Department of Electronics Engineering
National Chiao Tung University
Hsinchu, Taiwan

Fault Dictionary Size Reduction for Million-Gate Large Circuits

Yu-Ru Hong and Juinn-Dar Huang

Outline

- Introduction to Fault Dictionary
- Preliminaries
- Proposed Edge Factor (EF) Based Algorithm
- Implementation Tech. on EF Calculation
- Experimental Results
- Conclusions

Fault Diagnosis

- Fault detection
  - apply test vectors to find whether a circuit under test (CUT) is good
- Fault diagnosis
  - determine the **cause and location** of the fault
  - improve the design or manufacturing process
- Cause-effect diagnosis
  - assume one or more than one fault model
  - use simulator to predict behavior of the CUT in the presence of modeled fault
  - compare the observed responses to the simulated results

Fault Dictionary for Fault Diagnosis

- Fault dictionary
  - a database storing the simulated responses for all modeled faults
  - used by some diagnostic algorithms
  - fast & self-contained
  - **can be extremely large**
- Full fault dictionary
  - for each fault f_i, store the full output responses o_i of the circuit to each test vector t_i
  - **size: O(|T||F||O|)**
    - |T|: # of test vectors; |F|: # of faults; |O|: # of the outputs
  - not practical even for small circuits

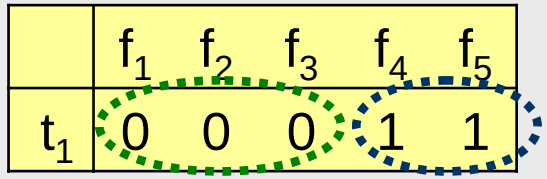
Previous Works

- Well-known fault dictionaries
  - pass-fail dictionary
  - vector dictionary
- B. Arslan and Orailoglu proposed an XOR approach in ICCD'02 [6]
  - is based on pass-fail dictionary
  - partition the test set and store a combined signature for each partition
  - trade minor loss in diagnostic resolution for greater compaction
  - need to construct a distinguishability table of $\mathbf{O(|T||F|^2)}$
- **The dictionary size reduction should be done more efficiently**!

Preliminaries

	f_1	f_2	f_3	f_4	f_5
t_1	0	0	0	1	1
t_2	0	1	1	1	1
t_3	1	1	0	0	1
t_4	0	1	1	0	0

Pass-fail Dictionary D1



F_{E1}= {f_1, f_2, f_3};
F_{E2} = {f_4, f_5}

Fault Equivalence Set

	(f_1,f_2)	(f_1,f_3)	(f_1,f_4)	(f_1,f_5)	(f_2,f_3)	(f_2,f_4)	(f_2,f_5)	(f_3,f_4)	(f_3,f_5)	(f_4,f_5)
t_1	0	0	1	1	0	1	1	1	1	0
t_2	1	1	1	1	0	0	0	0	0	0
t_3	0	1	1	0	1	1	0	0	1	1
t_4	1	1	0	0	0	1	1	1	1	0

Distinguishability Table A1

Greedy Algorithm in [6]

- Finding a minimum number of rows to cover all fault pairs
  - an NP-complete problem
  - A sub-optimal heuristic solution: greedy algorithm

	(f_1,f_2)	(f_1,f_3)	(f_1,f_4)	(f_1,f_5)	(f_2,f_3)	(f_2,f_4)	(f_2,f_5)	(f_3,f_4)	(f_3,f_5)	(f_4,f_5)
t_1			**1**	**1**		**1**	**1**	**1**	**1**	
t_2	**1**	1	1	1						
t_3		**1**	1		**1**	1			1	**1**
t_4	1	1				1	1	1	1	

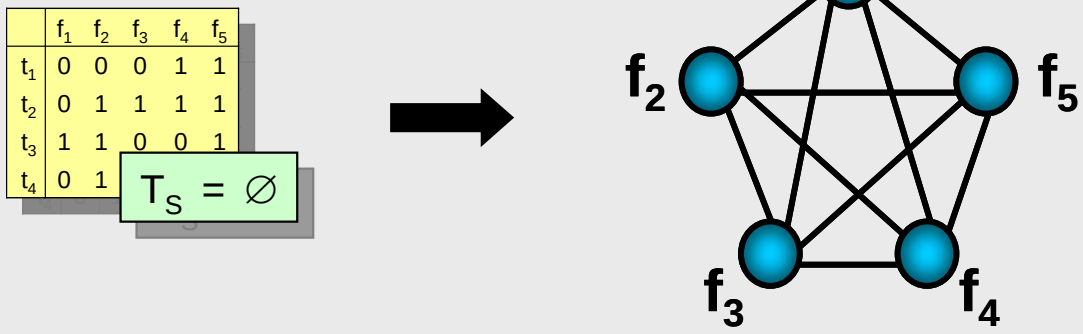
Selected test set $T_S = \{t_1,t_3,t_2\}$

The time and space complexity of the distinguishability table is very high: $O(|T||F|^2)$!

Problem Description

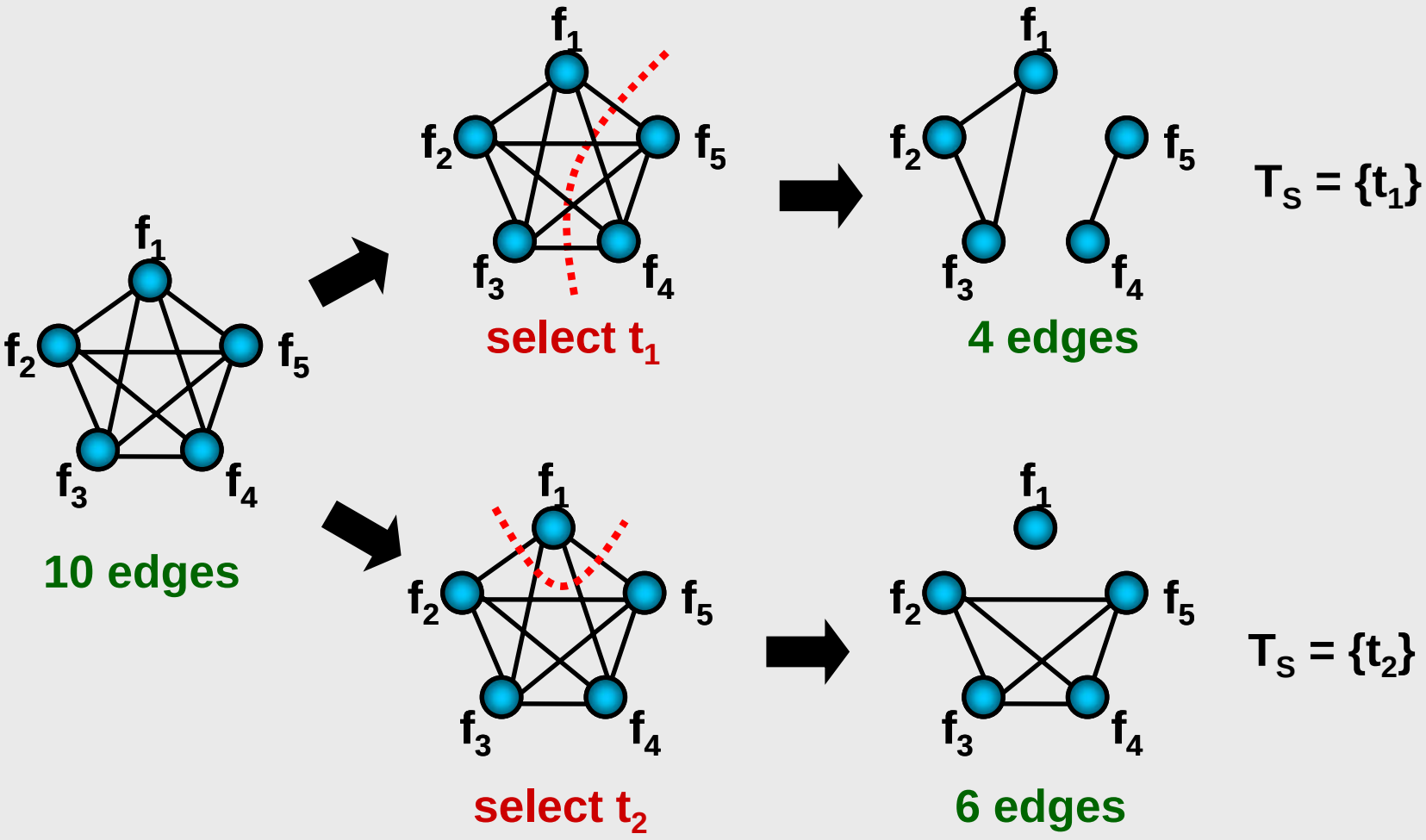
- Input: a pass-fail dictionary D
- Output: a selected test set T_S
- Objective: adopt the similar greedy algorithm as in [6], while bypassing the distinguishability table for much more efficient runtime and memory usage

Another View: Graph



- An undirected graph G
  - edge (v_i, v_j) $\Leftrightarrow$ fault pair (f_i, f_j) is indistinguishable
  - $T_S = \varnothing \Rightarrow$ G is a complete graph
- T_S partitions G into **connected components**
  - Faults in a connected component $\Leftrightarrow$ **fault equivalence set**
  - a connected component $\Leftrightarrow$ a **complete graph**

An Illustrating Example



Goal: Minimize the number of the **remaining edges** in G

Edge Factor

- Assume a given T_s partitions the graph into n fault equivalent sets, F_{E1}, F_{E2}, F_{E3}, …, F_{En}. The number of remaining edges in the graph is

$$\binom{|F_{E1}|}{2}+\binom{|F_{E2}|}{2}+\cdots+\binom{|F_{En}|}{2}$$

$$=\frac{|F_{E1}|(|F_{E1}|-1)}{2}+\frac{|F_{E2}|(|F_{E2}|-1)}{2}+\cdots\frac{|F_{En}|(|F_{En}|-1)}{2}$$

$$=\frac{|F_{E1}|^2+|F_{E2}|^2+\cdots|F_{En}|^2-(|F_{E1}|+|F_{E2}|+\cdots|F_{En}|)}{2}$$

$$=\frac{EF(T_s)-|F|}{2} \quad \text{where } EF(T_s)=\sum_{i=1}^{n}|F_{Ei}|^2$$

- The smaller the ***Edge Factor (EF)***, the higher the diagnostic resolution of the test set

Proposed EF_based Algorithm

```
EF_based(F, T, N) {
  Ts ← ∅
  do
    for i ← 1 to |T|
      if ti ∉ Ts
        Yi ← Ts ∪ {ti}
        calculate EF(Yi)
    identify tj with the smallest EF
    Ts ← Ts ∪ {tj}
  while (EF(Ts) > |F| and |Ts| < N)
  return Ts
}
```

N is the given upper bound of the number of selected vectors
$\mathbf{T_s}$ is the set of selected test vectors
$\mathbf{Y_i}$ is a temporary test set

How to Calculate EF (1/2)

- Edge factor
  - SID
  - hash table

$$EF(T_s) = \sum_{i=1}^{n} | F_{Ei} |^2$$

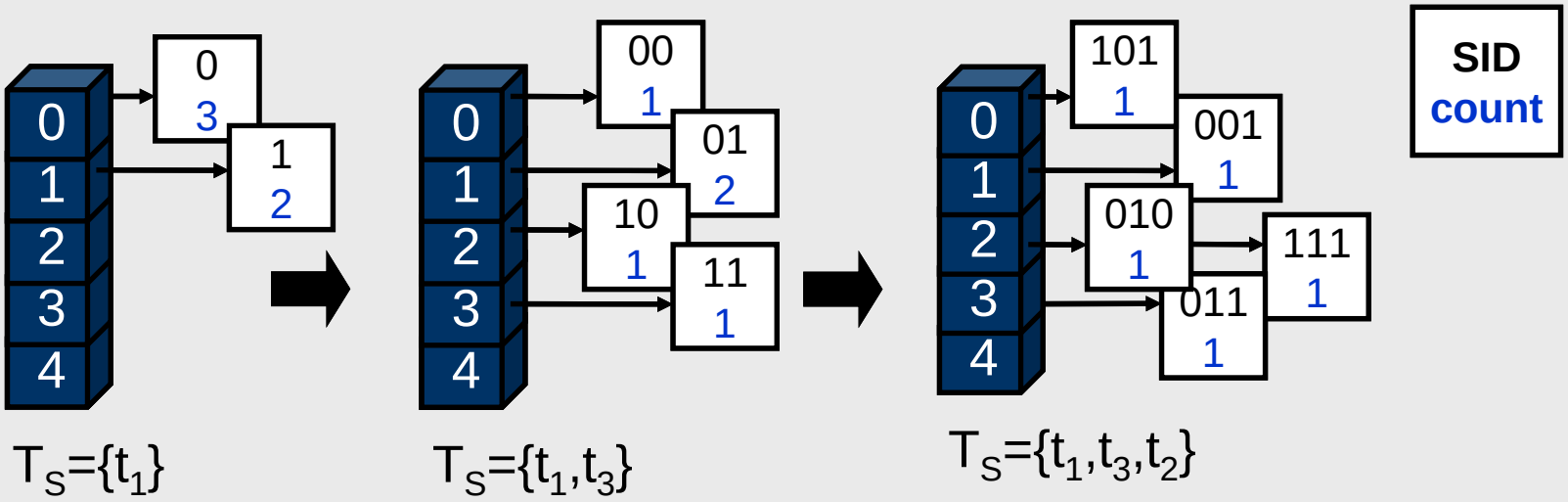
	f_1	f_2	f_3	f_4	f_5
t_1	0	0	0	1	1
t_2	0	1	1	1	1
t_3	1	1	0	0	1
t_4	0	1	1	0	0

Pass-fail Dictionary D1

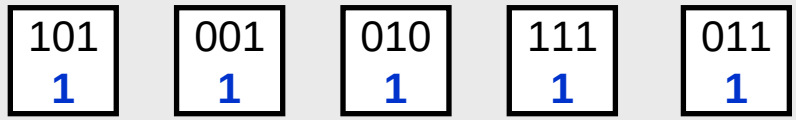
	f_1	f_2	f_3	f_4	f_5
$T_S=\{t_1\}$	0	0	0	1	1
$T_S=\{t_1,t_3\}$	01	01	00	10	11
$T_S=\{t_1, t_3, t_2\}$	010	011	001	101	111

SID

How to Calculate EF (2/2)



Hash Table



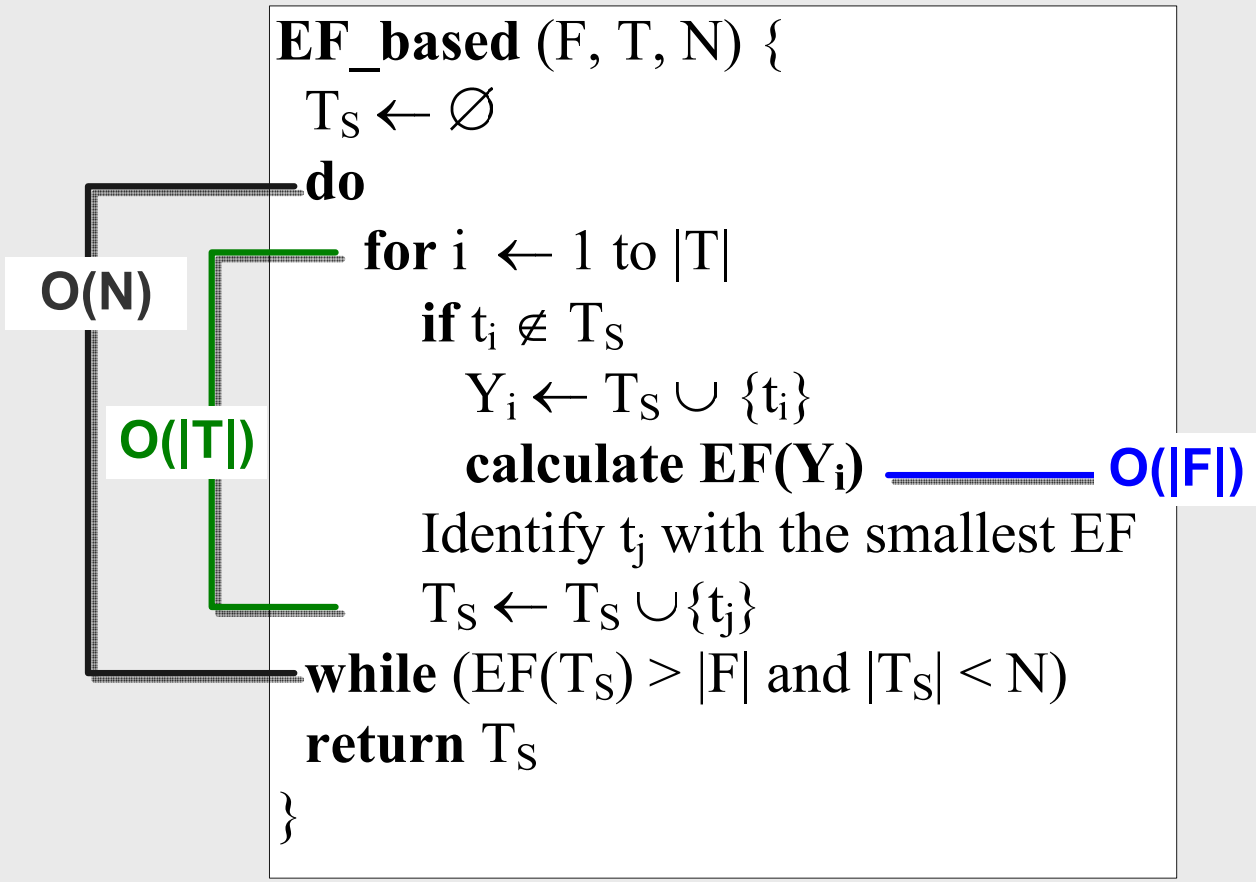
EF = 1^2 + 1^2 + 1^2 + 1^2 + 1^2 = 5

$|F_{E1}|^2 + |F_{E2}|^2 + |F_{E3}|^2 + |F_{E4}|^2 + |F_{E5}|^2$

- Time of calculating an EF(T_S) is **O(|F|)**

Complexity of EF_based Algorithm

- Time complexity: **O(N|T||F|)**
- Space complexity: **O(|T||F|)**



Experimental Setup

- The proposed algorithm is implemented using C++ on a Pentium4 3.0GHz platform
- ISCAS’85 and ’89 benchmark circuits
- Large randomly generated fault dictionaries
- Atalanta test generator + HOPE fault simulator
- All fault dictionaries are preprocessed by XORing the responses

Performance Comparison

- ISCAS'85 benchmark circuits
- Our results are comparable to those in [6]

Circuit	|T|	|F|	$\lceil \log_2 \|F\| \rceil$	DR before reduction	DR after reduction	DR in [6]
c1355	86	1574	11	0.999241	0.997302	0.997274
c1908	120	1879	11	0.999403	0.997000	0.997063
c2670	106	2747	12	0.997739	0.996207	0.996088
c3540	150	3428	12	0.998169	0.996568	0.996694
c5315	125	5350	13	0.999776	0.999268	0.999083
c6288	30	7744	13	0.999815	0.999578	0.999555
c7552	217	7550	13	0.999566	0.998992	0.998871

Performance Evaluation

- ISCAS'89 benchmark circuits

Circuit	|T|	|F|	$\lceil \log_2 \|F\| \rceil$	DR after reduction	DR Loss	CPU (s)	Mem (MB)
s9234	384	6927	13	0.993234	0.24%	3.56	2
s13207	460	9815	14	0.998303	0.13%	6.72	4
s15850	438	11725	14	0.997933	0.09%	7.69	6
s35932	68	39094	16	0.989422	0.02%	4.02	6
s38417	901	31180	15	0.999587	0.04%	41.69	28
s38584	654	36303	16	0.997943	0.03%	42.47	24

- Summary of ISCAS'85+'89 benchmark sets
  - average size reduction: **87.8%**
  - average DR loss: **0.16%**
  - runtime: less than one minute

Runtime & Memory Evaluation

- Large randomly generated fault dictionaries

$\|T\|$	$\|F\|$	$\lceil \log_2 \|F\| \rceil$	CPU (s)	Mem (MB)
1000	50000	16	95.970	48
1000	100000	17	222.510	96
1000	500000	19	1579.220	492
1000	**1000000**	**20**	**3733.180**	**972**

Distinguishability table: $10^3*(10^6)^2 = \mathbf{10^{15}}$!

- A modest computer nowadays can run the proposed algorithm to process million-gate circuits

Conclusions

- Proposed EF-based Algorithm
  - introduce edge factor as a guidance
    - bypass distinguishability table
    - is promising to be used in other algorithms
  - provide detailed implementation techniques
  - low time and space complexity

- Fast runtime & low memory usage
- A feasible solution for million-gate circuits

Thank you for your attention!