ATG FOR SSFs IN SEQ. CIRCUITS TEST OF BRIDGING FAULTS

Weidong Li

Electronics System, Dept. of EE.
Linköping University, Sweden

Testing of Digital Systems, 2000-11-29

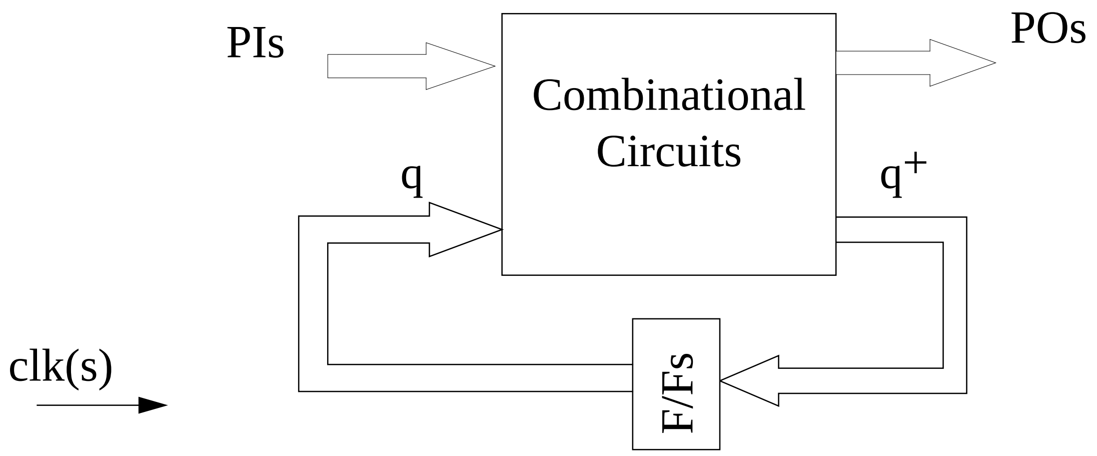
ATG FOR SSFs IN SEQUENTIAL CIRCUITS

- ✘ **Introduction**
- ✘ **TG using iterative array models**
- ✘ **Simulation-based TG**
- ✘ **Random TG**
- ✘ **Other TG methods**

Introduction

✘ **Sequential circuits:**

Synchronous seq. circuits and **asynchronous** seq. circuits



✘ **What make(s) differences?**

States: controllability (how to ensure state q) and observability (how to know state q+)

State for sure?

TG Using Iterative Array Models

✘ Basic instinct

State => Stateless

✘ Terminology

Iterative combinational array: Unrolling of syn. seq. circuits into combinational cell array.

* time domain ==> space domain

Time frame: each cell in iterative comb. array (both state and PIs at certain clk period).

✘ TG using iterative array models

Direct extension of from combinational circuits:

* Target Fault => Combinational test vector & Error propagation
* Combinational vector => PIs and State q (line justification)
* "Justify" state q (state initiation)
* Error propagation => POs and State q+
* POs => Bingo! State q+ => "propagate" q+ to POs

Known initial state => state generation from initial state ==> NP-problem

Unknown initial state => state learning (self-initialization) ==> NP-problem

weidongl@isy.liu.se
http://www.es.isy.liu.se/

TG Using Iterative Array Models (cont.)

✘ **General algorithm (Under assumption that clock line is fault-free)**

```
r=1
repeat
  Begin
    Build model with r time frames
    ignore the POs in the first r-1 frames
    ignore the q+ outputs in the last frame
    q(1)=given initial state
    if (test generation success) then return
        SUCCESS
    r=r+1
  end
until r=fmax
return FAILURE
```

Algorithm for known initial state

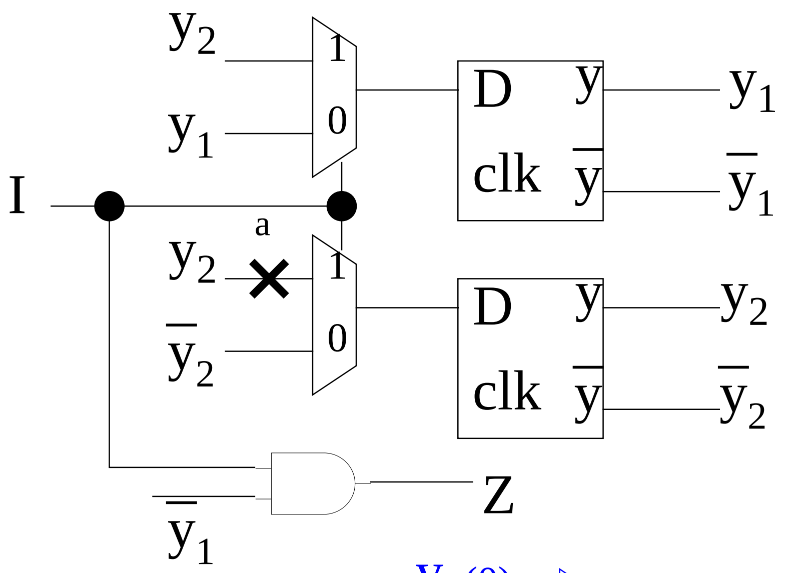
```
r=1
p=0
repeat
  begin
    build model with p+r time frames
    ignore the POs in the first p+r-1 frames
    ignore the q+ outputs in the last frame
    if (test generation success) and every q
        input in the first frame has value x)
        then return SUCCESS
    increment r or p
  end
until (r+p=fmax)
return FAILURE
```

Algorithm for unknown initial state

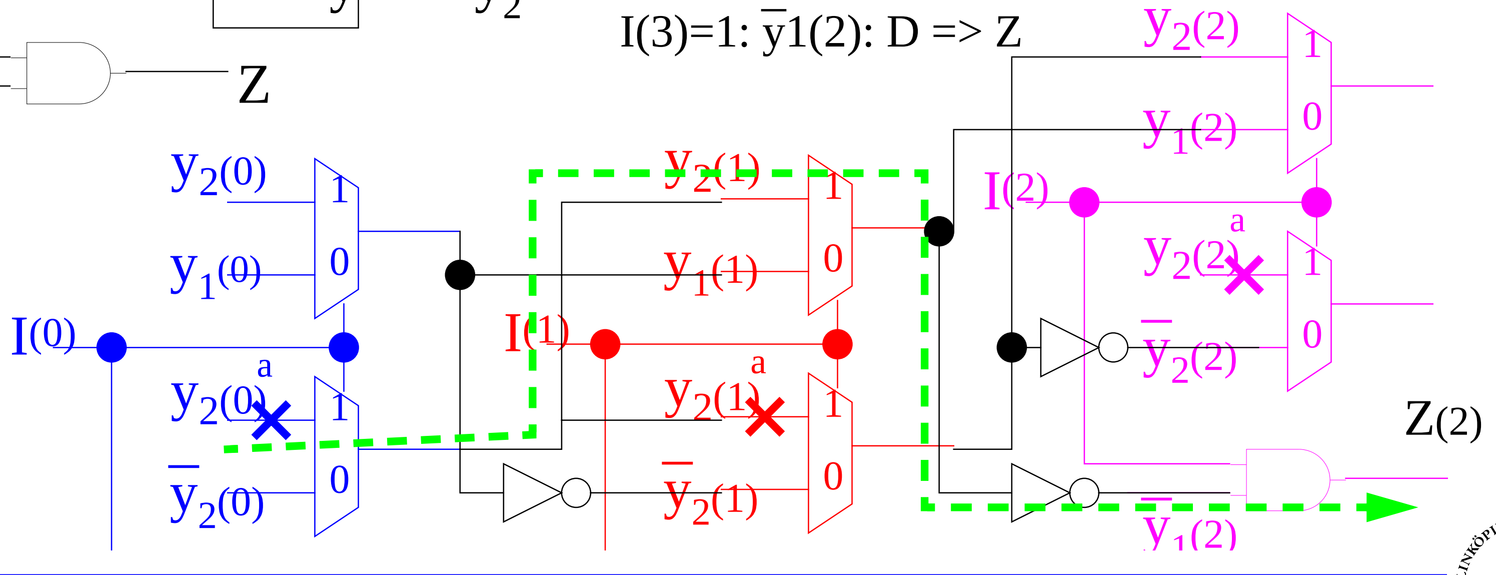
$f_{\max}$: maximum number of frames, determined by cost function.

TG Using Iterative Array Models (cont.)

Example 6.18:



line a: s-a-1 fault. Known state (0, 0).
Time frame 1:
Fault activation: I(1)=1.
Fault propagation ==> PO Z: y2=>y1=>Z
Time frame 2:
I(2)=1: y2(1) $\overline{D}$ => y1(2).
Time frame 3:
I(3)=1: $\overline{y}$1(2): D => Z



TG Using Iterative Array Models (cont.)

✘ **Drawbacks:**

* non a priori problem (unknown *r*/*p*)

* Discard of r-1 (r+p-1) computations => reuse

* Fault absorption could cause infinite loop => state monitor and/or $f_{\max}$.

● TG Using Iterative Array Models (cont.)

✘ **EBT (Extended BackTrace)**

* Don’t like r>0

* Increase only p

✘ **Critical-path TG**

* Start with a known state (reset)

* Start with fault activation, search the error propagation path towards POs

weidongl@isy.liu.se
http://www.es.isy.liu.se/

TG Using Iterative Array Models (cont.)

✘ Clock(s)

Fault at clock line => No state updating

Explicit clock line

Propagate both $1/q_i$ and $0/q_i$ fault effect to POs (?)

✘ Complexity issues

Relative to

* number of cycles
* number of F/Fs per cycle
* number of cycles a F/F is involved in

MUCH MORE COMPLICATED

Asynchronous circuits

- ✗ **Difficulties**

    Race problem

        Why it is less in synchronous circuits?

    Feedback identification

    Delay variation

- ✗ **Iterative models**

    PIs do not change until stable state is reached.

- ✗ **Discussion**

    “Global asynchronous, local synchronous” is a favour strategy for SoC. As observed from above, the asynchronous circuits are much difficult for testing. How to meet this challenge?

● Simulation-Based TG

✘ Principle

- ◆ Generate test vectors

  Random generate or non random generate

- ◆ Evaluate cost of test vectors according to the simulation
- ◆ Select test vectors to test sequence

✘ Cost function evaluation [Agrawal88]

- ◆ Initialization: number of F/Fs with unknown state.
- ◆ TG for group faults: summarize all cost value of activated faults.
- ◆ TG for individual faults: measurement from current state.

Heuristic optimization

● TG Using RTL Models

Know functionality of component, don’t care or unknown gate-level model.

✘ Extensions of *D*-Algorithm

- Keep overall structure, extend operations.
- Line-justification: find operation set to create fault.
- Implication based on component operators at RTL (state).
- Error-propagation: find operation set that propagate fault to output registers.
  In text book, the output registers are limited to data output (?).

● Advantage:

- Higher level ==> increase the efficiency
- IP-core
- Suitable for DFT methods like scan-test etc.

● Disadvantage:

- Limited to operations

TG Using RTL Models (cont.)

✘ Heuristic State-Space Search

SCIRTSS(Sequential CIRcuit Test Search System)

- Separation of datapath and control-part
- Utilize combinational TG to activate fault
- Find state set to propagate the fault involving heuristic function

$$H_n = G_n + wF_n$$

G_n: sequence length, w: weight, F_n: heuristic function.

F_n: fault proliferation function, or distance to goal node, or state trans.

Random TG

✘ Problem

- Initialization: initial state requires a deterministic sequences
- “Coloured lines: control lines are more weighted
- Evaluation

✘ Solution

- Initialization difficulty ==> semirandom
- “Coloured lines”==> nonuniform signal probability
- Evaluation ==> modify the goal function, take account of propagation

Other TG Methods

- **Hardware support for TG**
  - Advantage: concurrence
  - Disadvantage: reuse?
- **Artificial intelligence techniques**
  - HITEST: combines algorithmic procedures and user-provided knowledge

TESTING OF BRIDGING FAULTS

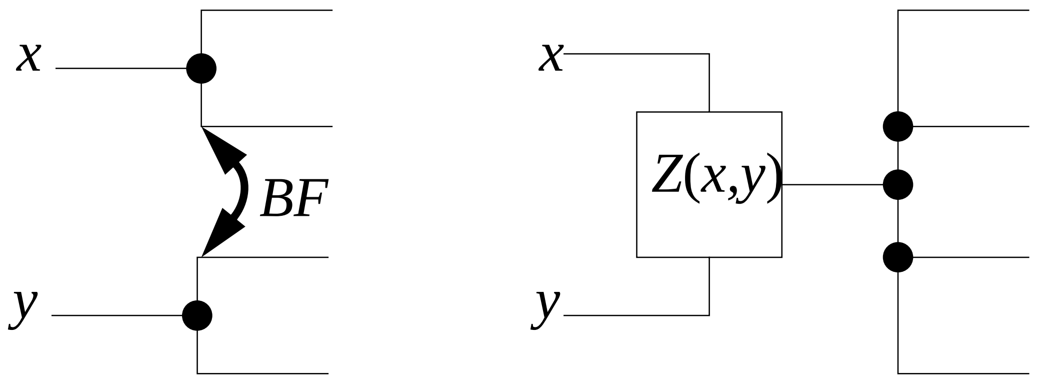
- ✘ Bridging-fault model
- ✘ Detection of NFBFs
- ✘ Detection of FBFs
- ✘ BFs simulation
- ✘ TG for BFs

Bridging-Fault Model

✘ **Bridging fault**

Faults are caused by shorts between two (or more) normally unconnected signal lines.

✘ **Bridging-fault model**



Logic function $Z(x, y)$

weidongl@isy.liu.se
http://www.es.isy.liu.se/

✘ BF model used in text book

- $Z(x, y) = x.y$ AND BFs
- $Z(x, y) = x+y$ OR BFs

✘ Multiple bridging-fault model

Multiple signal lines short

- AND/OR BFs?

Enough with this model? (Resistive BF model, Pattern dependent model etc.)

✘ Fault activation

Short signal lines must have different logic values.

Behave differently from proposed function.

Detection of NFBFs

AND NFBFs

- Non Feedback Bridging Faults: No combinational loop.
- Theorem 7.1: bridge between SSFs and BFs.

  A test detects the (*x*.*y*) iff either t detects *x* s-a-0 and sets *y*=0 or sets *x*=0 and *y* s-a-0.
- Corollary 7.1: Fanout-free x and y are inputs to the same OR/NOR gate, then the AND BF(*x*.*y*) dominates both *x* s-a-0 and *y* s-a-0.
- Corollary 7.2: Let x be a line with fanout and y a line without. If x and y are inputs to the same OR/NOR gate, then the AND BF(*x*.*y*) dominates *y* s-a-0.

Detection of FBFs

✘ AND FBFs

- Combinational logic ==> sequential logic

  (Loop ==> Oscillation possible)

- Theorem 7.2: A test *t* detects *f* s-a-0 and sets *b*=0 detects the AND FBF(*b*.*f*).

- Theorem 7.3: A test *t* detects *b* s-a-0 and sets *f*=0 without sensitizing *f* detects the AND FBF(*b*.*f*).

- Corollary 7.3: Let (*b*.*f*) be an AND FBF such that all paths between *b* and *f* have even inversion parity. A test *t* detects either *f* s-a-0 and sets *b*=0, or *b* s-a-0 and sets *f*=0, also detects (*b*.*f*).

- Corollary 7.4: Let (*b*.*f*) be an AND FBF such that any path between *b* and a primary output goes through *f*. A test *t* detects (*b*.*f*) iff *t* detects *f* s-a-0 and sets *b*=0.

Detection of FBFs(cont.)

✘ AND FBFs(cont.)

- Corollary 7.5: Let (*b*.*f*) be an AND FBF where *b* and *f* are such that *f*=1 whenever *b*=0(in the fault-free circuit). A test *t* detects (*b*.*f*) iff *t* detects *f* s-a-0 and sets *b*=0.

- Corollary 7.6: No signal test can detect an AND FBF (*b*.*f*) such that every path between *b* and a primary output goes through *f*, and *f*=0 whenever *b*=0.

Bridging Faults Simulation

- **Complexity (compare with SSFs)**
  - Structure and function
  - The number of feasible BFs
- **Implicit simulation method**
  - Based on relation between BFs and SSFs
  - Layout information available ==> Complexity reduction!
  - AND BFs: detects *x* s-a-0, find possible sets *y* *y*=0 in the neighbours, check propagation path.

Good fault coverage of SSFs ==> good fault coverage of BFs?

● TG for BFs

✘ **TG of BFs = SSFs + sets**

- ◆ Based of relationship of BFs and SSFs
- ◆ Generate TG for *x* s-a-v
- ◆ Justify y $y=\overline{v}$
- ◆ FBFs: find y $y=\overline{v}$ between *x*'s successors and implications

Of the same complexity order.

weidongl@isy.liu.se
http://www.es.isy.liu.se/