A Symbolic Technique for the Efficient ATPG of Speed-Independent Circuits

M.A. Peña  E. Pastor  J. Cortadella

Universitat Politecnica de Catalunya

Overview

- Goal: improve a pre-existing ATPG methodology for asynchronous circuits
- Theoretical observations:
  - Faults can be seen as particular deviations of the specified behaviour
  - It is enough to analyse stable states
- Use symbolic verification techniques based on BDDs
Outline

- Testing asynchronous circuits
- Overall approach
- Specification and circuit
- Fault effects
- Fundamental mode analysis: “a la synchronous”
- Example
- Conclusions

Testing asynchronous circuits

- High complexity
  - Arbitrary interconnection of gates + no clock
  - State is characterised by ALL signals
- The circuit cannot be “single-stepped”
  - Controllability and observability are very costly
  - Synchronous techniques cannot be used!
- Testing is still an open problem!
- Speed-Independent circuits
  - Unbounded gate delays
  - Negligible delays on wires
Motivation

- **Roig et al. (DAC’97)**
  - Generic ATPG under the input stuck-at fault model
  - Test vectors suitable for synchronous testers
    - Random TPG
    - 3 phase (fault activation, state justification, state differentiation)
    - Fault simulation
  - Good fault coverage but 3-phase is too costly

- **Beerel et al. (Integration’92)**
  - SI-circuits are self-checking for output stuck-at faults
  - The fault effect is observable at the stable states

ATPG overview

- Specification (STG)
- Confront STG and CUT
- Discrepant states: Inhibited transitions & Premature firings
- Generate back-trace
- Sequence of test vectors for fault “f”
- Circuit Under Test (CUT)
- Inject a fault “f”: CUT
- Symbolic verification & analysis
Petri nets and SI-circuits

Speed-Independent synthesis
Petri nets and SI-circuits
Petri nets and SI-circuits

[Diagram of Petri nets and SI-circuits]
Petri nets and SI-circuits

Diagram showing a Petri net and an equivalent SI-circuit.
State space

Exhaustive analysis

Theoretical observations

- Effects of stuck-at faults: Hazewindus (PhD’92)
  - Inhibited transitions
  - Premature firings

- Testability of SI-circuits: Beerel et al. (Integration’92)
  - Fault effects can be observed at stable states
Fault effects

Inhibited transitions

Stuck at 0

Inhibited transitions

Stuck at 0

d+ and e+ are inhibited!
Fault effects

Premature firings

Stuck at 1

Stuck at 1 \( e^+ \) fires prematurely!
Total stable states

Fundamental mode analysis

“a la synchronous”
**FM vs. exhaustive**

- **State space reduction:**
  - 73% less states are generated
- **Memory requirements**
  - 82% less BDD nodes
- **CPU time for state generation**
  - 13% more costly
- **Complexity reduction without loss of fault coverage**
- **Bigger circuits can be tested!**

---

**ATPG for SI-circuits**

- **Confront faulty circuit and specification by means of efficient symbolic traversal**

  - Discrepant stable states $\Rightarrow$ the circuit does not behave as specified $\Rightarrow$ the fault is testable

  - Backwards traversal $\Rightarrow$ sequence of transitions $\Rightarrow$ test vectors
ATPG for SI-circuits: traversal

Input fire synchronically (one at a time)

Non-inputs fire separately

Check binary codes

Inject a fault in the circuit

Traverse spec. and circuit synchronically
  - Deadlocks: no gate is excited, inhibited transition
  - Failure states: signal excited but not specified, premature firing

Extract test vectors from the backwards sequence of transitions
Example: traversal

00000

Stuck at 0

Stuck at 0

Example: traversal

00000

a+

10000
Example: traversal

Stuck at 0

C- is inhibited in the circuit!
Example: pattern generation

```
Example: pattern generation

ab cde
d+ 10000
e+ 10010
10011
b+ 11111
c-

Test vectors
ab cde
a+ 00000
b+ 10111
< 11 011 >
```

Conclusions

- **Novel ATPG technique for SI circuits**
  - Functional equivalence of faults instead of structural
  - Known symbolic techniques using BDDs
  - Fundamental mode: “a la synchronous” test vectors
- **Complexity reduction**
- **100% coverage of output SAT-faults**
- **Future work**
  - Complete implementation + integration with **testify**
  - Extend fundamental mode to multiple input changes