ATPG SYSTEM DESIGN FLOW

1.Objective

Give a circuit, ATPG will return a list of test patterns and a high fault coverage with good performance

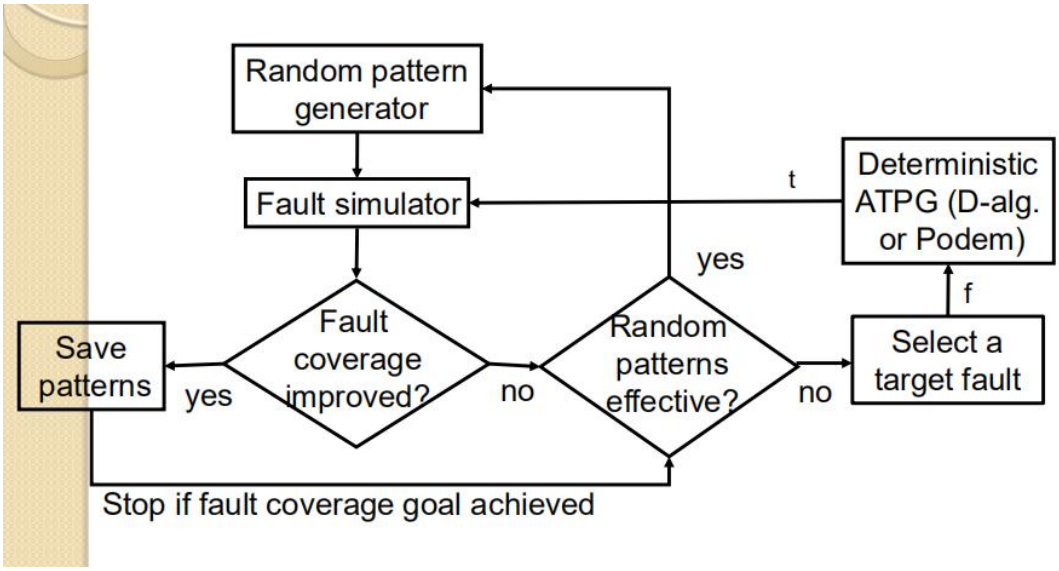
2.The functions for ATPG

List all the sub functions that have been implemented in current stage

- readCircuit()
- lev()
- getFullFaultList()
- getReducedFaltList()
- getRandomInputPattern()
- fault simulation including pfs(), dfs()
- Atpg including D_alg() and podem()

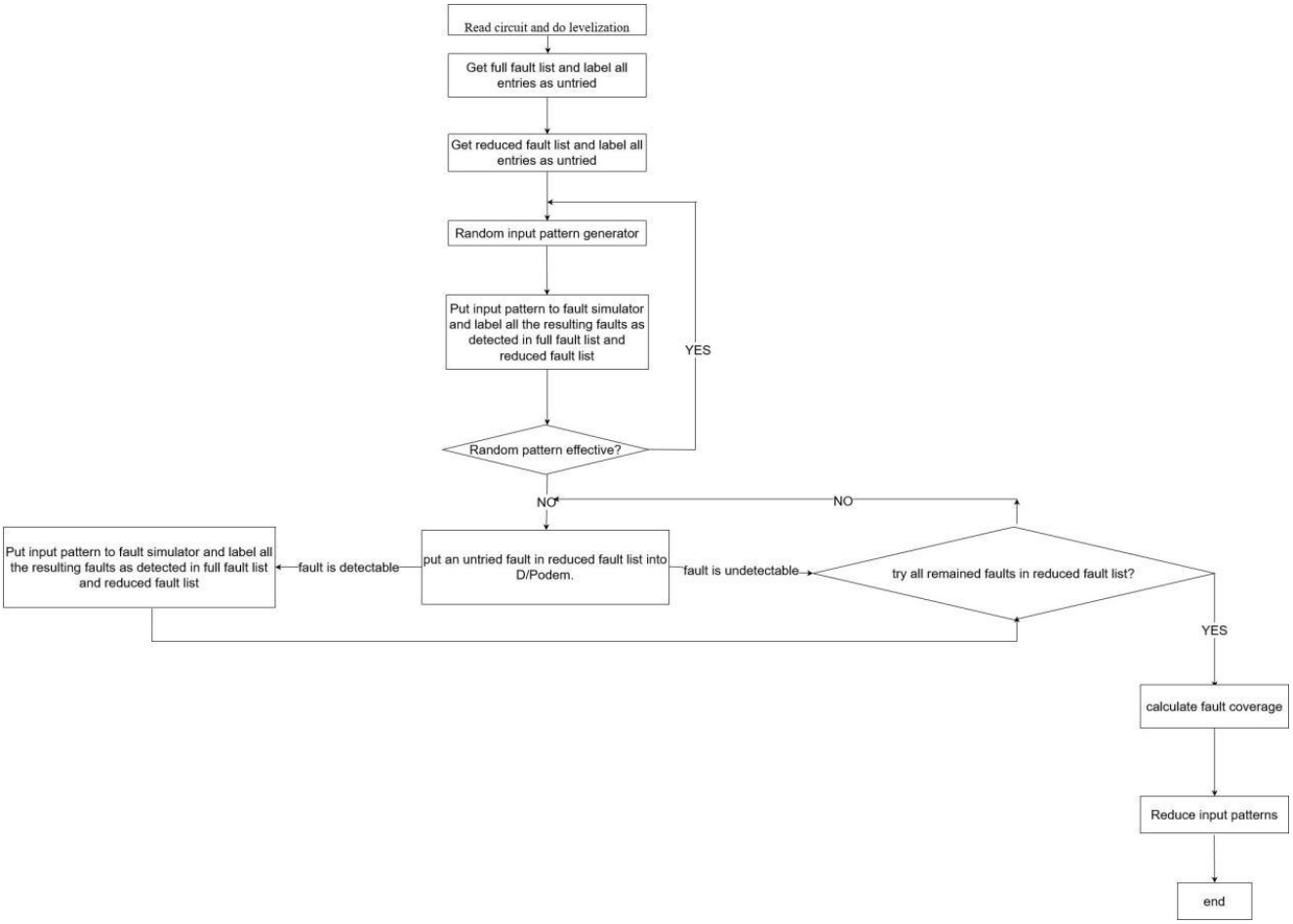
3. Base design flow

The screenshot is from EE658 slides

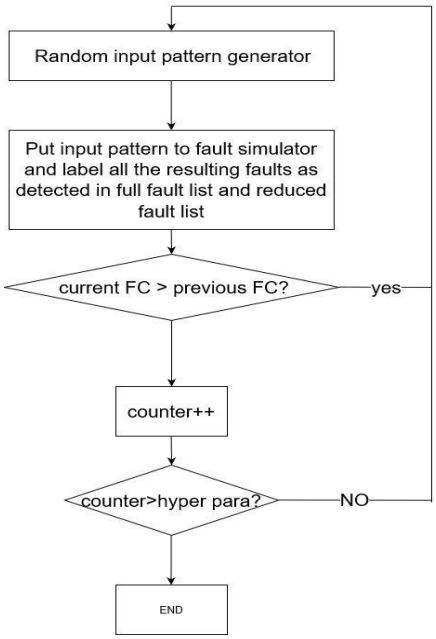


4. Implementation

ATPG flowchart



Random pattern effective flowchart



ATPG implementation result

.ckt	Hyper-para	Timing(s)	Fault Coverage	The number of input pattern
C17	1	0.0048	100%	5
C1	1	0.0049	94%	5
C2	2	0.09	88%	6
C3	2	0.07	82%	6
C4	2	0.018	92%	6
fa	1	0.003	100%	5
X3mult	4	0.01	100%	7
Add2	5	0.014	100%	6
C432	43	11	98%	50
C499	49	39	97%	49
C880	86	5	100%	64
C1355	135	105	99.7%	88

How to determine hyper parameter

- For bigger circuits, we need to give them more “patience”, which means that we need to set a bigger hyper parameter and generate input pattern more times
- For given circuit, there is a positive correlation between the number of nodes and hyper parameter
- The equation is $hyperpara = node\_number / 10$

The detailed explanations for every steps are as follows

1) Read circuit
2) Do levelization
3) Get full fault list and label all entries as untried. Full fault dict is used to label all the faults in full fault list. Key is one of the fault in given circuit, Value includes three possible value -1, 0, 1. -1 means untried, 0 means undetectable, 1 means detectable
4) Get reduced fault list and label all entries as untried. Reduced fault dict is used to label all the faults in reduced fault list. Key is one of the fault in given circuit, Value includes three possible value -1, 0, 1. -1 means untried, 0 means undetectable, 1 means detectable
5) Put random input pattern to fault simulator and label all the resulting faults as detected in full fault dict and reduced fault dict.
6) Use Monte Carlo algorithm as the condition to judge whether random patterns is effective or not and save all the input patterns in input pattern list.(different circuit with different hyper parameter)
7) When we need Deterministic ATPG, we try all the remained

faults in reduced fault dict and put them into D/Podem.

8) For every remained faults in reduced fault dict, if we find a untried fault that can be detected, we get a new input pattern and go step9
9) Once we get a new input pattern, we save this input pattern in input pattern list and put it into fault simulator to check whether there are other faults that can be detected. If true, we label these faults as detected in reduced fault dict and full fault dict.
10) If all the faults in reduced fault dict can be detected, return 100% for fault coverage. Otherwise, fault coverage is calculated fault coverage based on full fault dict
11) Reduce input patterns by pattern_information.py without changing fault coverage