0	1x10 0110	1x10 0110
Z/1	xxx1 xx00 0010	xxx1 xx00 0010

X→ Don't Care : Manual Calculations Match the Tool's.

Example of calculating Test Pattern for w1/1

Proof of Test Pattern '0010' for $SA \rightarrow W_1 / 1$





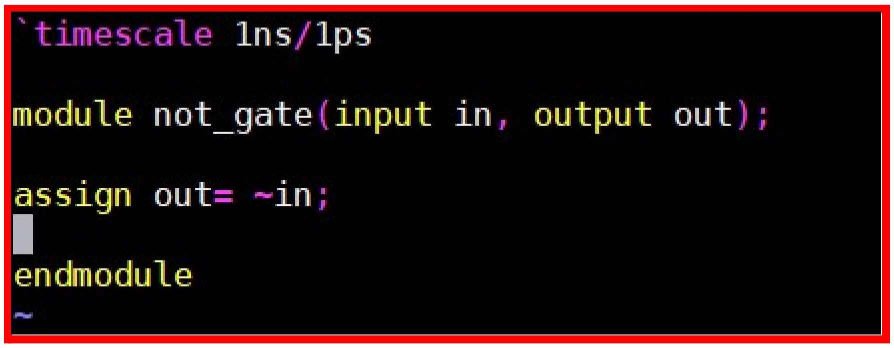
* we want 'w_1' to be zero ⇒ A = 0 , B = 0
* we want 'C' to be non Controlling ⇒ C = 1
* ⇒ $w_2 = 0$ ⇒ $w_3 = 1$
* we want 'D' to be non Controlling ⇒ D = 0
* ⇒ $W_5 = 1$
* ⇒ Z = 0

Different outputs

* w_1 is stuck at 1, even when A = 0, B = 0
* C = 1 (Non Controlling)
* ⇒ $W_2 = 1$
* ⇒ $W_3 = 0$
* D = 0 (Non Controlling)
* ⇒ $W_5 = 0$
* ⇒ Z = 1

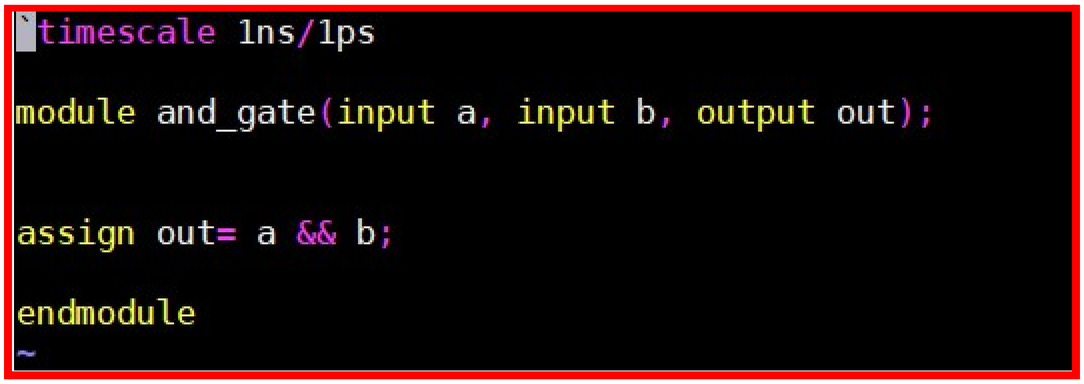
PART 2

Not Gate



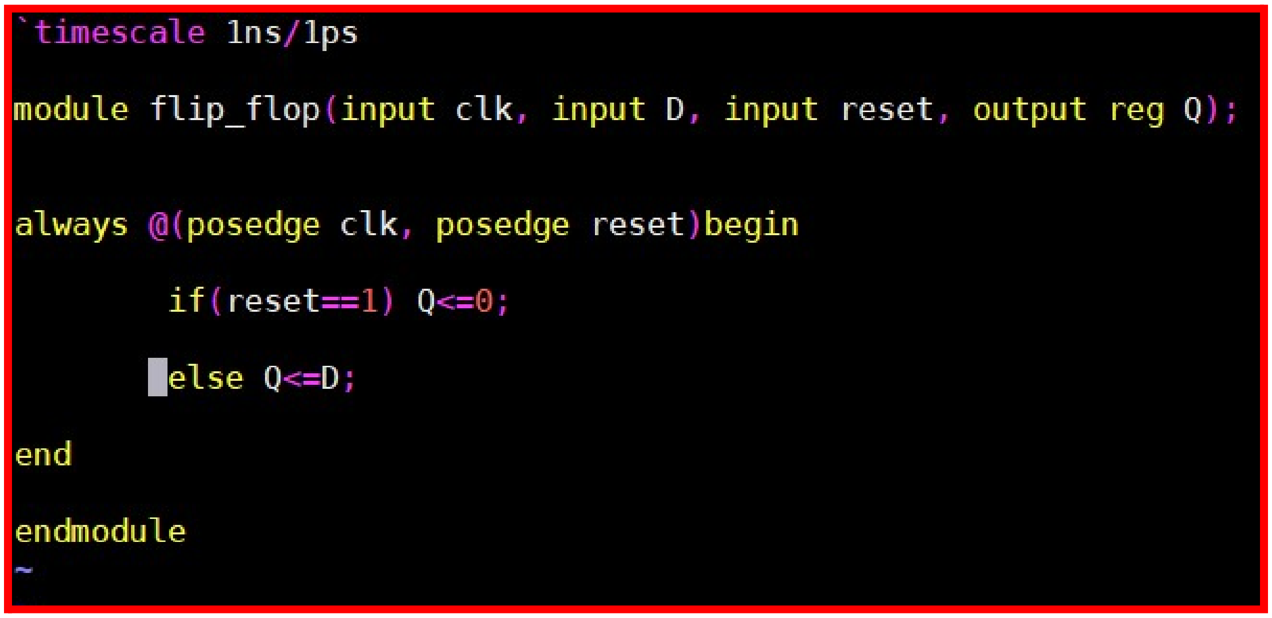
```
`timescale 1ns/1ps

module not_gate(input in, output out);

assign out= ~in;

endmodule
~
```

And Gate



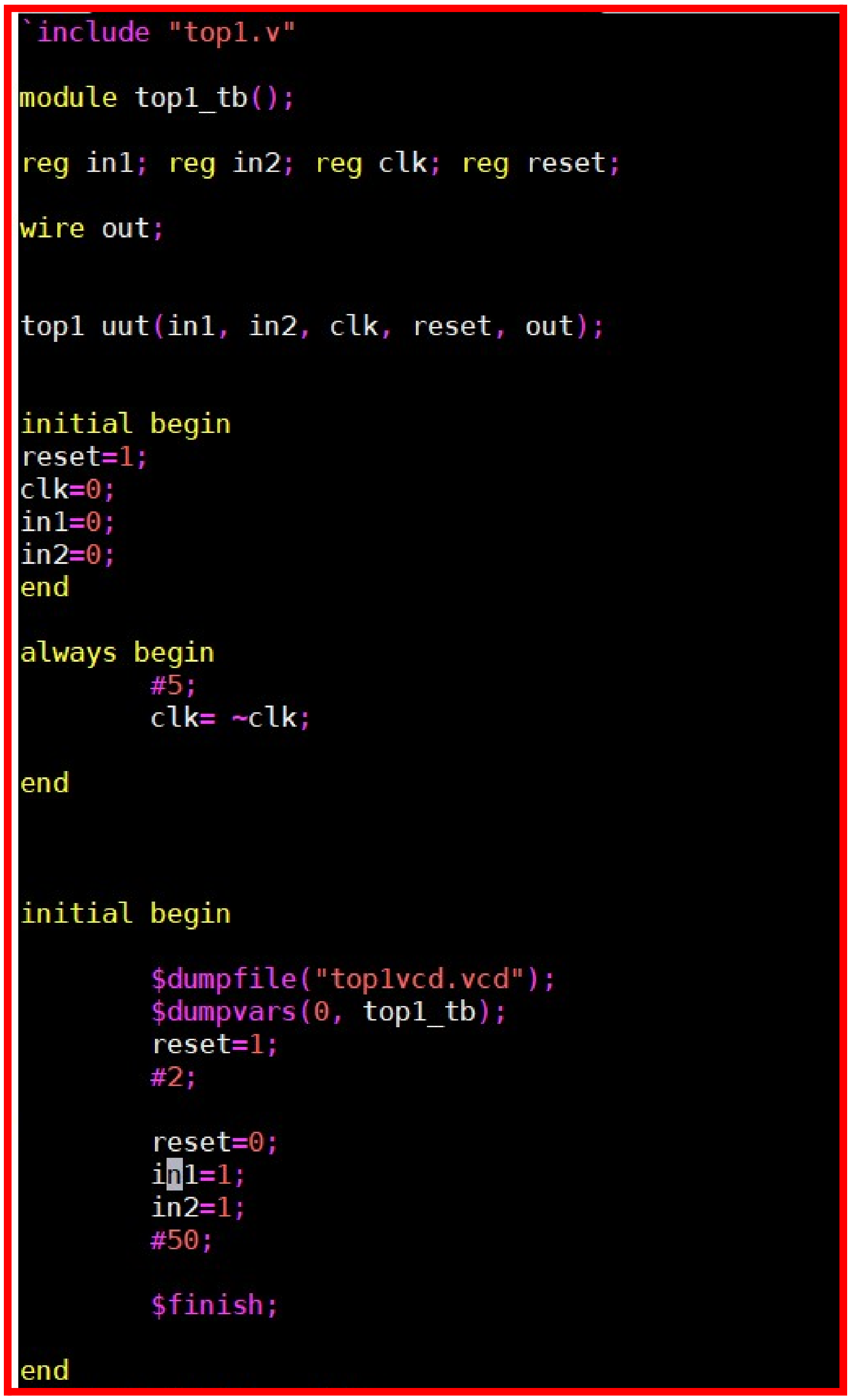
```
`timescale 1ns/1ps

module and_gate(input a, input b, output out);


assign out= a && b;

endmodule
~
```

Flip Flop



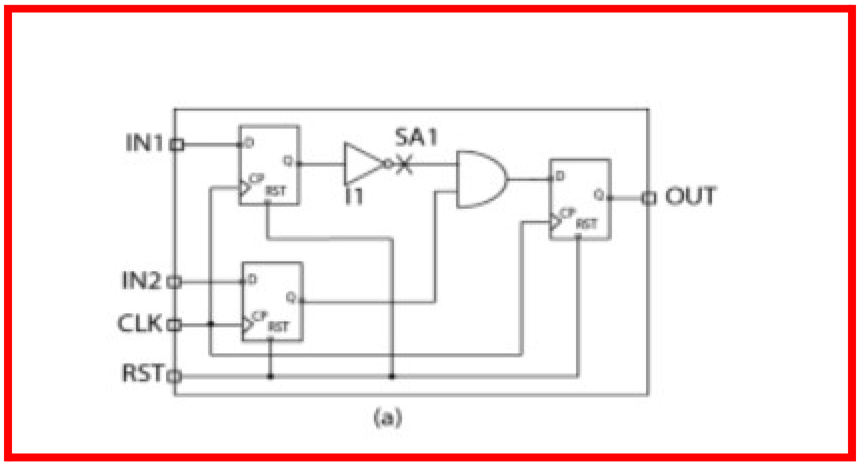
```
`timescale 1ns/1ps

module flip_flop(input clk, input D, input reset, output reg Q);


always @(posedge clk, posedge reset)begin

        if(reset==1) Q<=0;

       else Q<=D;

end

endmodule
~
```

Test Bench

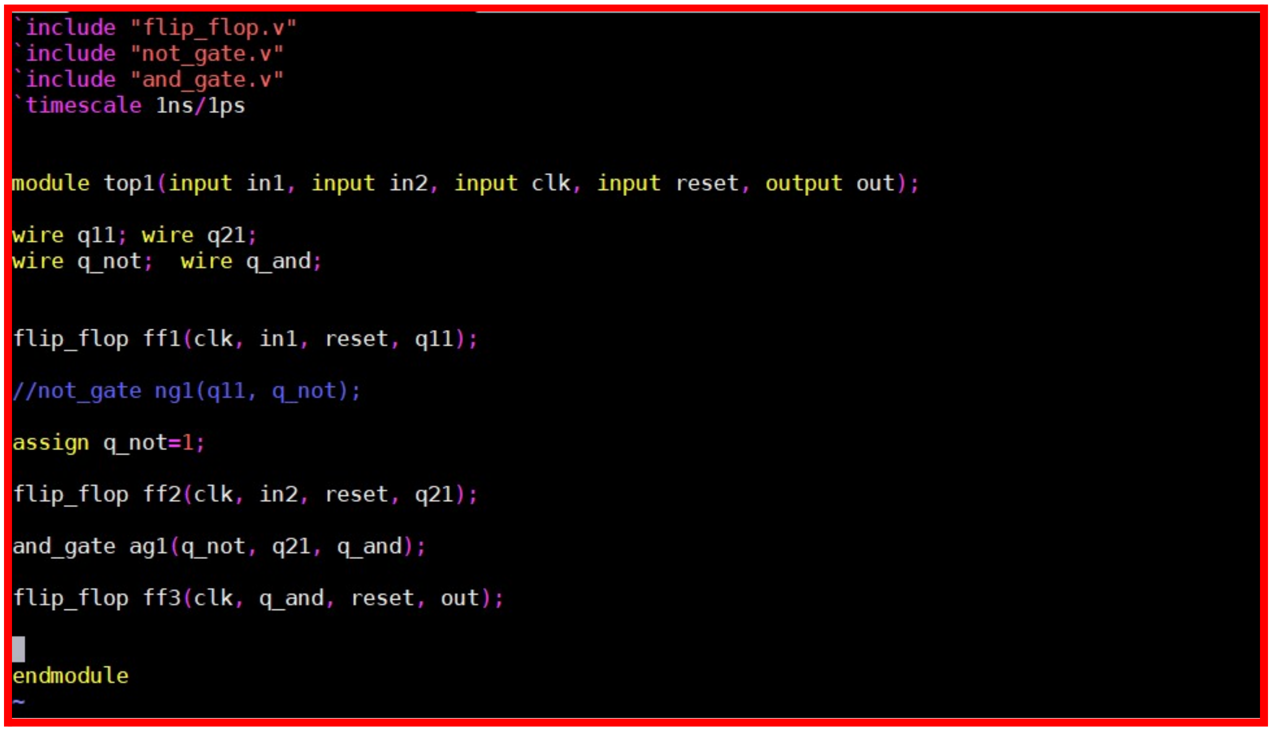


```
`include "top1.v"

module top1_tb();

reg in1; reg in2; reg clk; reg reset;

wire out;


top1 uut(in1, in2, clk, reset, out);


initial begin
reset=1;
clk=0;
in1=0;
in2=0;
end

always begin
        #5;
        clk= ~clk;

end



initial begin

        $dumpfile("top1vcd.vcd");
        $dumpvars(0, top1_tb);
        reset=1;
        #2;

        reset=0;
        in1=1;
        in2=1;
        #50;

        $finish;

end
```

Part A



Top Module

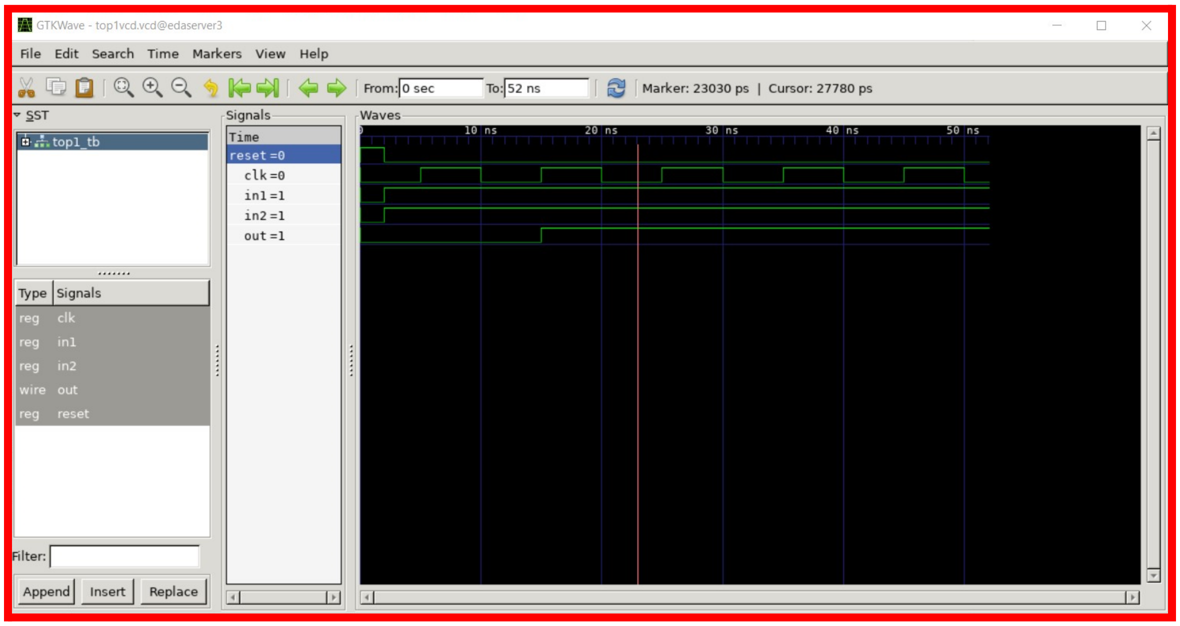


```
`include "flip_flop.v"
`include "not_gate.v"
`include "and_gate.v"
`timescale 1ns/1ps


module top1(input in1, input in2, input clk, input reset, output out);

wire q11; wire q21;
wire q_not;  wire q_and;


flip_flop ff1(clk, in1, reset, q11);

//not_gate ng1(q11, q_not);

assign q_not=1;

flip_flop ff2(clk, in2, reset, q21);

and_gate ag1(q_not, q21, q_and);

flip_flop ff3(clk, q_and, reset, out);


endmodule
~
```

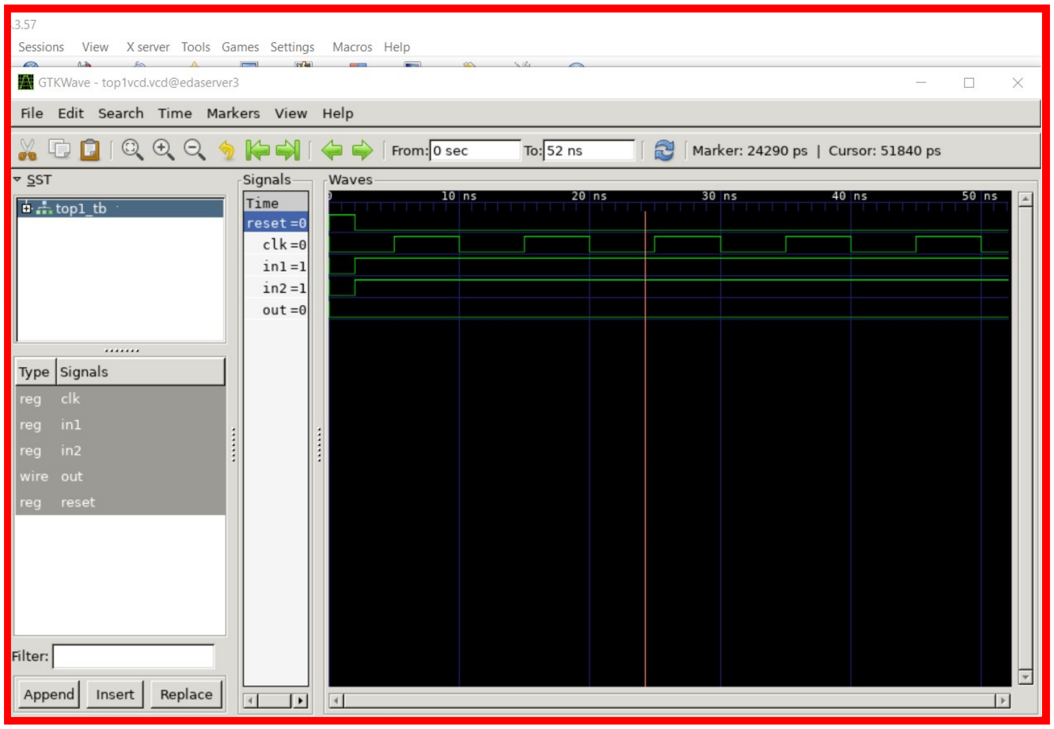
We will measure the output for the two conditions: -
i) Stuck at 1 (q_not=1).
ii) q_not driven by the not gate according to the input provided.

We generate VCD file twice: -
i) For the top module above.
ii) Uncommenting the commented line and commenting the ‘assign q_not=1’ line.

Waveform for Stuck at 1 Fault

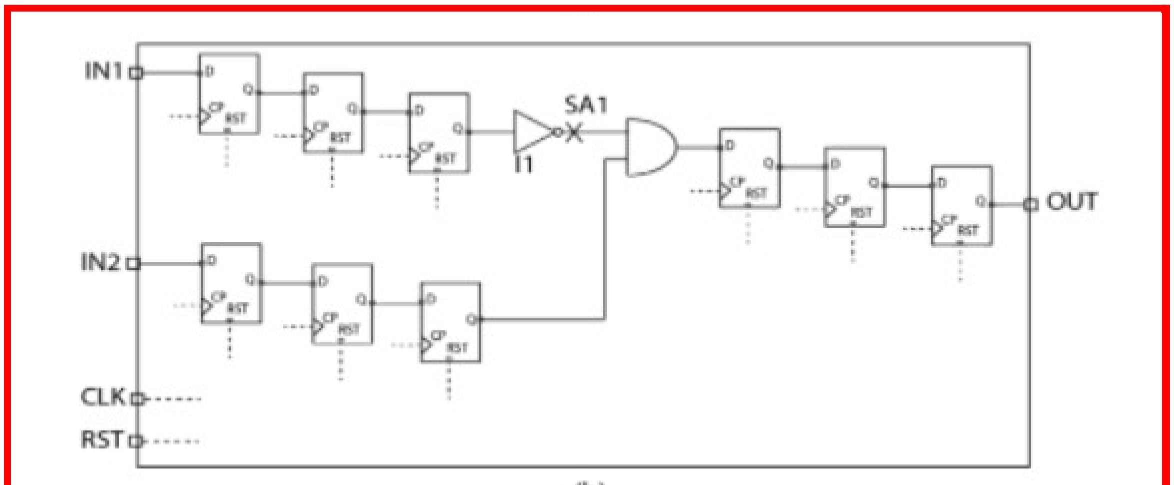


Waveform Without the Stuck at 1 Fault



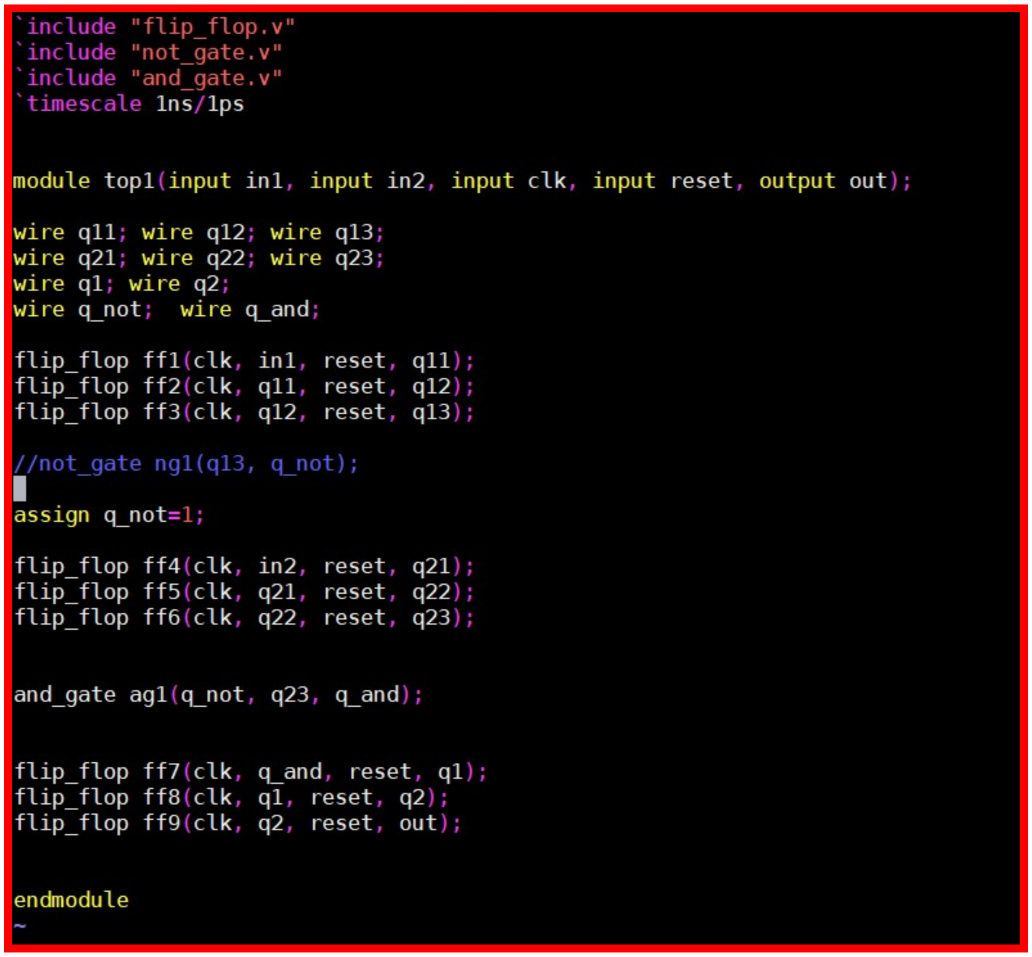
As we can see, when the second positive edge of the clock arrives, the out incase of SA(1) goes to 1, but the ‘out’ in case of no-fault stays 0. Hence, we have detected the error.

Part B



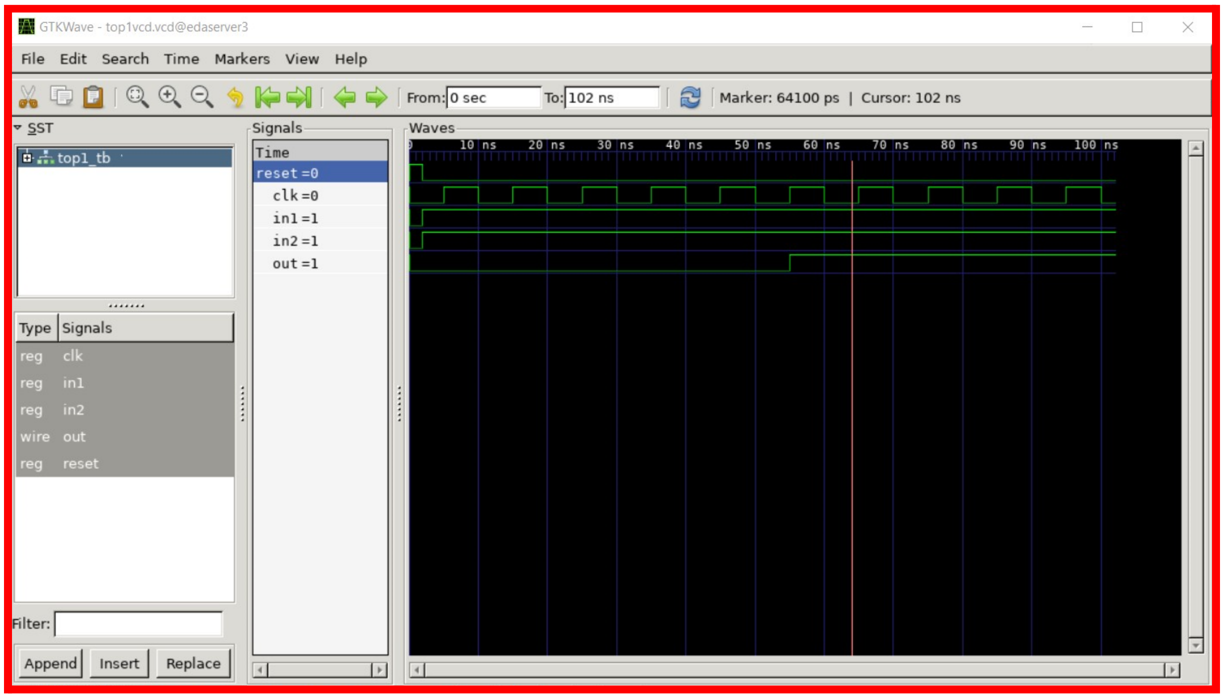
(b)

Top Module

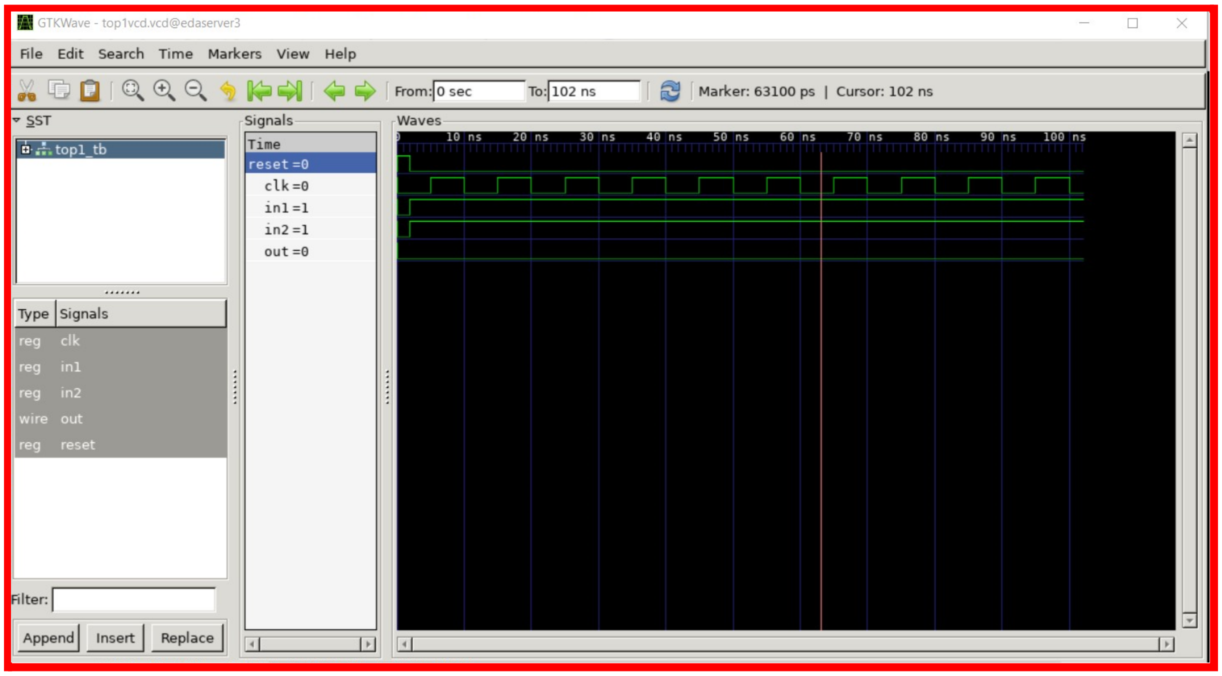


```
`include "flip_flop.v"
`include "not_gate.v"
`include "and_gate.v"
`timescale 1ns/1ps


module top1(input in1, input in2, input clk, input reset, output out);

wire q11; wire q12; wire q13;
wire q21; wire q22; wire q23;
wire q1; wire q2;
wire q_not;  wire q_and;

flip_flop ff1(clk, in1, reset, q11);
flip_flop ff2(clk, q11, reset, q12);
flip_flop ff3(clk, q12, reset, q13);

//not_gate ng1(q13, q_not);

assign q_not=1;

flip_flop ff4(clk, in2, reset, q21);
flip_flop ff5(clk, q21, reset, q22);
flip_flop ff6(clk, q22, reset, q23);


and_gate ag1(q_not, q23, q_and);


flip_flop ff7(clk, q_and, reset, q1);
flip_flop ff8(clk, q1, reset, q2);
flip_flop ff9(clk, q2, reset, out);


endmodule
~
```

Waveform for Stuck at 1 Fault



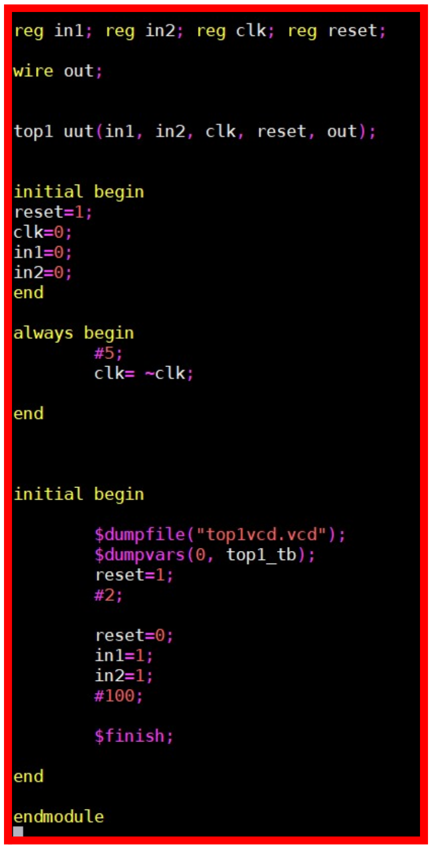
Waveform Without the Stuck at 1 Fault



We can see that ‘out’ is different for the two cases (55ns). When the 6th clock edge comes after the reset is de-asserted, the output for in1=1 and in2=1 is available.

<u>Part C</u>

In the case of ‘part b’, we have more flip flops and thus, the number of clock cycles required to get the output of the input for the first time has increased. As you can see, in the test bench, I had waited for 50ns after de-asserting the reset signal with Tclk=10ns. But this waiting period was not enough as the output signal flipped after 50ns, so I had to go back to the test bench and increase the waiting period before finishing the simulation window.



```
reg in1; reg in2; reg clk; reg reset;

wire out;


top1 uut(in1, in2, clk, reset, out);


initial begin
reset=1;
clk=0;
in1=0;
in2=0;
end

always begin
        #5;
        clk= ~clk;

end



initial begin

        $dumpfile("top1vcd.vcd");
        $dumpvars(0, top1_tb);
        reset=1;
        #2;

        reset=0;
        in1=1;
        in2=1;
        #100;

        $finish;

end

endmodule
```

Had I not counted the delay due to flip flops, I might have concluded that this is a redundant fault as ‘out’ in both cases (SA/1 and no SA) comes out to be the same. But the fact is due to 6 flip flops, the output of the inputs would be available after 6 clock cycles from the reset being de-asserted.

As we increase the flip flops in the path, the delay to get the output would further increase and hence the testing time would increase, resulting in a longer time to market.

Also, since there are more elements, it is more laborious to make the netlist.
—--