ASSIGNMENT 3

ROLL NO 193079024

Name – Sarvesh Kale

Q1.) FSM EQUIVALENCE

I used 51 variables ,3 of them marks the initial state of the two fsm ,those variables are 1,2,and 51 respectively .the outputs of fsm1 are variables 6,10,14,18,22,26,30,34 and output from fsm2 are 43,44,45,46,47,48,49,50 respectively ,then we XOR the outputs at each time step and if it is SAT then that means that at some time step , the two outputs of the fsm mismatch and that is why you get a SAT ,since we used XOR of the output ,if the two fsm are same then output from XOR is always 0 , in other words we should get UNSAT.

$minisat_static assignment3q1.cnf output3.txt

The above command runs the cnf expression in DIMACS format in the .cnf file and outputs it to output3.txt

```
sarvesh@sarvesh-Inspiron-3541:~/minisat_for_class/test$ cat output3.txt
SAT
-1 -2 3 -4 5 6 7 8 9 -10 11 -12 -13 14 15 -16 17 18 19 20 21 -22 23 -24 -25 26 27 -28 29 30 31 32 33 -34 35 -36 37
38 -39 40 41 -42 -43 44 -45 -46 47 -48 -49 50 -51 0
sarvesh@sarvesh-Inspiron-3541:~/minisat_for_class/test$
```

The cnf expressions of the two fsm are given below . The expressions are written in dimacs format the same we use to pass to the SAT solver

FSM1

Each zero encountered marks end of cnf expression

q0' is 4

-4 -1 -2 -3 0 -4 1 2 0 -4 2 3 0 4 -1 2 -3 0 4 1 -2 0 4 -2 3 0

the next state transition for q1' for left FSM in assignment q1' is 5

-1 -3 -5 0 -1 3 5 0 1 -3 5 0 1 3 -5 0

output transition output is 6

-1 -2 6 0 -1 2 -6 0 1 -2 -6 0 1 2 6 0

FSM2

Each zero encountered marks end of cnf expression

51 is initial state ,35 is output next state and 3 is input

-51 -35 -3 0 35 3 -51 0 51 -35 3 0 51 35 -3 0

43 is output ,output transition equation given below .

43 -35 0 -43 35 0

The answer is coming as SAT so we conclude that the two FSM are not equal In terms of their output during the eight time steps.

Q2.) ATPG using SAT solver

Approach =>

We are given four faults ,the strategy to detect faults is that you assume one of them is stuck at fault and then you use the cone method for generating test patterns .there are four faults given so we assume one fault at a time .I have used minisat SAT solver ,and the way that I find the test patterns is that after getting output dependent on the node where stuck at fault exist ,I feed the CNF expression of it into SAT solver and then I get a SAT output ,now to get another SAT output I modify my CNF expression and add the negated variable assignment I got while running it the first time .We add negated variables because we do not want the SAT solver to return us the same output .

Condition 1 :Assume stuck at falut at lower input of NOR

```
sarvesh@sarvesh-Inspiron-3541:~/minisat_for_class/test$ cat file.txt
c this is s_a_0 @ lower input of nor
c u v w x = 1 2 3 4
p cnf 4 8
-3 -4 0
3 4 0
1 -2 0
-1 2 0
c everything beyond this is solution
1 2 -3 4 0
1 2 3 -4 0
-1 -2 3 -4 0
-1 -2 -3 4 0
```

The cnf expression is (~w + ~x).(w + x).(u + ~v).(~u + v)

So the test patterns got by running the SAT solver 4 times is

After running for first time using command minisat_static file.txt output.txt we got SAT and assignment to variables was -1 -2 3 -4 ,now when we run it second time we do not want this to be true so we want the ~(~u.~v.w.~x) to be true so we add (u + v + ~w + x) after the whole cnf expression which is (1 2 -3 4 0) , so the test patterns we got are the following .

{u,v,w,x}= {(u=0,v=0,w=1,x=0)},

{(u=0,v=0,w=0,x=1)},

{(u=1,v=1,w=0,x=1)},

{(u=1,v=1,w=1,x=0)}

Condition 2.) Assume stuck at fault at upper input of NOR

```
sarvesh@sarvesh-Inspiron-3541:~/minisat_for_class/test$ cat file2.txt
c this is s_a_1 @ upper input of nor
c u v w x = 1 2 3 4
p cnf 4 8
1 -2 0
-1 2 0
3 -4 0
-3 4 0
c evrything after this is a test vector
1 2 3 4 0
1 2 -3 -4 0
-1 -2 -3 -4 0
-1 -2 3 4 0
```

The cnf expression is (u + ~v).(~u + v).(w + ~x).(~w + x)

The approach is similar to what was done earlier; we get four test patterns, so the test vectors are

u,v,w,x =(0, 0, 0, 0),

(0, 0, 1, 1),

(1, 1, 1, 1),

(1, 1, 0, 0),

The assignments are shown as one to one correspondence from lhs to rhs

After running the cnf for first time we get 0 0 0 0, so we want opposite of this as true so we negate the whole assignment and add that as an additional clause and get another output 0 0 1 1,we again negate this add this as cnf expression at the end and again run the solver ,after 4 runs the SAT solver gives UNSAT result so we conclude that these are our four test patterns

Condition 3.) Assume stuck at fault at upper NAND gate output

```
sarvesh@sarvesh-Inspiron-3541:~/minisat_for_class/test$ cat file3.txt
c this is s_a_0 @ upper intermediate
c u v w x = 1 2 3 4
p cnf 4 7
-1 2 0
1 -2 0
3 -4 0
-3 4 0
c everything below this is test vector
1 2 3 4 0
1 2 -3 -4 0
-1 -2 -3 -4 0
-1 -2 3 4 0
```

The cnf expression obtained by cone method is

(~u + v).(u + ~v).(w + ~x).(~w + x)

We approached in a similar manner, the test vectors we get are

u,v,w,x = (0, 0, 0, 0) ,

(0, 0, 1, 1),

(1, 1, 1, 1),

(1, 1, 0, 0) .

There is one to one correspondence between u,v,w,x and one vector on the rhs .We run the SAT solver first time and get the SAT assignments as u,v,w,x as all 0 then we want opposite of this to be true so we negate the assignment and add that as an additional clause and increase the number of clause value by one ,after adding 4 additional clauses to the 4 previous clauses that we began with ,we get UNSAT that is that there is no other test vector apart from these vectors that can detect the stuck at fault at the output of upper NAND gate .

Condition 4.) Assume stuck at fault at lower output of NAND

```
sarvesh@sarvesh-Inspiron-3541:~/minisat_for_class/test$ cat file4.txt
c this is s_a_0 @ lower intermediate
c u v w x = 1 2 3 4
p cnf 4 7
-3 0
4 0
-3 4 0
1 -2 0
-1 2 0
c everything below this is a test vector
1 2 3 -4 0
-1 -2 3 -4 0
```

The cnf expression is (~u)(x)(~w + x)(u + ~v)(~u + v).

We used a similar approach as mentioned ,we ran the minisat_static solver two times before it gave an UNSAT answer and we kept adding negations of the variable assignments obtained after each run at the end of this file and ran the SAT solver again .there is also another way ,just take the cnf expression and run a picosat SAT solver on it with –all flag and your file which has DIMACS formatted cnf expressions in it ,it will give you all assignments for which the the cnf evaluates to true ,but I got to know about picosat a little later so I kept the files of minisat_static .

The test pattern is

u,v,w,x = (0, 0, 0, 1)

(1, 1, 0, 1).