DIGITAL SYSTEMS TESTING

Design and Implementation of Logic Simulator, Deductive Fault Simulator and PODEM (Part 2)

Name : Pranjali Borkar
GT ID : 903181565

LOGIC SIMULATOR

Structure Description

After reading the textfile, we store the gate type in a node named 'type', input 1 and 2 nets in arrays' ip1' and 'ip2' and output nets in 'op'.We use the Networkx library in Python for this.We basically create the gates as nodes and attach the necessary attributes to it.

Now we have it1,it2 and ot as flags that denote whether a gate has been evaluated or not.So, if the 'ot' of any gate is 0 and 'it1' and 'it2' are 1 then the gate has not been evaluated. And hence we can go ahead and perform the calculation and store the output value. Thus, the primary gates are first calculated.

Next we move on to the next level of gates that are driven by the outputs of the primary gates.In this way finally all the outputs are calculated.

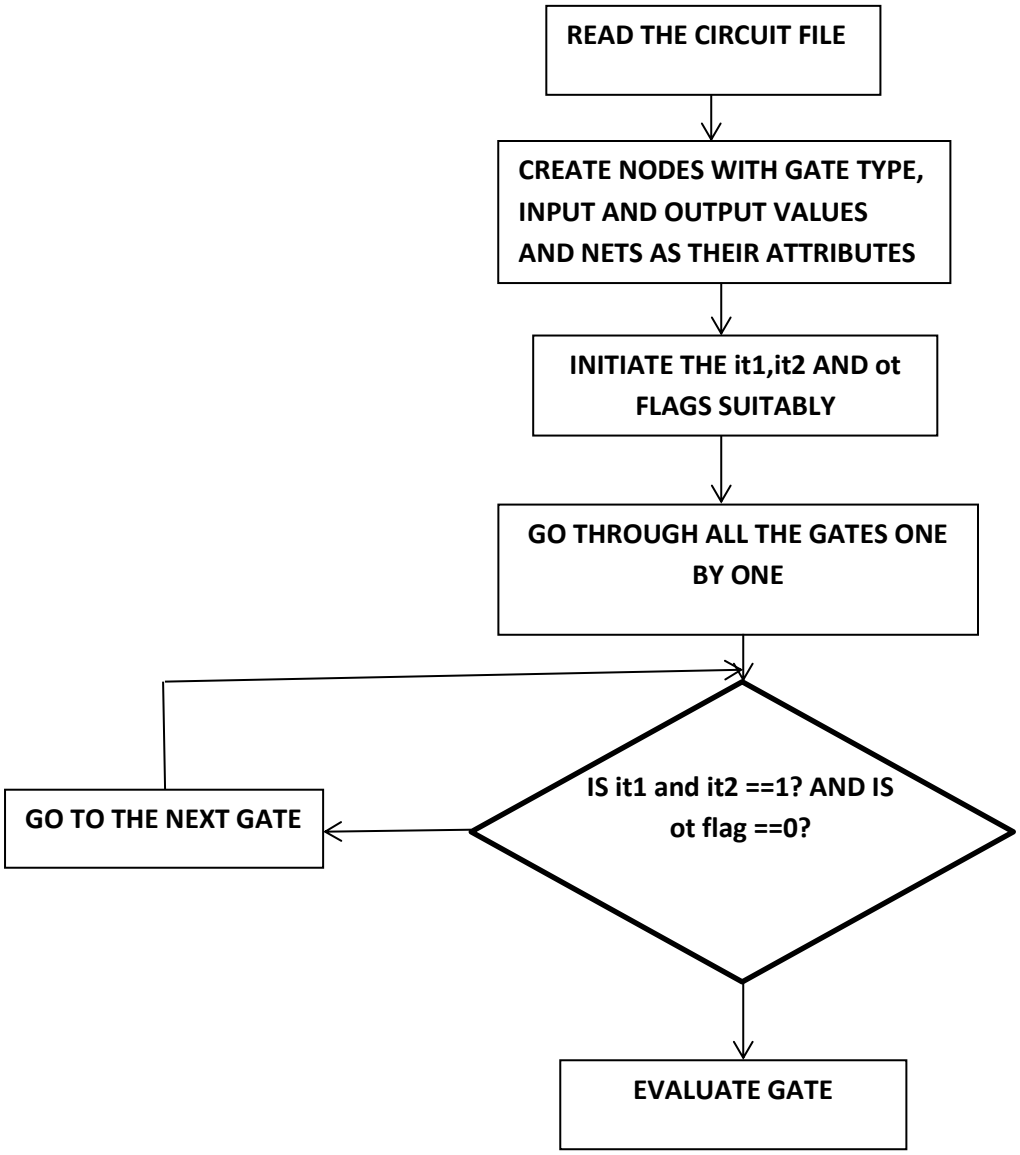
Here, we have a five valued logic simulator that has1,0,x,D,Dbar as the logic values.

1. gate : Each gate has a number or index by which it is referred.
2. type : A string input which holds the type of the gate. (INPUT and OUTPUT are not stored as gates)
3. ip1, ip2 & op: There are two inputs and 1 output for each gate. (For NOT and BUF gates, ip2 is not considered.)
4.IP1,IP2,OP-Values at ip1, ip2 & op respectively.
5. flag: The flag ot is set if that particular gate has been evaluated. This will prevent repeated evaluation of the gate

RESULTS:

Circuit Name	Input	Output
s27.chat	1110101	1 0 0 1
	0001010	0 1 0 0
	1010101	1 0 0 1
	0110111	0 0 0 1
	1010001	1 0 0 1
s298f_2.chat	10101010101010101	0 0 0 0 0 0 1 0 1 0 1 0 0 0 1 1 1 0 0 0
	01011110000000111	0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0
	11111000001111000	0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0
	11100001110001100	0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 1
	01111011110000000	1 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 1 1 0 1
s344f_2.chat	10101010101010111111111	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1
	01011110000000111000000	0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0
	11111000001111000111111	0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 1 0 1 0
	11100001110001100000000	0 0 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 1 0
	01111011110000000111111	1 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0
s349f_2.chat	10101010101010111111111	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1
	01011110000000111000000	0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 0 0 0 0
	11111000001111000111111	0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 1 1 1 1 0 0
	11100001110001100000000	0 0 0 0 1 1 0 1 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 1
	01111011110000000111111	1 0 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0

FLOW CHART:


READ THE CIRCUIT FILE
CREATE NODES WITH GATE TYPE, INPUT AND OUTPUT VALUES AND NETS AS THEIR ATTRIBUTES
INITIATE THE it1,it2 AND ot FLAGS SUITABLY
GO THROUGH ALL THE GATES ONE BY ONE
IS it1 and it2 ==1? AND IS ot flag ==0?
GO TO THE NEXT GATE
EVALUATE GATE

FAULT SIMULATOR

Implementation

In the first part of the project, we did logic simulation.Now, in this part we do deductive fault simulation for the same. Here, the things we consider are the logic values and the fault lists that have to propagated through the circuit and has to be observed at the output nets.

We develop rules for each gate and each logic combination for deductive fault simulation

We also add the local faults to the fault lists at every net.

INVERTER

The same fault list has to be carried forward from the input net to the output net .

BUFFER

The same fault list has to be carried forward from the input net to the output net .

AND

For inputs 0 and 0

Intersection of the input fault lists

For input 0 and 1 or 1 and 0

Depending on the inputs,

Subtraction of fault lists

For inputs 1 and 1

Union of the fault lists

OR

For inputs 0 and 0

Union of the fault lists

For input 0 and 1 or 1 and 0

Depending on the inputs,

Subtraction of fault lists

For inputs 1 and 1

Intersection of the input fault lists

NAND

For inputs 0 and 0

Intersection of the input fault lists

For input 0 and 1 or 1 and 0

Depending on the inputs,

Subtraction of fault lists

For inputs 1 and 1

Union of the fault lists

NOR

For inputs 0 and 0

Union of the fault lists

For input 0 and 1 or 1 and 0

Depending on the inputs,

Subtraction of fault lists

For inputs 1 and 1

Intersection of the input fault lists

Now after every fault list calculation is done, we add the local fault to the output net and also assign the fault list to the input net that the current output net is assigned to.

To calculate the fault coverage -

1. Random vectors are generated and the faults that are generated are stored into a vector (or array).
2. Multiple random vectors are given as input and the above step is repeated.
3. The duplicate entries are removed and the fault coverage is calculated by dividing the total number of faults identified by the random input vectors with the total number of faults in the error list.

Results:

PART A

s27

5 [('9', 1)], ('5', 0)]
7 [('1', 0)], ('12', 0)], [[('9', 1)], ('13', 0)], [('7', 0)]]
9 [('9', 1)]
11 ('3', 0), [('11', 1)]]

s298_f

18 [('18', 1)]
19 [('15', 0), [('39', 1)], [('19', 1)]]
20 [[('15', 0), [('41', 1)]], [('20', 1)]]
21 [('21', 1)]
22 [[('67', 1)], [('15', 0), [('68', 1)]], [('173', 0)]], ('45', 1)], [('22', 1)]]
23 [[('23', 1)]]
24 [[('7', 0)], ('108', 1)], [('107', 1)]], [('5', 0), [('109', 1)]], [('48', 0)]], [('95', 0)], [[('6', 1)], ('102', 0)], [('66', 0)]], [('64', 1)]], [('3', 0), [('110', 1)]], [('49', 0)]], [('24', 0)]]
25 [[('2', 1), [('159', 0)]], [('158', 0)]], [('161', 0)], [('167', 1)]], [('166', 0)]], ('51', 1)], [('25', 1)]]
26 [('9', 0)], ('186', 1)], [('115', 1)]], [('52', 1)]], [('2', 1), [('133', 0)]], [('132', 0)]], [('135', 0)], [('169', 1)]], [('95', 0)],
[[('6', 1)], ('102', 0)], [('66', 0)]], [('64', 1)]], ('182', 1)], [[('138', 0)], ('183', 1)], [('170', 1)]], [('53', 1)]], [('26', 0)]]
27 [[('10', 1), [('103', 1)]], [[('3', 0), [('106', 1)]], [('105', 1)]], [('116', 0)]], [('4', 1), [('95', 0)], [[('6', 1)], ('102', 0)],
[('66', 0)]], [('64', 1)]], [('117', 0)]], [('54', 0)]], [('27', 1)]]
28 [[('4', 1), [('141', 0)]], [[('3', 0)], ('144', 1)], [('143', 0)]], [('119', 1)]], [('5', 0), [('146', 0)], [('120', 1)]], [('118', 1)]],
[('56', 1)]], [('95', 0)], [[('6', 1)], ('102', 0)], [('66', 0)]], [('64', 1)]], [('57', 1)]], [('28', 0)]]
29 [[('5', 0), [[('12', 1)], ('122', 0)], [('58', 0)]], [('29', 1)]]
30 [[('30', 1)]]
31 [[('31', 1)]]
32 [[('9', 0)], ('32', 0)]
33 [[('11', 0)], ('33', 0)]
34 [[('7', 0)], ('34', 0)]
35 [[('10', 1)], ('35', 1)]
36 [[('12', 1)], ('36', 1)]
37 [[('8', 1)], ('37', 1)]

s344f_2

25 [('16', 1), [('1', 0)], ('177', 0)], [('176', 0)]], [('1', 0)], ('182', 1)], [('181', 0)]], [('51', 1)]], [('25', 0)]]
26 [[('184', 0)]], [('52', 0)]], [('26', 1)]]
27 [('1', 0)], ('77', 1)], [[('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)], [('76', 1)]], [('53', 0)]], [[('16', 1)], ('54', 0)], [('27', 0)]]
28 [('16', 1), [[('4', 1), [('1', 0)], ('77', 1)], [[[[('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)],
[('76', 1)]], [('92', 1)]], [('139', 0)]], [('55', 1)]], [('28', 1)]]
29 [('16', 1), [('29', 0)]]
30 [('16', 1), [[('6', 1), [[('1', 0)], ('77', 1)], [[('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)],
[('76', 1)]], [('92', 1)]], [('137', 0)]], [('57', 1)]], [('30', 1)]]
31 [('16', 1), [('31', 0)]]
32 [[('1', 0)], ('77', 1)], [[('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)], [('76', 1)]], [('92', 1)]], [[('8', 1), [('1', 0)], ('77', 1)], [[[[('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)], [('76', 1)]], [('106', 0)]], [('105', 1)]], [('60', 0)]], [('32', 1)]]
33 [('1', 0)], ('77', 1)], [('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)], [('76', 1)]],

[('92', 1)]], [('61', 1)]], [[[('9', 0), [('109', 1)]], [[('1', 0)], ('77', 1)], [('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)], [('76', 1)]], [('110', 1)]], [('108', 0)]], [('62', 1)]], [('33', 0)]]
34 [[('1', 0)], ('77', 1)], [('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)], [('76', 1)]], [('92', 1)]], [[('10', 1), [('1', 0)], ('77', 1)], [('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)], [('76', 1)]], [('112', 0)]], [('111', 1)]], [('64', 0)]], [('34', 1)]]
35 [[('1', 0)], ('77', 1)], [('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)], [('76', 1)]], [('92', 1)]], [('65', 1)]], [('11', 0), [('115', 1)]], [[('1', 0)], ('77', 1)], [('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)],
[('188', 1)]], ('78', 1)], [('76', 1)]], [('116', 1)]], [('114', 0)]], [('66', 1)]], [('35', 0)]]
36 [('12', 1), [('95', 1)]], [[('73', 1)]], [('96', 1)]], [('67', 0)]], ('36', 1)]
37 [('13', 0), [('73', 1)], ('144', 0)], [('97', 0)]], [('68', 1)]], ('37', 0)]
38 [('14', 1), [('99', 1)]], [[[('73', 1)]], [('100', 1)]], [('69', 0)]], ('38', 1)]
39 [('15', 0), [('73', 1)]]], ('146', 0)], [('101', 0)]], [('70', 1)]], ('39', 0)]
48 [[('2', 1), [('71', 1)]], [('48', 0)]]
49 [[('3', 0), [('91', 0)]], ('72', 1)], [('49', 1)]]
43 [[('4', 1)], ('43', 0)]
42 [[('5', 0)], ('42', 1)]
41 [[('6', 1)], ('41', 0)]
40 [[('7', 0)], ('40', 1)]
47 [[('8', 1)], ('47', 0)]
46 [[('9', 0)], ('46', 1)]
45 [[('10', 1)], ('45', 0)]
44 [[('11', 0)], ('44', 1)]
50 [('1', 0)], ('77', 1)], [('2', 1)], ('189', 1)], [[('3', 0)], ('190', 1)], [('188', 1)]], ('78', 1)], [('76', 1)]], ('50', 0)]

s349_f

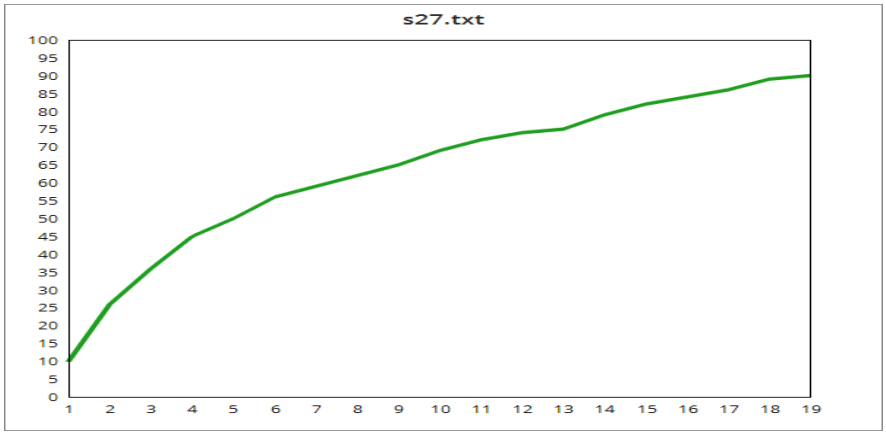
25 [('16', 1), [[('1', 0), [('176', 0)]],[('174', 1)]], [('180', 0)]], [('51', 1)]], [('25', 0)]]
26 [('182', 0)]], [('52', 0)]], [('26', 1)]]
27 [('16', 1), [[('3', 0), [('179', 0)]], [('1', 0)], ('73', 1)], [('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], [('183', 0)]], [('53', 1)]], [('27', 0)]]
28 [('16', 1), [[('4', 1), [('1', 0)], ('73', 1)], [[('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)],
[('72', 1)]], [('116', 1)]], [('109', 0)]], [('54', 1)]], [('28', 1)]]
29 [('16', 1), [('29', 0)]]
30 [('16', 1), [[('6', 1), [[('1', 0)], ('73', 1)], [[('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)],
[('72', 1)]], [('116', 1)]], [('113', 0)]], [('56', 1)]], [('30', 1)]]
31 ('16', 1), [('31', 0)]]
32 [('1', 0)], ('73', 1)], [('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], [('116', 1)]], [[('8', 1), [('1', 0)], ('73', 1)], [[('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], [('120', 0)]], [('118', 1)]], [('58', 0)]], [('32', 1)]]
33 [[('1', 0)], ('73', 1)], [[('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], [('122', 1)]], [('9', 0), [('123', 1)]], [('121', 0)]], [('60', 1)]], [('1', 0)], ('73', 1)], [('2', 1)], ('188', 1)], [[('3', 0)],
('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], [('116', 1)]], [('61', 1)]], [('33', 0)]]
34 [[('1', 0)], ('73', 1)], [('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], [('116', 1)]], [[('10', 1), [('1', 0)], ('73', 1)], [[('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], [('126', 0)]], [('124', 1)]], [('62', 0)]], [('34', 1)]]
35 [('1', 0)], ('73', 1)], [('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], [('128', 1)]], [('11', 0), [('129', 1)]], [('127', 0)]], [('64', 1)]], [[[[('1', 0)], ('73', 1)], [[[[('2', 1)], ('188', 1)]
,
[[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], [('116', 1)]], [('65', 1)]], [('35', 0)]]
36 [[('138', 1)]], [('130', 1)]], [('12', 1), [('131', 1)]], [('66', 0)]], ('36', 1)]
37 [('13', 0), [('138', 1)]]], ('171', 0)], [('133', 0)]], [('67', 1)]], ('37', 0)]
38 [('138', 1)]], [('134', 1)]], [('14', 1), [('135', 1)]], [('68', 0)]], ('38', 1)]
39 [[('15', 0), [('138', 1)]]], ('173', 0)], [('137', 0)]], [('69', 1)]], ('39', 0)]
40 [[('3', 0), [('181', 0)]], ('70', 1)], [('40', 1)]]
41 [[('2', 1), [('71', 1)]], [('41', 0)]]

42 [[('1', 0)], ('73', 1)], [('2', 1)], ('188', 1)], [[('3', 0)], ('189', 1)], [('187', 1)]], ('74', 1)], [('72', 1)]], ('42', 0)]
43 [[('11', 0)], ('43', 1)]
44 [[('10', 1)], ('44', 0)]
45 [[('9', 0)], ('45', 1)]
46 [[('8', 1)], ('46', 0)]
47 [[('7', 0)], ('47', 1)]
48 [[('6', 1)], ('48', 0)]
49 [[('5', 0)], ('49', 1)]
50 [[('4', 1)], ('50', 0)]

PART B:

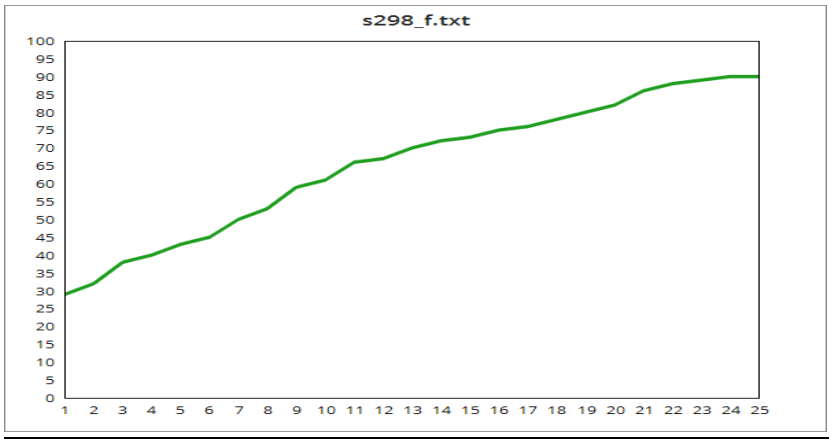
s27.txt

Number of input vectors required for 75% coverage 13
Number of input vectors required for 90 % coverage are - 17



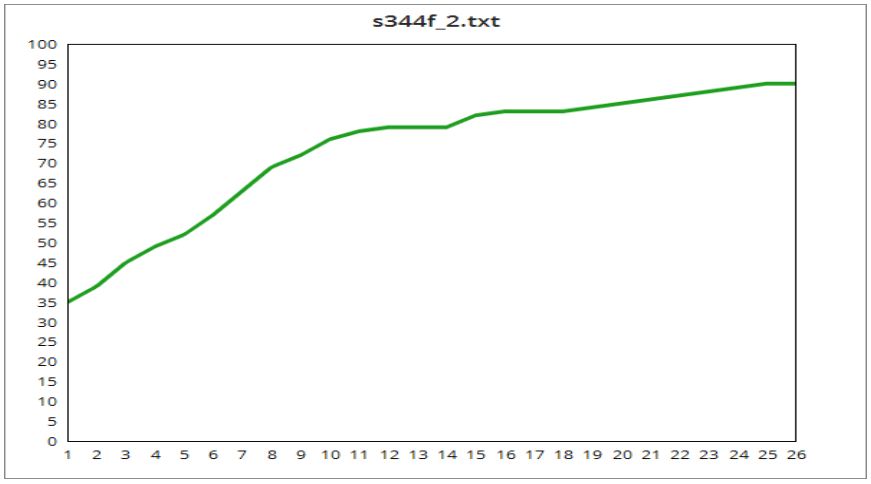
s298f_2.txt

Number of input vectors required for 75 % coverage are - 16
Number of input vectors required for 90 % coverage are - 26



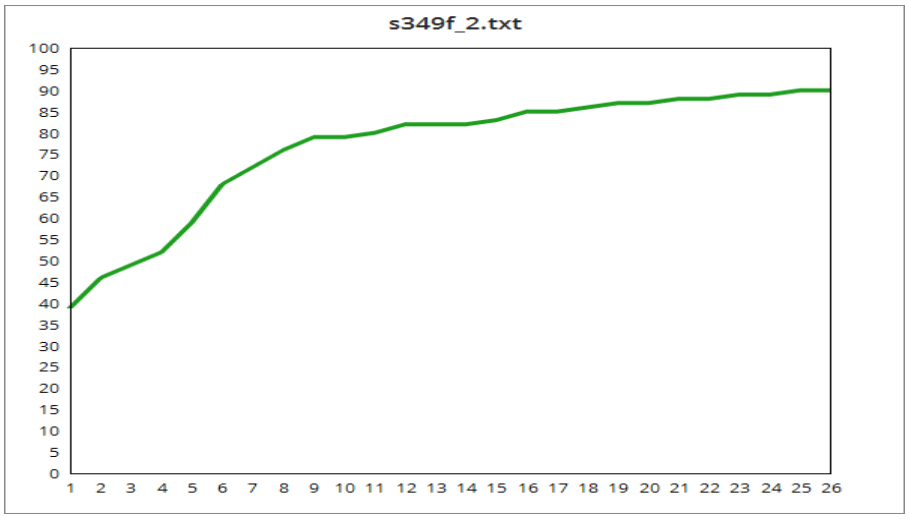
s344f_2.txt

Number of input vectors required for 75 per cent coverage are - 9
Number of input vectors required for 90 per cent coverage are – 21



s349f_2.txt

Number of input vectors required for 75 per cent coverage are - 14
Number of input vectors required for 90 per cent coverage are - 25



Part 3 : PODEM

The Pseudo code used is as follows:

Each of these functions have been implemented separately.

PODEM()
begin
if (error at PO) then return SUCCESS
if (test not possible) then return FAILURE
(k,Vk) =Objective()
(j,Vj) = Backtrace(k,Vk) /* j is a PI */
Imply (j,Vj)
if *PODEM()* = SUCCESS then return SUCCESS
/* reverse decision */
Imply (j,Vj)
if *PODEM()* =SUCCESS then return SUCCESS
Imply (j,x)
return FAILURE
end

Backtrace (k, Vk)
/* map objective into PI assignment */
begin
V=Vk
while *k* is a gate output
begin
i = inversion of *k*
select an input (j) of *k* with value *x*
v=v^i
k=j
end
/* *k* is a PI */
return *(k,v)*
end

Explanation: The Backtrace function basically performs a back calculation of gate input values till it reaches any primary input in order to satisfy a perticular objective. We do this by exoring the inversion parity of the particulat gate with its value at that gate.We do this untill we reach a primary input.

Objective()
begin
/* the target fault is *I s-a*-v */
if (the value of *I* is *x)* then return *(/,V)*
select a gate *(G)* from the *D-frontier*
select an input (j) of G with value *x*
c = controlling value of G
return (j,C)
end

Explanation: The Objective function has two parts.The first provides objectives in order to activate the fault.The second one provides objectives after the fault has been activated and we now need to find the test vector.The output of the objective function is fed to the Backtrace function.For activating the fault we first check if the value on the fault net has been changed to a D or a Dbar.If not,we return the fault net and the complement of the fault value as the objective.Once that has been done, the objective function now aims at propagating the D or Dbar towards the output nets.

Imply()

Explanation: The Imply function has two parts too.The first part is a 5 valued logic simulator with 0,1,x,D,Dbar as the logic values.After every logic simulation we check whether the fault has been activated or not.The second part is the formation of the D frontier.Here, we go through all the gates and check which gates fulfill the conditions for being on the D frontier.

RESULTS:

s27

(16,0) - **x0x10x0**

(10,1) - **10x10x0**

(12,0) - **1xxx1xx**

(18,1) **- 10x0010**

s298f_2

(70,1)**- 01x1xxxxxxxxxx0xx**

(73,0)**- 111xxxxxxxxxx0xx**

(26,1)**- xx1x1xxx0xxxxxxx**

(92,0)**- x10101xxxxxx0x0xx**

s344f_2

(71,1)**-10xxxxxxxxxxxxxxxxxxxxx**

(16,0)**-xx0xxxxxxxxxxx1xxxxxxx**

(91,1)**-111xxxxxxxxxxxxxxxxxxxx**

(166,0)**-01x00xxxxx011xx0xxxxxxx**

s349f_2

(25,1) **-xxxxxxxxxxxxxxx1xxxxxxx**

(51,0) **-00xxxxxxxxxxxxx0xxxxxxx**

(105,0)**- 01x0000xxx01xx10xxxxxxx**

(7,0) - **xxxxxx1xxxxxxxxxxxxxxxx**