



# Fault Tolerant Computing

CS 530

## Testing Sequential Circuits

Yashwant K. Malaiya  
Colorado State University

# Why Testing Sequential Circuits is Hard

- **To test a sequential circuit we need to**
  - Initialize it into a known state (reset it: it's easy)
  - Apply a test sequence to it long enough to
    - Sensitize the fault
    - Propagate the error to output
    - Both can take many clock cycles
  - For  $n$  flip-flops, there are  $2^n$  states, but these can occur in different order, causing numerous possible sequences.
- **Common Approach: reduce the problem to that of combinational testing**

# Outline

- **Sequential circuits without feedback**
  - BIST
- **Sequential circuits with feedback**
  - DFT: scan
  - Extended D-algorithm (mention only)
  - Functional testing of FSMs
  - Initialization problem
- **Incremental testing for complex systems**

Note that FSM (finite state machine) testing concepts are applicable to both hardware sequential circuits and software that has “states”.

# Testing Sequential Circuits

- In general need to test for both
  - Functionality (examined here)
  - Timing (components too slow, too fast, not synchronized)
- Parts of a sequential circuit:
  - **Combinational logic:** faults: stuck 0/1, delay
  - **Flip-flops:** faults: input, output stuck 0/1, delay  
no latch 0/1 capability

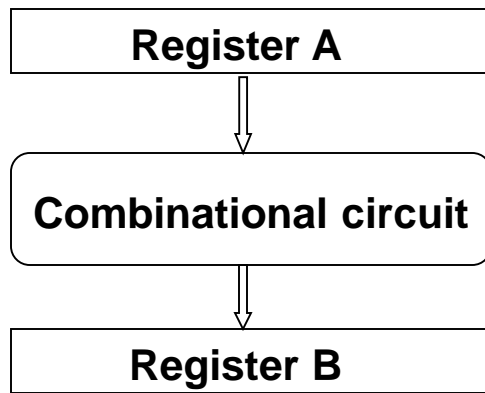
Malaiya, & Narayanaswamy, "Testing for Timing Faults in Synchronous Sequential Integrated Circuits," Intl. Test Conf., 1983.

Alassadi, Malaiya & Jayasumana, "Faulty Behavior of Storage Elements and its Effects on Sequential Circuits" IEEE Trans. VLSI Systems, Dec. 1993

# Two common cases that simplify testing

- **When the combinational logic is sandwiched between two registers, and there is no feedback**
  - Problem reduced to testing the combinational logic
  - Many faults in flip-flops are equivalent to faults in the combinational logic, thus faults in flip-flops can be ignored for simplification.
- **When there is feedback, but during testing feedback can be disabled when needed**
  - Scan approaches

# Testing Sequential circuits without feedback



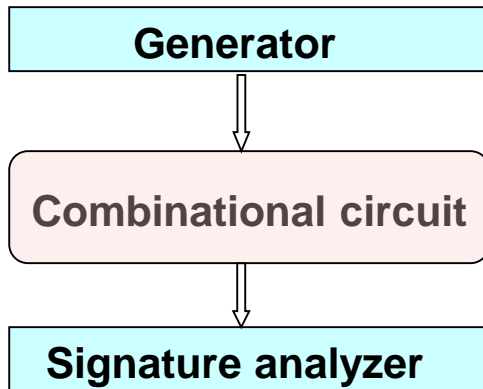
1. **Generate test vectors for the combinational circuit.**
2. **Place a vector in A**
3. **Examine response in B**
4. **Steps 2,3 can be done using**
  - a. **Machine instructions**
  - b. **Microinstructions**
  - c. **Specialized hardware**

# Built-in Self-Test Approach

- **Some registers can double as test pattern generators during testing.**
- **Some registers can also work as “signature analyzers” during testing**
  - **A Signature analyzer compresses a large number of output vectors into a single vector called “signature”.**
- **Well known designs for generators and signature analyzers are available.**

**The theory behind both generators and signature analyzers often involves “polynomial division” using “LFSRs (Linear feedback shift registers” , the same algorithm is used for calculating CRC in computer networks!**

# BIST (Built-in self-test)



- **Generator generates pseudorandom vectors.** Often an ALFSR.
- **Signature analyzer compresses all successive responses into a signature.** Usually an LFSR.
- **Compared with known good signature.**
- **Aliasing probability: prob. that a bad circuit can result in good signature.** Generally very small.

ALFSR: autonomous linear feedback shift register.

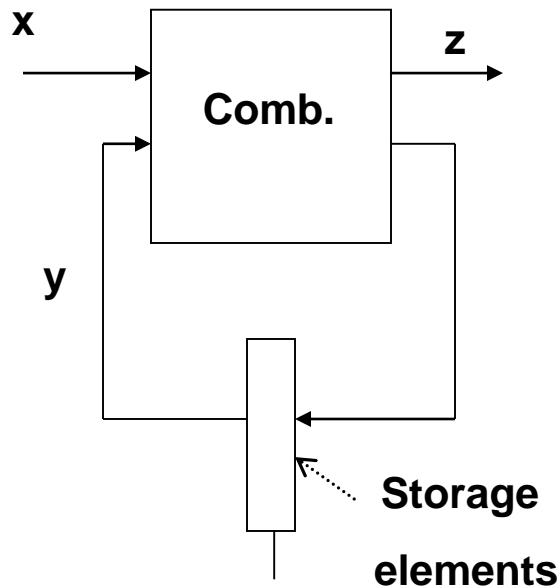
Better generators include our **antirandom** test generator.

# Sequential Circuits with feedback

Alexander cutting the Gordian Knot



# Sequential Circuits with feedback

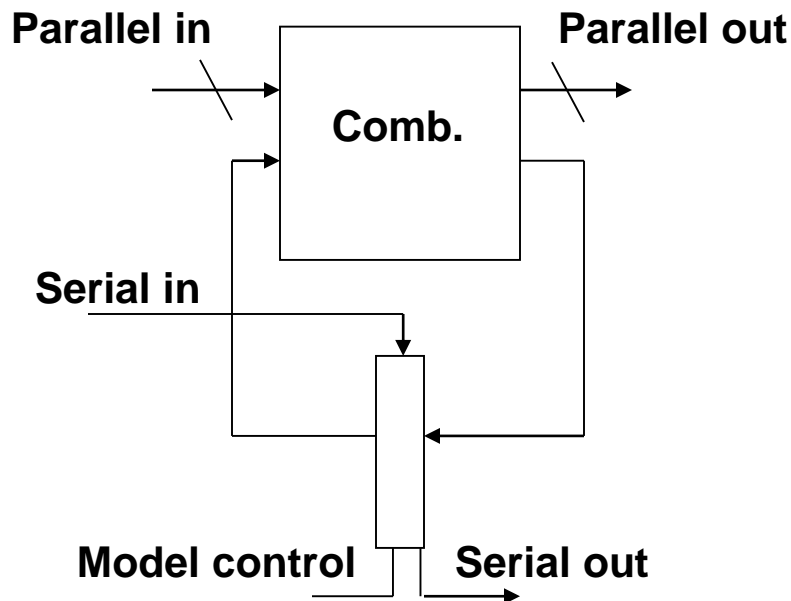


Some software programs are frequently tested as FSMs

- **Example: processor control unit**
  - Input: status, opcode
  - Output: control lines
- **Structural testing issues:**
  - How to obtain a desired  $(x,y)$  vector
  - How to propagate error
- **Functional testing issues:**
  - Prove all transitions/outputs are correct

# Sequential Circuits with feedback: Scan-chain approach

- **Design for testability:** feedback-less during testing. The flip-flops can be configured to form **scan-chains**.



- **Scan Design:** modes
  - **Normal mode:** parallel in/out
  - **Test mode:** serial in/out
- **Sequence of operations**
  - **Scan a vector:** test mode
  - **Latch response:** normal mode
  - **Scan response out:** test mode
- **If scan-chain too long**
  - **Use Multiple chain**
  - **Use Partial scan**

# D-Algorithm: Sequential Circuits

- You can use D-algorithm for sequential circuits, but
  - It can be cumbersome to generate tests
  - Test application time can be very long
- See literature if you are interested. For example  
[Test Generation for Sequential Circuits](#)

# Sequential Circuit Initialization Problem

- **At power-up the state is undefined.**
- **How to get FSM in a known state?**
  1. **Applying an input sequence: theoretical importance only**
  2. **Resetting to an initial state**
- **Resetting always used in practice**
  - using a reset/clear line
  - **Computers: power-on sequence to initialize PC, SP, interface registers etc.**

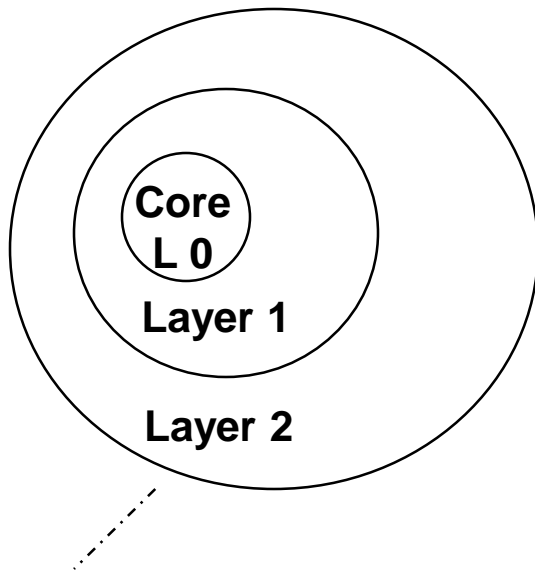
# Functional Testing of FSMs

- **Given:** state table **Objective:** confirm it is obeyed
- **Two alternative approaches:**
  1. **If the State is directly observable:**
    1. **Much easier**
    2. Use *Euler path*, if it exists, to minimize testing
  2. **Only outputs observable: “*Checking sequence*”**
    - a. Prove all states exist
    - b. For all inputs for each state, these are correct
      - Next state
      - Outputs
- **Theoretical, Complex and lengthy** (*see literature if interested*)

# Testing Complex Systems

- A Complex system includes several **components**
- Approach:
  - Assume a **fault model** for each *component*
  - Assume a system model that takes into account interaction of components
- Design a testing strategy such that these are adequately tested
  - each individual component
  - Interaction of components
- A “**component**” may be a
  - Physical component
  - Segment of the functionality

# Incremental Testing Approach



D Brahme, JA Abraham, Functional Testing of Microprocessors, IEEE Trans Comp, Jun 1984, pp. 475- 485.

- Partition system into **layers** such that layer  $i$  can be exercised using only layers  $0, \dots, i-1$ .
- Test components in each layer in the sequence  $L0, L1, \dots, Ln$ .
- Layering may require
  - Assumptions
  - Disabling feedback during testing
- Proofs of complete coverage can be constructed.
- Fault isolation can be done.

# Incremental testing: Example: Processor Based systems

## Sequence for testing:

- **Self-test processor:**
  - Basic instructions
  - Addressing modes
  - Complex instructions
- **Test Memory system and buses using processor**
- **Test I/O devices/ports**
- **Test peripheral devices**
- **Test software integrity**

# References

- See earlier references in previous Lecture Notes.
- D Brahme, JA Abraham, **Functional Testing of Microprocessors**, IEEE Trans Comp, Jun 1984, pp. 475- 485.
- V.D. Agrawal, C.R. Kime, K.K. Saluja, **A Tutorial on Built-in Self-Test Part 1: Principles**. IEEE Design & Test January 1993, Pages: 73 - 82 Part 2: Applications, IEEE Design & Test April 1993, pp. 69 - 77.
- C-C. Liaw, S. Y. Su, and Y. K. Malaiya. "**Test generation for delay faults using stuck-at-fault test set.**" Proc. of International Test Conf. 1980, pp. 167-175