VLSI Testing PA3 Report

1. Testcase result

circuit number	number of gates	number of total TDFs	number of detected faults	number of undetected faults	transition delay fault coverage
C17	6	34	23	11	67.64%
C432	245	1110	3	1107	0.27%
C499	554	2390	1552	838	64.93%
C880	545	2104	792	1312	37.64%
C1355	554	2726	593	2133	21.75%
C2670	1785	6520	4668	1852	71.59%
C3540	2082	7910	1142	6768	14.43%
C6288	4800	17376	16532	844	95.14%
C7552	5679	19456	17421	2035	89.54%

2. Code explanation

a. First we need to generate fault list (this function has got some changes) using similar algorithm as the original one. The difference is that we only collapse equivalent faults for BUF and NOT gates, because some equivalent stuck-at faults are not equivalent TDFs. For example, in c17.ckt, g2's input from PI3 s-a-0 is equivalent to g2's output s-a-1, but when applying pattern T'01111 1', the first fault is not activated, while the second fault is activated and detected. (question)

```
33    switch (n->type) {
34          case   NOT:
35      case   BUF:
36        f->eqv_fault_num = 1;
37        for (wptr wptr_ele: w->inode.front()->iwire) {
38          if (wptr_ele->onode.size() > 1) f->eqv_fault_num++;
39        }
40        break;
41      case INPUT:
42      case   AND:
43      case   NOR:
44      case    OR:
45      case  NAND:
46      case   EQV:
47      case   XOR: f->eqv_fault_num = 1; break;
48    }
```

b. Applying the first pattern V1 (in file faultsim.cpp) .

```
56 //check V1
57   for(i = 0; i < cktin.size(); i++) {
58     cktin[i]->value = ctoi(vec[i]);
59   }
60
61
62    nckt = sort_wlist.size();
63   for (i = 0; i < nckt; i++) {
64     if (i < cktin.size()) {
65       sort_wlist[i]->flag |= CHANGED;
66     }
67     else {
68       sort_wlist[i]->value = U;
69     }
70   }
```

c. Simulate V1 and check for the activated faults. We mark the activated faults.

```
//simulate V1 and check activated faults

  sim();

 for (auto pos = flist_undetect.cbegin(); pos != flist_undetect.cend(); ++pos){
                f = *pos;

    if (f->fault_type == sort_wlist[f->to_swlist]->value)
      f->activate = TRUE;
    else
      f->activate = FALSE;
  }
```

d. Apply second pattern and run fault simulation (faultsim.cpp). The only change in fault simulation is that we need to additionally check if a fault is activated.

```
 86  // V2 for every input, set its value to the current vector value
 87   for(i = 0; i < cktin.size(); i++) {
 88     cktin[i]->value = ctoi(vec[i-1]);
 89   j = cktin.size();
 90         cktin[0]->value = ctoi(vec[j]);
 91
 92
 93   }
 94
 95
 96   /* initialize the circuit - mark all inputs as changed and all other
 97    * nodes as unknown (2) */
 98  nckt = sort_wlist.size();
 99   for (i = 0; i < nckt; i++) {
100     if (i < cktin.size()) {
101       sort_wlist[i]->flag |= CHANGED;
102     }
103     else {
104       sort_wlist[i]->value = U;
105     }
106   }
107
108
109   sim(); /* do a fault-free simulation, see sim.c */
```

e. Other changes made in code

get_faulty_wire (faultsim.cpp) – changed the fault type

Note:

In my code , I have print tdf faults before fault list. So please look at it. Like this way-

Student Id - R07943158