ECE 553: TESTING AND TESTABLE DESIGN OF DIGITAL SYSTES

Combinational ATPG Basics

Overview

- Structural vs. functional test
- Definitions
- Completeness
- Conditions for finding a test
- Algebras
- Types of Algorithms – classical
- Complexity
- Summary
- Appendices

Functional vs. Structural ATPG

Carry Circuit

Functional vs. Structural (Contd.)

- **Functional ATPG – generate complete set of tests for circuit input-output combinations**
  - 129 inputs, 65 outputs:
  - 2^{129} = 680,564,733,841,876,926,926,749, 214,863,536,422,912 patterns
  - Using 1 GHz ATE, would take 2.15 x 10^{22} years
- **Structural test:**
  - No redundant adder hardware, 64 bit slices
  - Each with 27 faults (using fault equivalence)
  - At most 64 x 27 = 1728 faults (tests)
  - Takes 0.000001728 s on 1 GHz ATE
- **Designer gives small set of functional tests – augment with structural tests to boost coverage to 98^{+} %**

Definition of Automatic Test-Pattern Generator

- **Operations on digital hardware:**
  - Inject fault into circuit modeled in computer
  - Use various ways to activate and propagate fault effect through hardware to circuit output
  - Output flips from expected to faulty signal
- ***Electron-beam* (*E-beam*) *test* observes internal signals**
  - "picture" of nodes charged to 0 and 1 in different colors
  - Too expensive
- ***Scan design* – add test hardware to all flip-flops to make them a shift register in test mode**
  - Can shift state in, scan state out
  - Widely used – makes sequential test combinational
  - Costs: 5 to 20% chip area, circuit delay, extra pin, longer test sequence

Algorithm Completeness

- **Definition: Algorithm is c*omplete* if it ultimately can search entire binary (decision) space, as needed, to generate a test**
- ***Untestable fault or Undetectable fault* – no test for it even after entire space is searched**
- **Combinational circuits only – untestable faults are *redundant,* showing the presence of unnecessary hardware**

Notation

Symbol	Meaning	Good Circuit	Failing Circuit	
D	1/0	1	0	Roth's Algebra
$\overline{D}$	0/1	0	1	
0	0/0	0	0	
1	1/1	1	1	
X	X/X	X	X	
G0	0/X	0	X	Muth's Additions
G1	1/X	1	X	
F0	X/0	X	0	
F1	X/1	X	1	

Roth's and Muth's Higher-Order Algebras

- **Represent two machines, which are simulated simultaneously by a computer program:**
  - **Good circuit machine (1st value)**
  - **Bad circuit machine (2nd value)**
- **Better to represent both in the algebra:**
  - **Need only 1 pass of ATPG to solve both**
  - **Good machine values that preclude bad machine values become obvious sooner & vice versa**
- **Needed for complete ATPG:**
  - **Combinational: Multi-path sensitization, Roth Algebra**
  - **Sequential: Muth Algebra -- good and bad machines may have different initial values due to fault**

Conditions for Finding a Test

- **Fault excitation – the signal value at the fault site must be different from the value of the stuck-at fault (thus fault site must contain a D or a $\overline{D}$)**
- **The fault effect must be propagated to a primary output (A D or a $\overline{D}$ must appear at the output)**
- **Some simple observations**
  - **There must be at least a D or a $\overline{D}$ on some circuit nets)**
  - **D's must form a chain to some output**

Exhaustive Algorithm

- **For *n*-input circuit, generate all 2^n input patterns**
- **Infeasible, unless circuit is partitioned into cones of logic, with $\leq$ 15 inputs**
  - **Perform exhaustive ATPG for each cone**
  - **Misses faults that require specific activation patterns for multiple cones to be tested**

Random-Pattern Generation

- **Flow chart for method**
- **Use to get tests for 60-80% of faults, then switch to D-algorithm or other ATPG for rest**

Boolean Difference Symbolic Method (Sellers *et al.*)

$g = G(X_1, X_2, \ldots, X_n)$ for the fault site

$f_j = F_j(g, X_1, X_2, \ldots, X_n)$

$1 \leq j \leq m$

$X_i = 0$ or 1 for $1 \leq i \leq n$

Boolean Difference (Sellers, Hsiao, Bearnson)

- **Shannon's Expansion Theorem:**
  $F(X_1, X_2, \ldots, X_n) = X_2 \bullet F(X_1, 1, \ldots, X_n) + \overline{X_2}\ F(X_1, 0, \ldots, X_n)$
- **Boolean Difference (partial derivative):**
  $$\frac{\partial F_j}{\partial g} = F_j(1, X_1, X_2, \ldots, X_n) \oplus F_j(0, X_1, \ldots, X_n)$$
- **Fault Detection Requirements for g stuck-at 0:**
  $G(X_1, X_2, \ldots, X_n) = 1$
  $$\frac{\partial F_j}{\partial g} = F_j(1, X_1, X_2, \ldots, X_n) \oplus F_j(0, X_1, \ldots, X_n) = 1$$

Path Sensitization Method - Example

1. **Fault Sensitization**
2. **Fault Propagation**
3. **Line Justification**

Path Sensitization Method -Example

- **Try path $f - h - k - L$. This path is blocked at j, since there is no way to justify the 1 on i**

Path Sensitization Method

- **Try simultaneous paths $f - h - k - L$ and $g - i - j - k - L$. These paths blocked at k because *D-frontier* (chain of D or $\overline{D}$) disappears**

Path Sensitization Method Circuit Example

- **Final try: path $g - i - j - k - L$ – test found!**

Computational Complexity

- **Ibarra and Sahni analysis – *NP-Complete***
  (no polynomial expression found for compute time, presumed to be exponential)
- **Worst case:**
  ***no_pi* inputs, 2^{no_pi} input combinations**
  ***no_ff* flip-flops, 4^{no_ff} initial flip-flop states**
  (good machine 0 or 1 $\times$ bad machine 0 or 1)
  work to forward or reverse simulate *n* logic gates α *n*
- **Complexity: O ($n \times 2^{no_pi} \times 4^{no_ff}$)**

Summary

- Basic definitions explained
- Developed notation and required algebra that will be used for test generation and fault simulation
- Basics of test generation developed
- Complexity of test generation addressed
- Appendix contains historical reference to the stuck-at fault model, an example of BDD, an instantiation of SAT problem.

Appendices

Origins of Stuck-Faults

- **Eldred (1959) – First use of structural testing for the Honeywell *Datamatic 1000* computer**
- **Galey, Norby, Roth (1961) – First publication of stuck-at-0 and stuck-at-1 faults**
- **Seshu & Freeman (1962) – Use of stuck-faults for parallel fault simulation**
- **Poage (1963) – Theoretical analysis of stuck-at faults**

Circuit and Binary Decision Tree

(a) Circuit.

(b) Binary decision tree.

Binary Decision Diagram

- **BDD – Follow path from source to sink node – product of literals along path gives Boolean value at sink**
- **Rightmost path: $A\,\bar{B}\,\bar{C} = 1$**
- **Problem: Size varies greatly with variable order**

History of Algorithm Speedups

Algorithm	Est. speedup over D-ALG (normalized to D-ALG time)	Year
D-ALG	1	1966
PODEM	7	1981
FAN	23	1983
TOPS	292	1987
SOCRATES	1574 † ATPG System	1988
Waicukauski et al.	2189 † ATPG System	1990
EST	8765 † ATPG System	1991
TRAN	3005 † ATPG System	1993
Recursive learning	485	1995
Tafertshofer et al.	25057	1997

Analog Fault Modeling Impractical for Logic ATPG

- **Huge # of different possible analog faults in digital circuit**
- **Exponential complexity of ATPG algorithm – a 20 flip-flop circuit can take days of computing**
  - **Cannot afford to go to a lower-level model**
- **Most test-pattern generators for digital circuits cannot even model at the transistor switch level (see textbook for 5 examples of switch-level ATPG)**

Boolean Satisfiability

- **2SAT:** $x_i \bar{x}_j + x_j \bar{x}_k + x_l \bar{x}_m \ldots = 0$

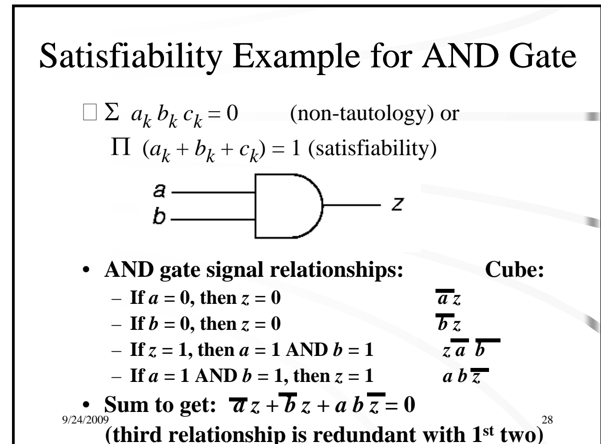
  $\vdots$

  $x_p x_y + x_r \bar{x}_s + x_t \bar{x}_u \ldots = 0$

- **3SAT:** $x_i \bar{x}_j x_k + x_j \bar{x}_k \bar{x}_l + \bar{x}_l x_m \bar{x}_n \ldots = 0$

  $\vdots$

  $x_p x_y + x_r \bar{x}_s x_t + x_t x_u \bar{x}_v \ldots = 0$



Pseudo-Boolean and Boolean False Functions

- ***Pseudo-Boolean function*: use ordinary + -- integer arithmetic operators**
  - **Complementation of x represented by $1 - x$**
  - $F_{pseudo\text{—}Bool} = 2z + ab - az - bz - abz = 0$
- ***Energy function representation*: let any variable be in the range (0, 1) in pseudo-Boolean function**
- ***Boolean false expression*:**

  $f_{AND}(a, b, z) = z \oplus (ab) = \bar{a} z + \bar{b} z + a b \bar{z}$