# Computer and VLSI design

Introduction to system and circuit testing

P.J. Drongowski
SandSoftwareSound.net

## Testing goals

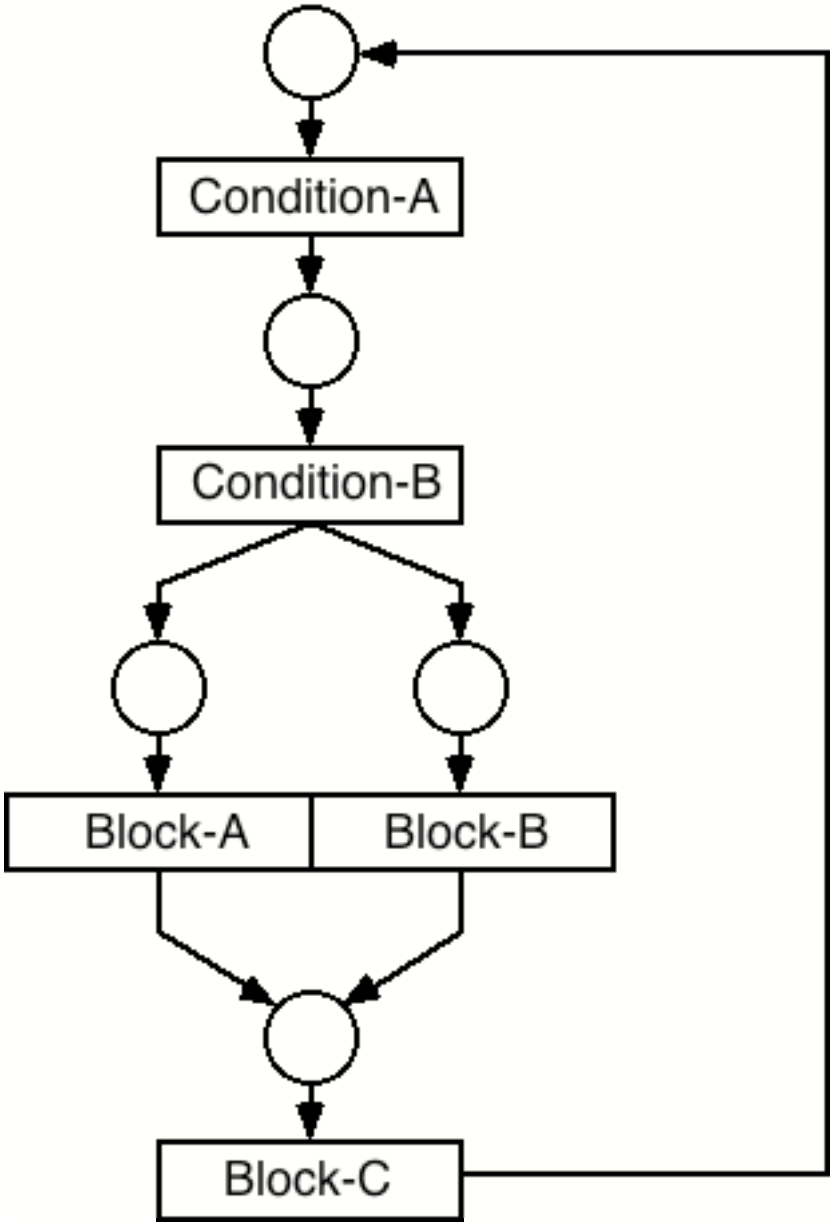
- Does the system meet the architects intent?
  - Engineer's concept versus implementation
  - Compatibility with specification
  - Does it run the operating system?
- Are there any defects?
  - Assumes no conceptual errors
  - Quality assurance-type testing
- Techniques
  - Formal modeling and verification
  - Diagnostic program
  - Built-in test hardware and routines
  - Symbolic testing and debugging

## Multilevel test strategy

- Online, operational testing
  - Detect and report "hard" and "soft" errors
  - Keep overhead low
- Diagnostic mode testing
  - Off-line, direct test of hardware features
  - Often used for field maintenance
  - Isolate fault to replaceable unit (chip or board)
- Hardware self-test
  - Check out hardware at operational speed
  - May use special microcode, BIT hardware
  - May require external test engine, fixture, etc.
- Quality assurance testing
  - Electrical / parametric testing
    - Performance over supply voltage range
    - Performance over temperature range
    - Voltage level under load
    - Timing / delay testing
  - Functional testing

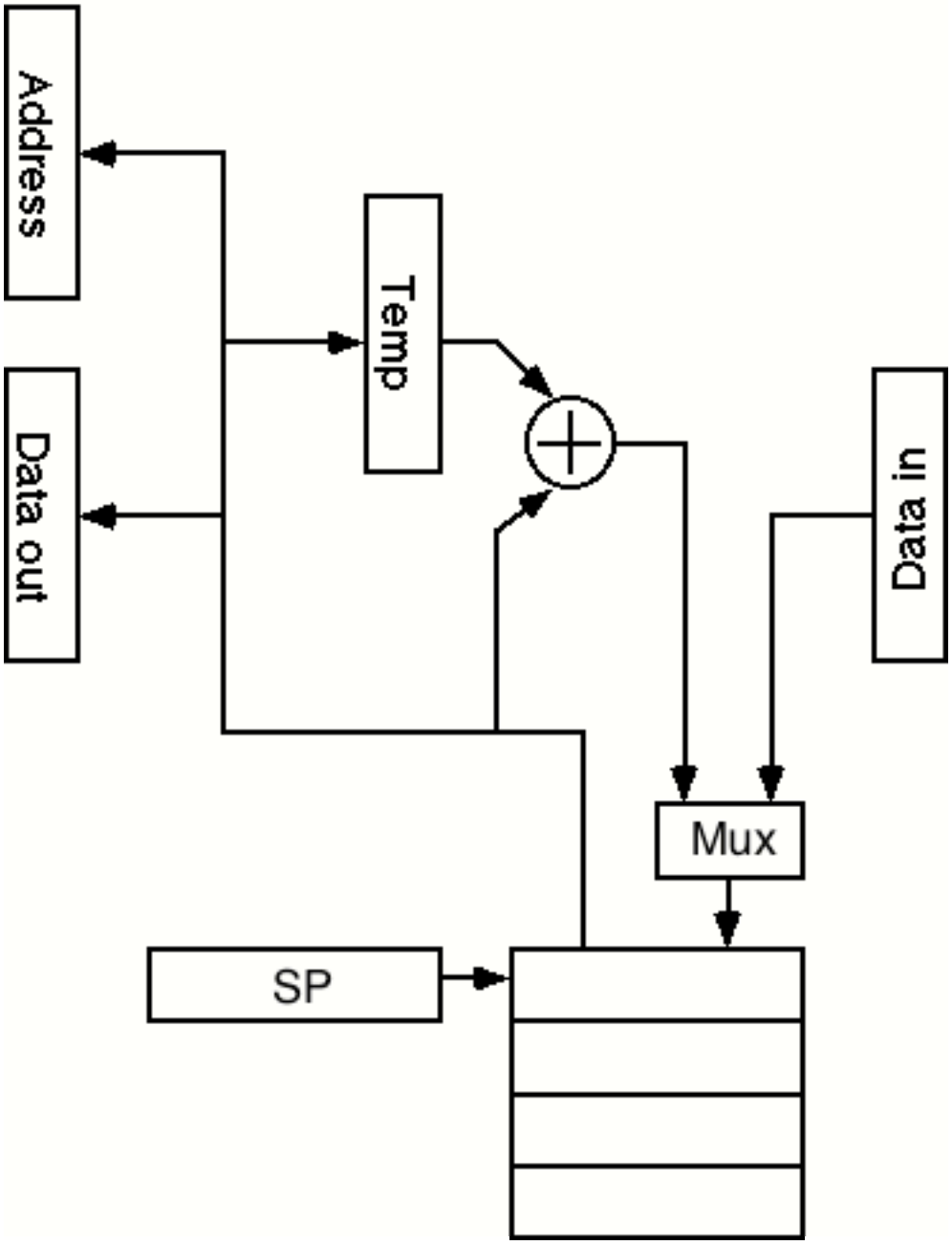
# Software path testing

```
while ( Condition-A )
     {
     if ( Condition-B )
          {
          Block-A
          }
     else {
          Block-B
          }
     Block-C
     }
```



# Hardware path testing

- Use structure to construct diagnostics
- Exercise each path
- Find s-a-0 and s-a-1 faults on path
- Exercise control flows (in microcode)

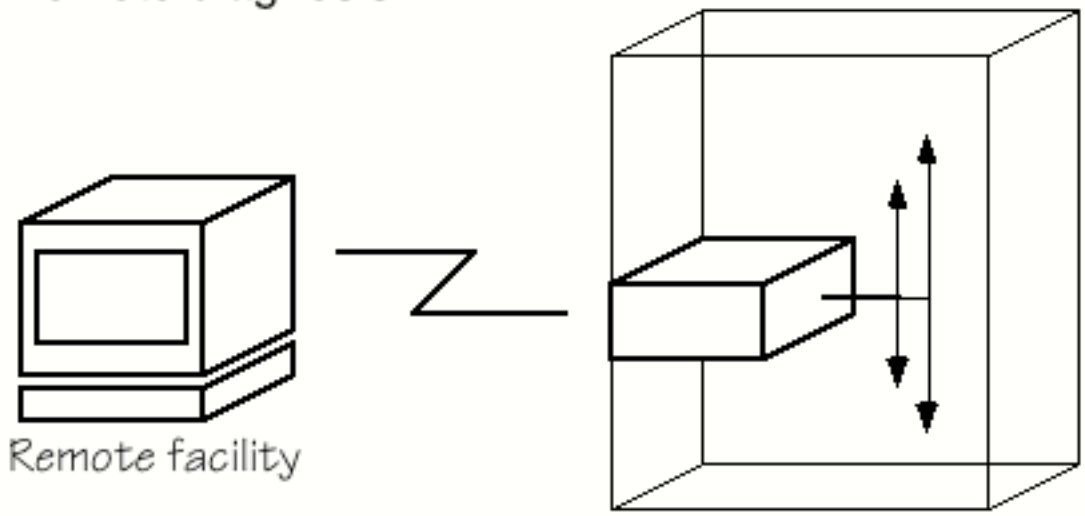


# Diagnostic programs

- Test ISA, I/O subsystem and hardware
- Good for built-in, start-up self test
- Exploit system structure if possible
- Procedure
  - Start with a few fundamental instructions
  - Progress to complex instructions
  - Test processor conditions and special cases
- Problems
  - Selective test values versus exhaustive
  - Control flow testing (branches and calls)
  - Exception conditions
  - Interrupt and trap sequences
- Example: Serial interface port
  - Loop send lines into receive lines
  - Is data being transmitted correctly?
  - Try different character size, stop/start, parity
  - Are interrupts received and vectored?
- Example: Primary memory
  - Failures
    - Pattern sensitivity (errors between adjacent cells)
    - Data loss (cannot hold data between refresh)
    - Addressing (errors in address decoding logic)
    - Multiple writes (writing to cell changes others)
    - Sense amplifier (slow recovery)
  - Simple patterns
    - Marching 0's & 1's
    - Checkerboard)
  - Walking patterns
  - Galloping patterns
  - Surround disturb patterns

# Diagnosis and repair

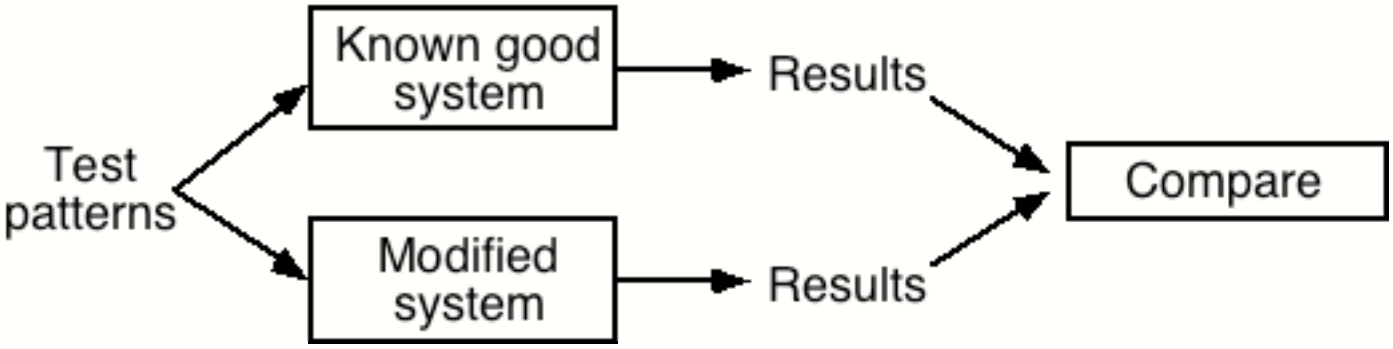
- Support for field maintenance
- Not only detect fault, but isolate to replaceable unit
- Repair / replacement strategy
  - Subsystem, board, chip
  - Must keep inventory of replacement units
  - Field service staffing, instrumentation and overhead
  - Cost of repair versus pricing (per call, contract, etc.)
  - Time to failure (MTTF)
  - Time to repair (MTTR)
- Built-in diagnosis
- Remote diagnosis




- Diagnostic processor
  - Loads microcode
  - Performs health and status monitoring
  - Act as off-line test processor (apply patterns, etc.)
  - Communication link to remote central service facility
  - Switch in redundant, back-up resources if necessary
  - Diagnosis, dispatch, repair
- Suitable for higher cost, high availability products

# Regression testing

- Maintenance: Messing with a working system
- Have changes introduced new bugs into the system?
- Method
  - Construct/model the system
  - Construct test patterns
  - Save patterns in a file (or design database)
  - Apply patterns to working (known good) system
  - Record test results in file (or design database)
  - After a change, apply patterns and acquire new results
  - Compare new test data against old results
- Problems
  - Thousands of test patterns/results
    - Automate comparison
    - Unix `diff` program
  - System structure may change
    - May require new test patterns
    - Incremental chages to test experiments



## Faults and failure modes

- Stuck-at-zero (s-a-0)
- Stuck-at-one (s-a-1)
- Single faults
- Multiple faults
- Transient failures
- Non-stuck-at faults
- Fault masking
- Sequential faults

## Test / experiments

- Fault detection - "Is there a fault?"
- Fault isolation - "Where is the fault?

## Test signals

- Primary inputs - "Inputs that can be driven"
- Primary outputs - "Outputs that can be sensed"

## Test activities

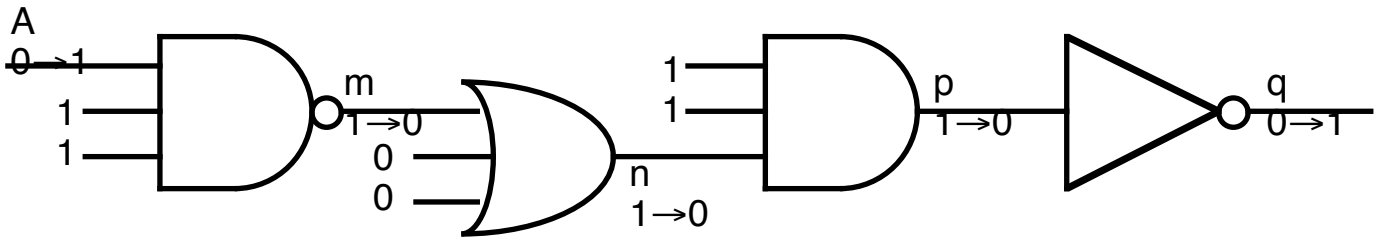
- Test generation
- Test evaluation
- Test application

# Test generation

- Objectives
  - Exercise the circuit to reveal or identify faults
  - Detect and isolate faults as fast as possible
  - Find minimum set of patterns for adequate test
- Factors
  - Controllability - primary inputs / probes
  - Observability - primary outputs / probes
  - Not all nodes are controllable or observable
- Problems
  - Ambiguous, incomplete or incorrect specification
  - Volume of test data
    - Example: 10,000 gates
    - Two states per gate: 20,000 experiments
  - Failure modes
    - Fault doesn't fit stuck-at model (e.g., CMOS)
    - Multiple faults
- Generation techniques
  - Exhaustive
    - Most complete testing
    - Takes too long to be practical
  - Random
    - Generate patterns at random
    - May not detect all faults (defects)
  - Path sensitization
    - Use knowledge of structure to generate patterns
    - Doesn't work for all gate networks
    - Inconsistent inputs prevent set-up, propagation
  - D-algorithm (J.P. Roth, 1966)
    - 3 tables of cubes: Primitive, failure, propagation
    - Cube intersection: check logic consistency
    - Pattern generation process for particular fault
      - Select failure D cube for the fault
      - Drive fault forward using propagation cubes
      - Determine inputs by implication (justification)

# Path sensitization

- Consider the example below (Kohavi, pg. 210)
- Test for s-a-1 fault at input A




- Suppose this is the only path from A to the primary output
- Procedure to test for s-a-1 at A
  - Apply a zero to input A
  - Apply all one's to remaining inputs of AND (NAND) gates
  - Apply all zero's to remaining inputs of OR (NOR) gates
- Procedure allows propagation of test signals
- This procedure will also test:
  - s-a-0 faults at m, n and p
  - s-a-1 fault at q
- Follow complementary procedure for s-a-0 at A

# Observations

- A set of tests which sensitize a set of paths containing all connections must only detect faults at primary inputs
- Successful if each output is connected to only one input
- Circuit structure is a "tree" in that case
- If an output is shared, the technique may not work
- Propagation values may conflict with "test" values
- Such pathological circuits are known
- Should try to sensitize several paths in one test for speed

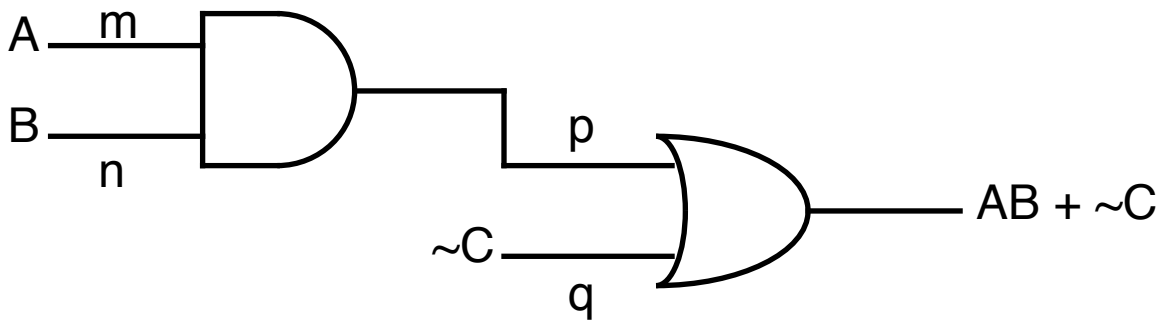
# Fault detection

- See "Switching and finite automata theory, Zvi Kohavi, McGraw-Hill.
- Faults
  - Apply test input values
  - Look for unexpected (erroneous) output values
- Exhaustive testing
  - Apply all possible input combinations
  - Too time consuming to be practical
- Typical fault model
  - Stuck-at-0 and stuck-at-1
  - Single faults
  - Loop-free combinational logic
- Goal: Try to minimize test time with acceptable coverage

# Fault table

- One row for every possible test (input combination)
- One column for every fault
- Mark entry if fault can be detected by input combination
  - Apply inputs (as represented by that row)
  - Fault is detectable if the output of the faulted circuit differs from the correct output value
- Find the minimal set of rows so that every column has at least one mark
- Such a set of inputs *covers* the fault table
- Problem is identical to prime implicant covering

# Fault table example




| ABC | $m_0$ | $n_0$ | $p_0$ | $q_0$ | $m_1$ | $n_1$ | $p_1$ | $q_1$ |
|---|---|---|---|---|---|---|---|---|
| 000 | | | | X | | | | |
| 001 | | | | | | | X | X |
| 010 | | | | X | | | | |
| 011 | | | | | X | | X | X |
| 100 | | | | X | | | | |
| 101 | | | | | | X | X | X |
| 110 | | | | | | | | |
| 111 | X | X | X | | | | | |

- To determine if output is s-a-0, try to switch it to one
- To determine if output is s-a-1, try to switch it to zero
- Minimal set = { 000, 011, 101, 111 }
- It may not be possible to isolate a fault
  - Consider input 111, for example
  - m s-a-0, n s-a-0 or p s-a-0 can be source of error

# Test evaluation

- Evaulate test quality
- Fault coverage
  - Grading
  - Percentage of all possible faults that will be detected by test patterns
- Alternative definitions
  - Structural
    - Possible fault: connection s-a-0 or s-a-1
    - Percentage of connections driven 1 and 0
  - Digital Equipment Corporation
    - Possible fault: transistor doesn't switch
    - Percentage of transistors switched on and off
  - Behavioral
    - No structural information
    - Possible fault: branch not taken
    - Percentage of control paths exercised
    - Or percentage of operations tested
- Fault insertion experiments
  - Determine fault coverage
  - Known good device (KGD)
  - Known good board (KGB)
  - Basic method
    - Apply patterns to KGD and save correct results
    - Insert faults in KGD
    - Apply patterns to faulty device
    - Compare output with correct results
    - Compute percentage detected
- Logic fault simulator
  - Takes the place of KGD or KGB
  - Incremental (interactive testing / grading)
- COSMOS
  - Compiled simulator for MOS circuits
  - Symbolic test pattern generation and fault simulation

# Test application

- Apply tests to real circuit
- Automatic test equipment (ATE)
- Stored pattern tester
  - Develop test patterns and determine coverage
  - Transfer patterns and KGD results to ATE station
  - Apply patterns and acquire test data
  - Compare with KGD and report faults
  - Programmability comes at a very high cost
- Comparison tester
  - Compare circuit with "golden" device (KGD)
  - Report differences as faults
  - Low cost: $20,000 to $100,000
  - Depend upon bug- and defect-free KGD
- Maximum test application rates
  - Tester must operate at very high speed
  - Hard to test high performance parts at speed
  - Higher speed means higher test throughput
- Analog electronics
  - Must measure voltages and currents
  - Different failure modes
  - Must test digital inputs over specified range
- Pulses and timing
  - Tester must catch pulses
  - Need to measure pulse width, delay, etc.
- Physical access (probes)
  - Need to connect probes to primary inputs / outputs
  - Connections to internal test points
  - Wafer probe - pre-packaging QA tests
  - Automated IC handler
    - Feeder rails to test site
    - Clamp device for testing
    - ATE directs device to good / reject output rails
  - PCB test fixtures

# Test fixtures

- Zero insertion force (ZIF) socket
  - Insert in socket, throw lever, make contact
  - Manual insertion, slow
- Wafer probe
  - Very fine wires to bonding pads
  - Put pads in standard locations (frame)
  - Only signals at pads can be mechanically probed
  - Can use E-beam, but not practical on large scale
- Bed of nails
  - PCB test fixture
  - Probe internal test points
  - Must plan ahead for physical access to test points