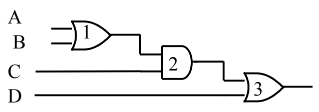
The Concept of a *Fault*

- Testing centers around detection of *faults* in a circuit.
- The digital world is made up of interconnected gates
  - Thus, only two things can fail - gates and their interconnections
- A faulty gate fails to perform its function correctly
- A faulty interconnection produces a “stuck at 1”, “stuck at 0”, or permanently tri-stated input.
- Problem: can you set up experiments that produce one result if the gate/interconnect is good, and another if it is bad?
- This leads to two closely related concepts.

Observability and Controllability

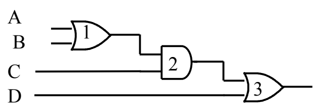
- Observability is the ability to observe a gate output directly at external pins
- Controllability is the ability to apply any and all desired inputs to a gate via external pins
- Test a gate completely, all combinations of inputs must be applied and the output corresponding to each input must be observed

Controllability: An Example



- How do you control the inputs to gate 2? Gate 3?
  - Gate 2: Set input A to 1 and apply four possible input combinations to inputs B, C (Sensitize Gate 1 to changes on B).
  - Gate 3: Set input A or B to 1 and apply all four possible input combinations to C, D (Sensitize Gate 2 to changes on C)

Observability: An Example



- How do you observe gate 1’s output? Gate 2?
  - Gate 1: Set input C to 1 and D to 0 (Set Gates 2 and 3 so that they pass output A)
  - Gate 2: Set input D to 0

Extending the Concept

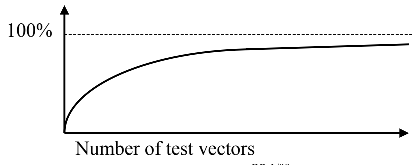
- A set of inputs and outputs designed to test a specific gate is called a *test vector*.
- A collection of inputs designed to test a specific gate or area completely is called a *test vector suite*.
- Once a test vector suite is developed for a block, you need to develop a way to get those inputs to the block and directly observer the block’s outputs

Some Problems

- #1: Four additional test vectors are required for every additional 2-input gate. The number of test vectors increases as the number of gates increases.
- #2: The number of gates is increasing much faster than the number of inputs/outputs. This is making both controllability and observability more difficult.
- #3: Sequential circuits can be notoriously difficult to test. For example, a 16-bit counter has to cycled through its entire count for complete testing.
- Some strategies have emerged to deal with these problems.

Automated TestVector Generation

- Software exists to analyze a digital design and perform the following functions:
  - Answer the question "What percentage of gates have been completely tested by the test vectors applied to the device?" This is known as *Fault Coverage*.
  - Generate test vectors to cover the remaining gates
- Fault Coverage curves look like:



Issues in Test Vector Generation

- Issue #1: Diminishing Returns -- Vectors start covering smaller and smaller fractions of the device
- Issues #2: Some tests may be nearly impossible or truly impossible (non-testable faults)
- Without extra hardware added explicitly to help with testing, fault coverage is typically in the 90& to 95% range.

Built-In Self Test (BIST)

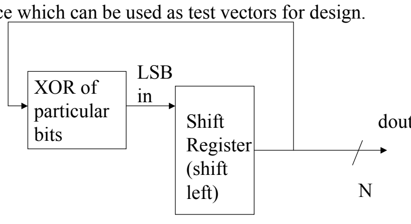
- One way to add reliablity to a component or part of a chip is with Built-In Self Test (BIST)
- An example is the Motorola 68030 MCU
  - Motorola has added a small set of instructions which they certify tests a significant fraction of the 68030's gates.
  - Can be incorporated into the low level startup process to detect and processor faults.

BIST Approach

- Add circuitry so that on powerup a small ROM or pseudo-random sequence generator applies a series of inputs
- The outputs are tested for a correct answer, usually by accumulating the outputs into a checksum value (either by XOR or Cyclic Redundancy Check - CRC)
- The psuedo random pattern generation and checksum accumulation does not need many gates
- The probability of passing self-test with an undiagnosed fault can be made small (depends on number of patterns).

Linear Feedback Shift Register (LFSR)

A Linear Feedback Shift Register can be used a pusedo-random sequence which can be used as test vectors for design.



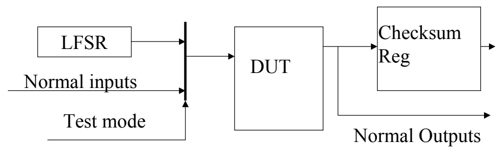
For a 16 bit register,

LSB = dout(1) xor dout(2) xor dout(4) xor dout(15)

LFSR (cont)

- For an N-bit shift register, the maximum length sequence that can be generated is 2^N-1 (sequence will repeat after this number of shifts).
  - For 8 bits, will generate 255 out of the possible 256 values.
- Assuming a reset value of '0' for the shift register, the missing code will be 255 for the following feedback combinations:
  - Lsb = 1 xor b(1) xor b(2) xor b(3) xor b(7)
  - Lsb = 1 xor b(3) xor b(4) xor b(5) xor b(7)
  - Lsb = 1 xor b(2) xor b(4) xor b(6) xor b(7)
- Each of the above feedback arrangements produces a different sequence.

Built in Self Test (BIST)

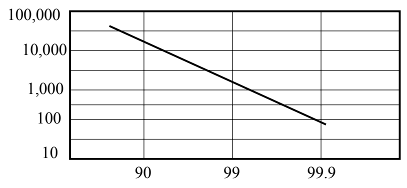


DUT = device under test

When in self test, inputs to DUT is taken from LFSR and checksum register accumulates the output. After some X number of test vectors have been applied, compare check sum output against a golden value – if the same, then system has passed self test.

Is Fault Coverage Sufficient Without Extra Hardware?

- Consider study results that show defect level vs. fault coverage

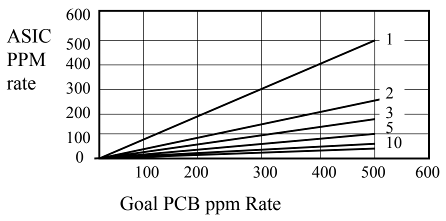


Source: 1980 study by Delco and Motorola, as cited in TI's IEEE 1149.1 Testability Primer

To get a single chip defect rate of 100 defective parts per million parts, you need 99.9% fault coverage! Do you really need a defect rate of less than 100 ppm?

The Effects of Single Chip Defects

- Consider the effects of having multiple chips on a board



Source: TI's IEEE 1149.1 Testability Primer

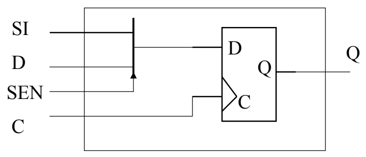
If you have 5 ASIC's on a board, you need to have around 60 ppm defective ASIC's to achieve 300 ppm defective boards.

What does all this mean?

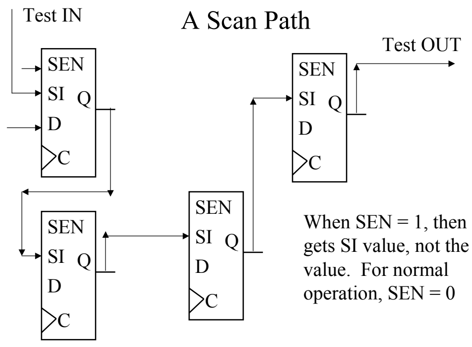
- You need fault coverage in the 99% to 99.9% range
- You need automated test vector generation and fault grading tools
  - Fault grading tells you % fault coverage for a test vector suite
- You need extra hardware to improve testability.

A Solution: Scan Paths

- Is there a way to apply inputs directly to gates and observer outputs directly?
- Yes - with a *Scan Path*. A Scan Path ties all FFs in your design into a shift register.
  - Requires Scannable DFFs



A Scan Path

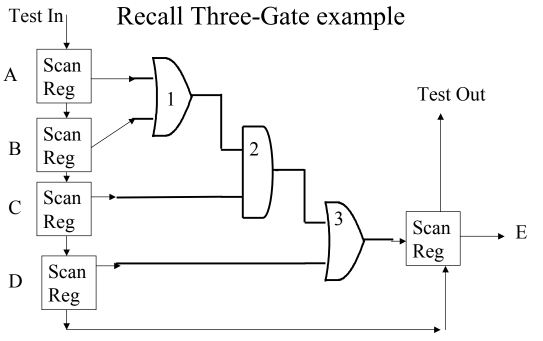


When SEN = 1, then DFF gets SI value, not the D value. For normal operation, SEN = 0

Scan Path Operation

- Assume ALL FFs in design are in tied into a Scan Path and let there be N FFs.
- Apply SEN = ‘1’, and clock test vector serially into scan path. Will take N clocks.
- Apply, SEN = ‘0’, and clock ONCE. This will test the design using the test vector!
- Apply SEN = ‘1’, and clock in next test vector. At same time, you clocking OUT the result of the last test vector!!!
- After first test vector, each test vector takes N+1 clocks!!

Recall Three-Gate example



A,B,C,D,E do not have to externally accessible!! Could be buried deep within the design (and usually they are!)

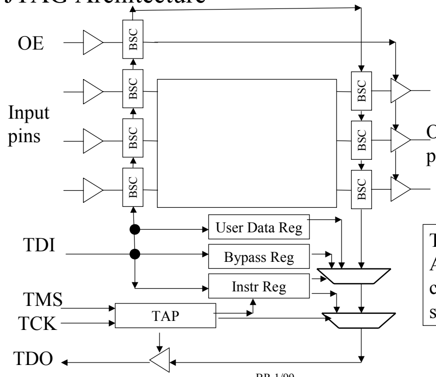
Introducing IEEE 1149.1 (JTAG)

- In order to standardize scan path design and support, the Joint Test Action Group, comprised of over 200 electronics companies, developed an industry-wide standard for scan path support
- It has been adopted as IEEE Standard 1149.1
- It is usually called “the JTAG port” in sales literature (or Boundary Scan)
- Every JTAG compatible port can be chained together to form a serial device
- Can be used to support fault dection at the single chip level, detection of board level faults, and even some PLDs/FPGAs can be programmed with it.

Basics of JTAG

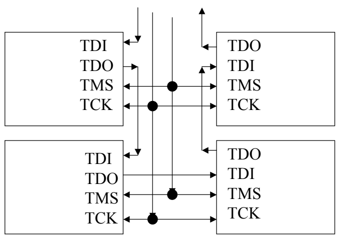
- The JTAG standard adds four signals to every compatible device
  - TCK: a clock signal for the test port
  - TMS: Test Mode Select - determines whether or not the scan path is active and controls the mode it is in.
  - TDI: Test Data In - serial data input
  - TDO: Test Data Out - serial data output
- The standard also specifies some standard operating modes
- For example, the the serial shift register can be set to bypass the chip, making access to other chips faster.
- Vendor specific commands be added.

JTAG Architecture



TAP = Test Access Port, control for serial bus.

JTAG for Board Level Fault Diagnosis



Only four external pins needed.

The JTAG port can drive outputs and capture inputs, board connections between pins can be tested!

Boundary Scan can be used to either stimulate ASICs or simply test board connections between chips.

Some Keys to an Effective Test Strategy

- Decide early, not late, on test strategy
  - Other test approaches beside BIST, JTAG
- Identify software tools for test vector generation, fault grading
- Have to plan test strategy through development, prototyping, manufacturing, and maintenance
- Adding a JTAG bus to a board (and JTAG compatible components) can be a cost effective way of adding a large amount of testability to your design

Summary

- The purpose of testing is to uncover faults in a system (chip or board)
- Faults are found by applying test vectors and observing results
- Test vectors can be generated manually or automatically
- Usually have to add extra hardware to improve testability.
- Scan paths can increase chip and board level testability.
- JTAG adds four pins and supports a variety of test modes; almost all electronics vendors support JTAG.