ECE382M.20: System-on-Chip (SoC) Design

Lecture 17 – SoC Testing

Sources:
Jacob A. Abraham

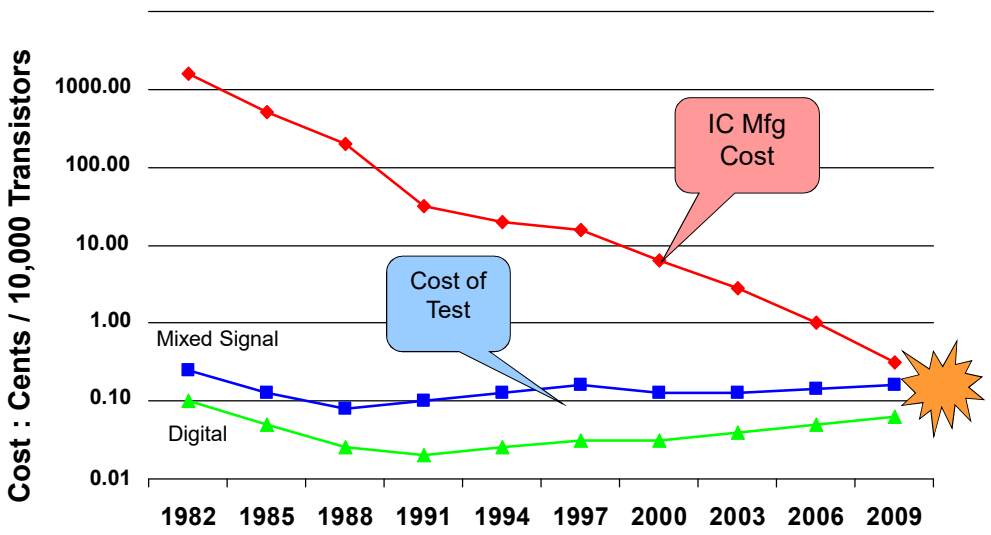
Andreas Gerstlauer
Electrical and Computer Engineering
University of Texas at Austin
gerstl@ece.utexas.edu


The University of Texas at Austin
Chandra Department of Electrical and Computer Engineering
Cockrell School of Engineering

Outline

- **SoC Manufacturing Test**
  - The testing problem
  - SoC testing costs
  - Design for Test (DFT)
- **SoC Testability Features**
  - Boundary Scan
  - P1500 standard
- **Built-In Self Test**
  - Functional Test Access Mechanism (TAM)

The Manufacturing Test Problem

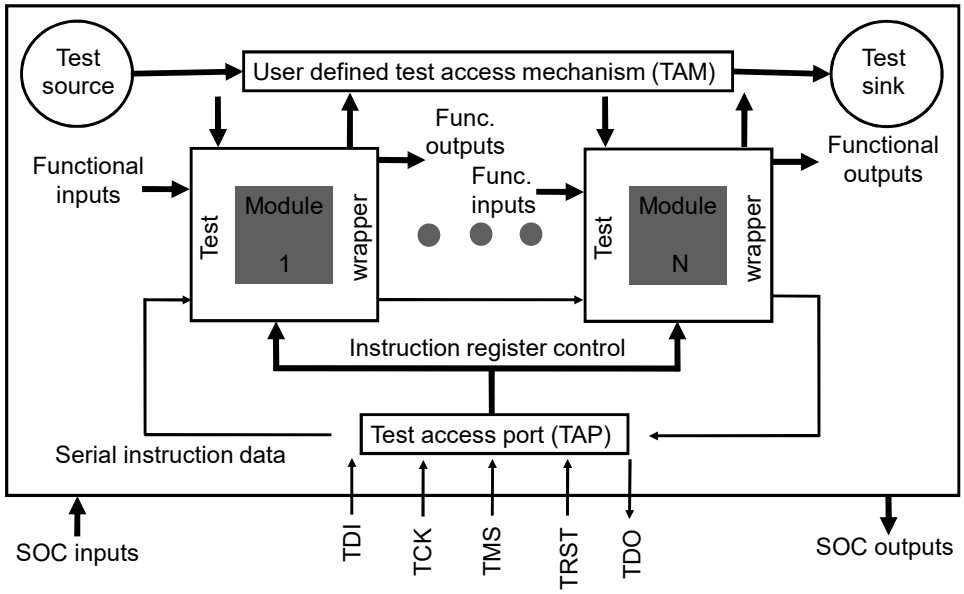


Partitioning for SoC Test

- **Partition according to test methodology:**
  - Logic blocks
  - Memory blocks
  - Analog blocks
- **Provide test access:**
  - Boundary scan
  - Analog test bus
- **Provide test-wrappers for cores**

➢ **Design for Test (DFT)**

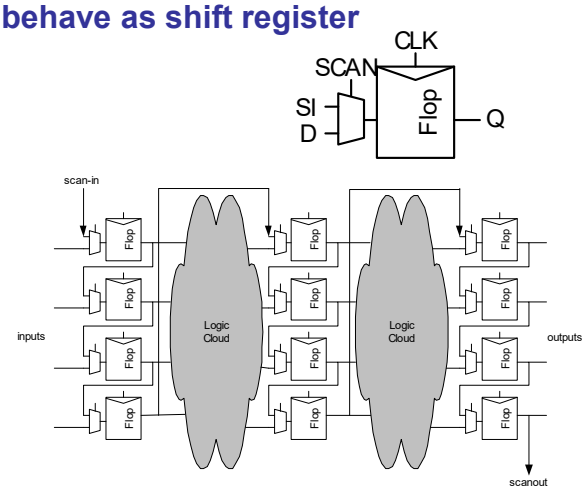
DFT Architecture for SOC



Source: Bushnell and Agrawal

Scan

- **Convert each flip-flop to a scan register**
  - Only costs one extra multiplexer
- **Normal mode: flip-flops behave as usual**
- **Scan mode: flip-flops behave as shift register**
  - Contents of flops can be scanned out and new values scanned in

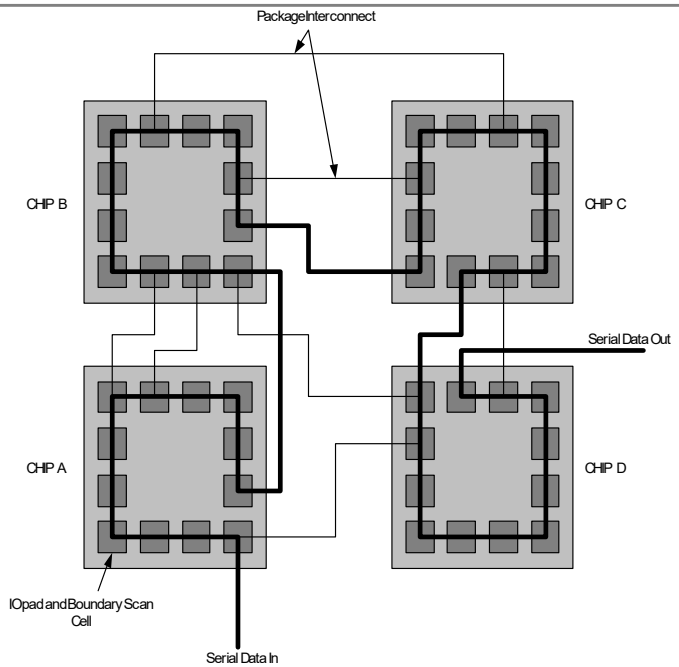


Boundary Scan

- **Testing boards is also difficult**
  - Need to verify solder joints are good
    - Drive a pin to 0, then to 1
    - Check that all connected pins get the values
- **Through-hold boards used "bed of nails"**
- **SMT and BGA boards cannot easily contact pins**

➢ **Build capability of observing and controlling pins into each chip to make board test easier**

Boundary Scan Example



Boundary Scan (IEEE 1149.1, JTAG)

- **Boundary scan is accessed through five pins**
  - TCK: test clock
  - TMS: test mode select
  - TDI: test data in
  - TDO: test data out
  - TRST*: test reset (optional)
- **Chips with internal scan chains can access the chains through boundary scan for unified test strategy.**

Additional DFT Components

- **Test source: Provides test vectors via on-chip LFSR, counter, ROM, or off-chip ATE.**
- **Test sink: Provides output verification using on-chip signature analyzer, or off-chip ATE.**
- **Test access mechanism (TAM): User-defined test data communication structure; carries test signals from source to module, and module to sink; tests module interconnects via test-wrappers; TAM may contain bus, boundary-scan and analog test bus components.**
- **Test controller: Boundary-scan test access port (TAP); receives control signals from outside; serially loads test instructions in test-wrappers.**

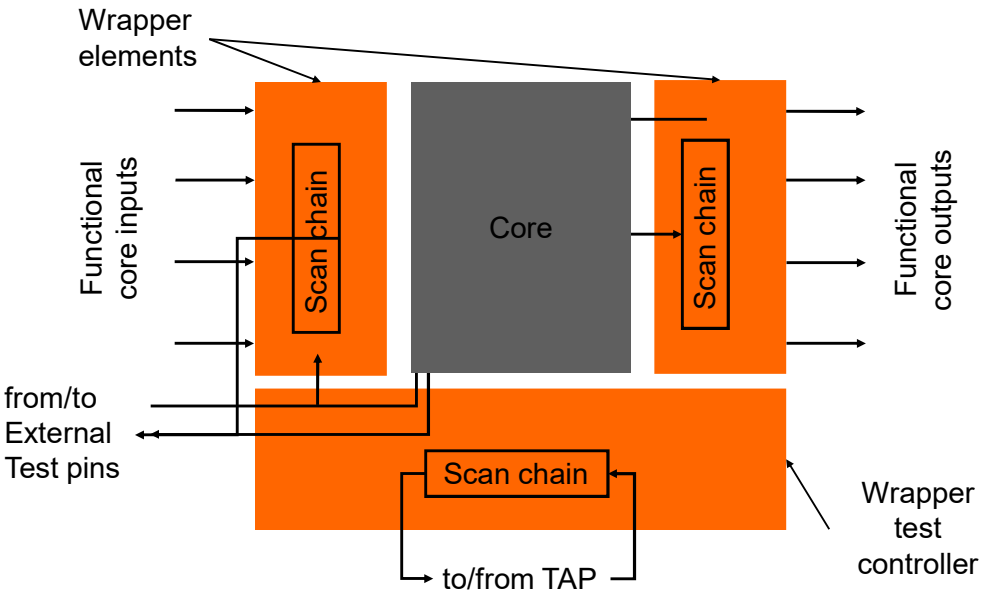
Source: H. Kerkhoff

Test Wrapper for a Core

- **Logic added around a core *to provide test access* to the embedded core**
- **Test-wrapper provides for each core input terminal**
  - An external test mode – Wrapper element observes core input terminal for *interconnect* test
  - An internal test mode – Wrapper element controls state of core input terminal for testing the logic *inside core*
- **For each core output terminal**
  - A normal mode – Host chip driven by core terminal
  - An external test mode – Host chip is driven by wrapper element for *interconnect* test
  - An internal test mode – Wrapper element *observes core outputs* for core test

Source: H. Kerkhoff

A Test-Wrapper



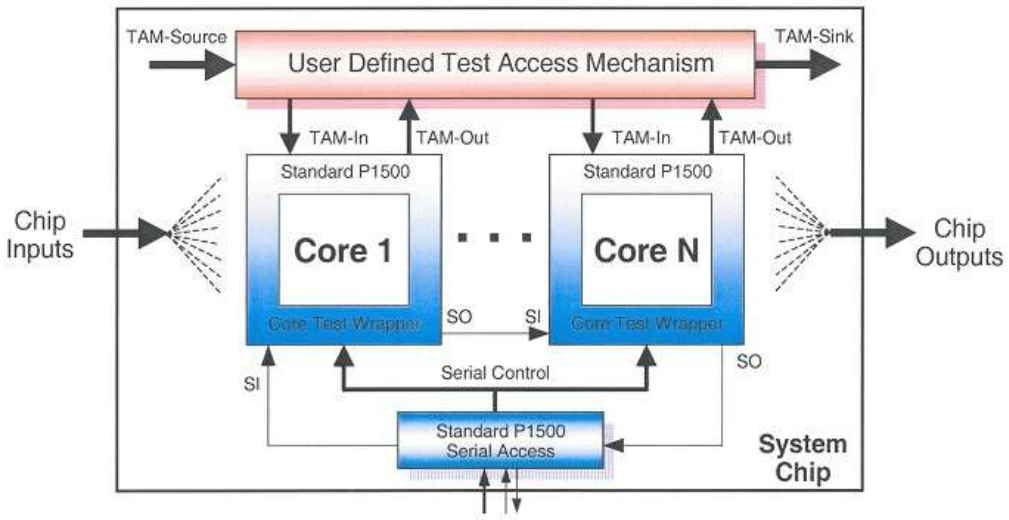
Source: H. Kerkhoff

Goals of IEEE P1500

- Core test *interface* between embedded core and system chip
- Test *reuse* for embedded cores
- Testability guarantee for system interconnect and logic
- Improve efficiency of test between core users and core providers

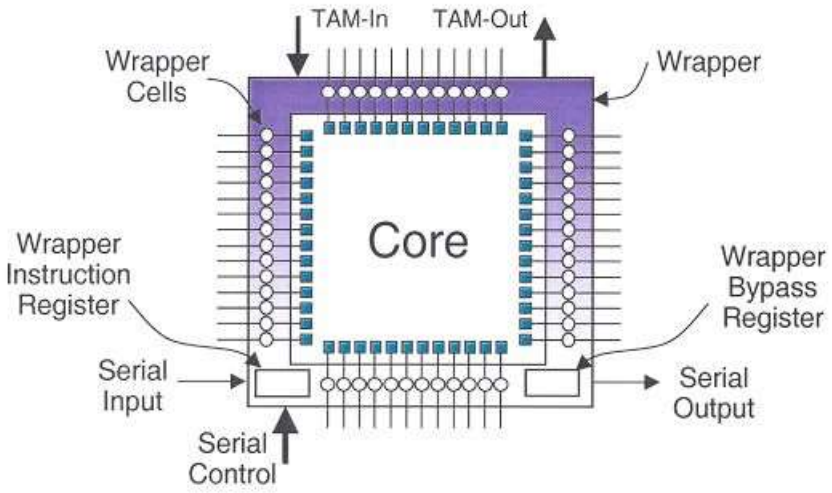
Source: H. Kerkhoff

Set-up of P1500 Architecture



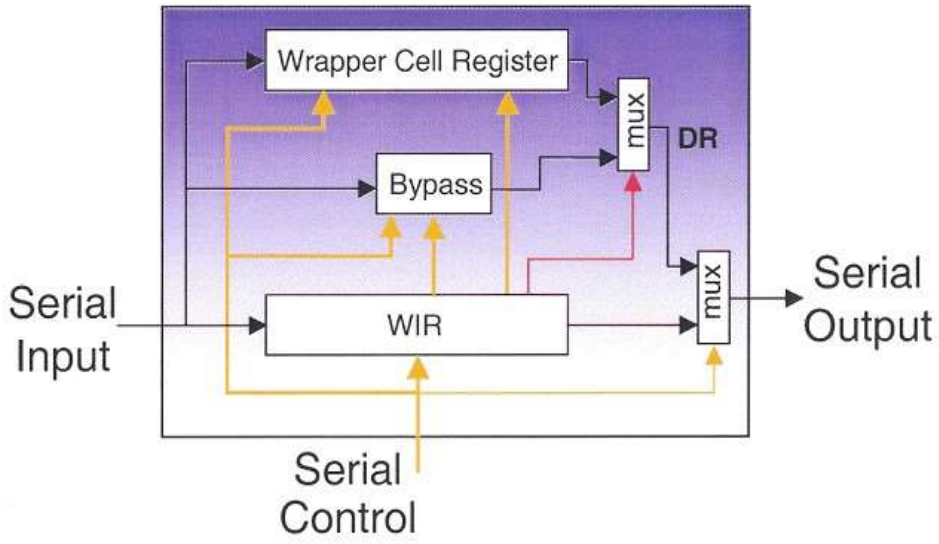
Source: H. Kerkhoff

Core including Wrapper Cells



Source: H. Kerkhoff

Wrapper Registers for P1500



Source: H. Kerkhoff

Outline

- ✓ SoC Manufacturing Test
  - ✓ The testing problem
  - ✓ SoC testing costs
  - ✓ Design for test (DFT)
- ✓ SoC Testability Features
  - ✓ Boundary Scan
  - ✓ P1500 standard
- **Built-In Self Test**
  - Functional Test Access Mechanism (TAM)

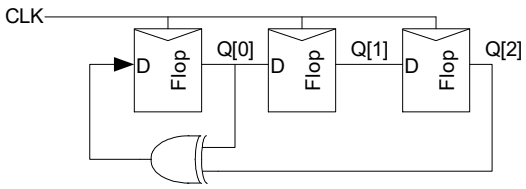
Built-In Self Test (BIST)

- **Increasing circuit complexity, tester cost**
  - Interest in techniques which integrate some tester capabilities on the chip
  - Reduce tester costs
- **Approach:**
  - **Pseudo-random (or pseudo-exhaustive) pattern generator (PRPG)** on the chip
  - Compress test responses into "**signature**" using **multi-input shift register (MISR)**

➢ **Integrating pattern generation and response evaluation on chip – BIST**

Pseudo-Random Sequences

- ***Linear Feedback Shift Register***
  - Shift register with input taken from XOR of state
  - *Pseudo-Random Sequence Generator*

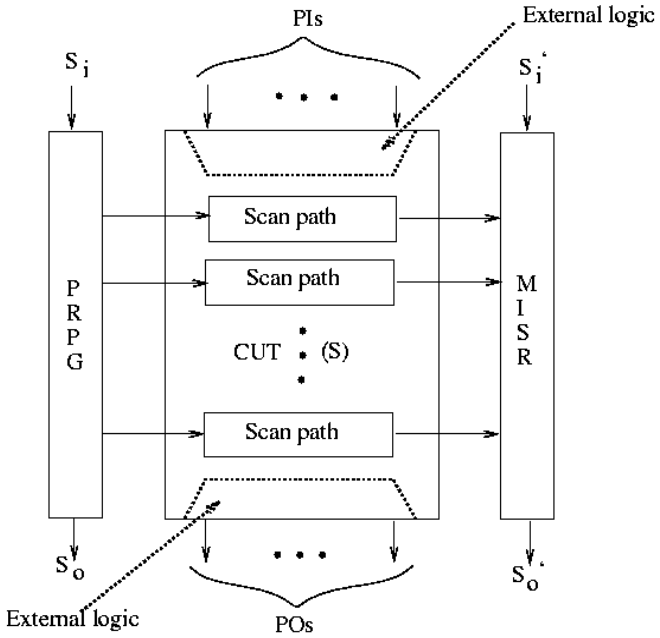


Step	Q
0	111
1	110
2	101
3	010
4	100
5	001
6	011
7	111

(repeats)

Can also be used to **compress** test responses

Example of BIST



Technique called **STUMPS** (from IBM)

Test Access Mechanisms (TAMs)

- **Non-functional access**
  - Uses a kind of access to core not allowed during the normal functional operation
  - Generally based on scan chains or other design for test (DFT) structures → Slow serial access
  - Direct access to core test pins through external pins → Fast, but high pin overhead
  - Can also use the embedded processor as the test source/sink → Needs wrappers around the core under test
- **Functional access**
  - Embedded processor is the test source/sink → No DFT structures or wrappers around the cores

Functional TAMs

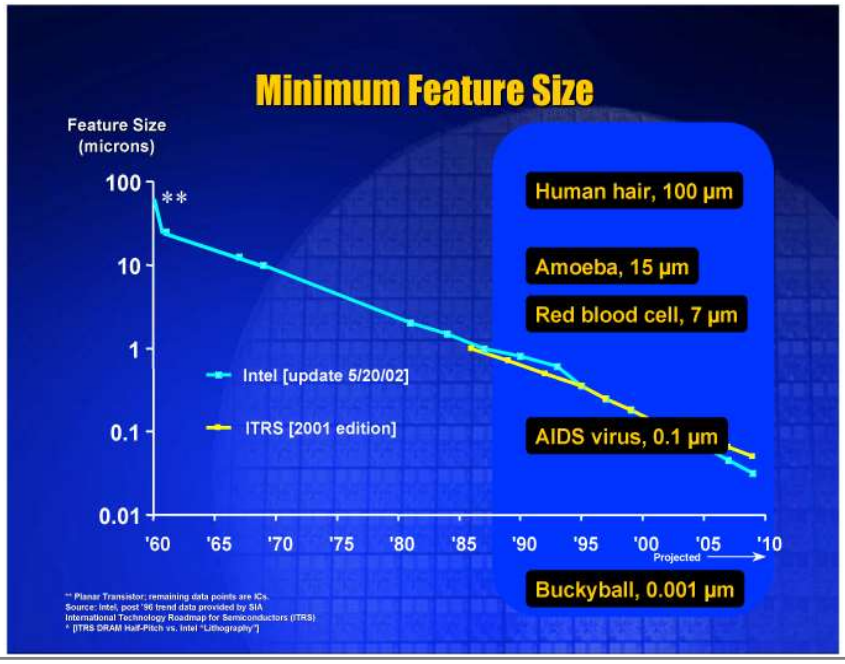
- **Software-Based Self Test (SBST): Use the intelligence of the embedded processor to test the SOC**
- **At-speed tests are possible**
- **Cores in the SOC can be of three kinds**
  1. White box -- internals visible, structure changeable
  2. Grey box – all the internals visible, but structure of the core cannot be changed
  3. Black box – no internals visible, no change can be made on the core
- **Any methodology for testing black box cores should not depend on knowledge of the core's internals**

Why is Conventional Test Successful?

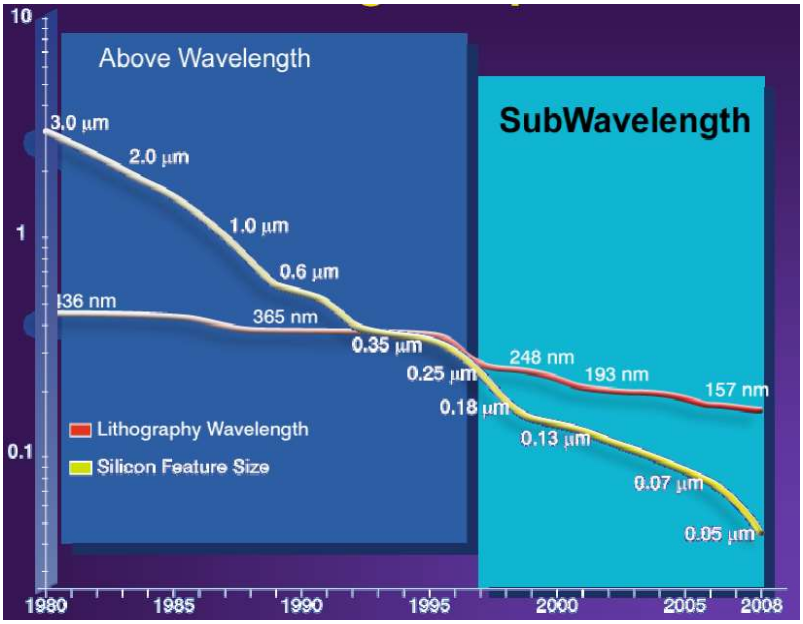
- **Two innovations have allowed test to keep up with complex designs**
  - The stuck-at fault model
    - The model allows structural test generation, with a number of faults which is linear in the size of the circuit
  - Partitioning the circuit
    - Partitioning the circuit (with scan latches for example), alleviates the test problem so that test generation does not have to deal with the entire circuit

➢ **Do these two assumptions hold for Deep SubMicron (DSM) circuits?**

IC Technology

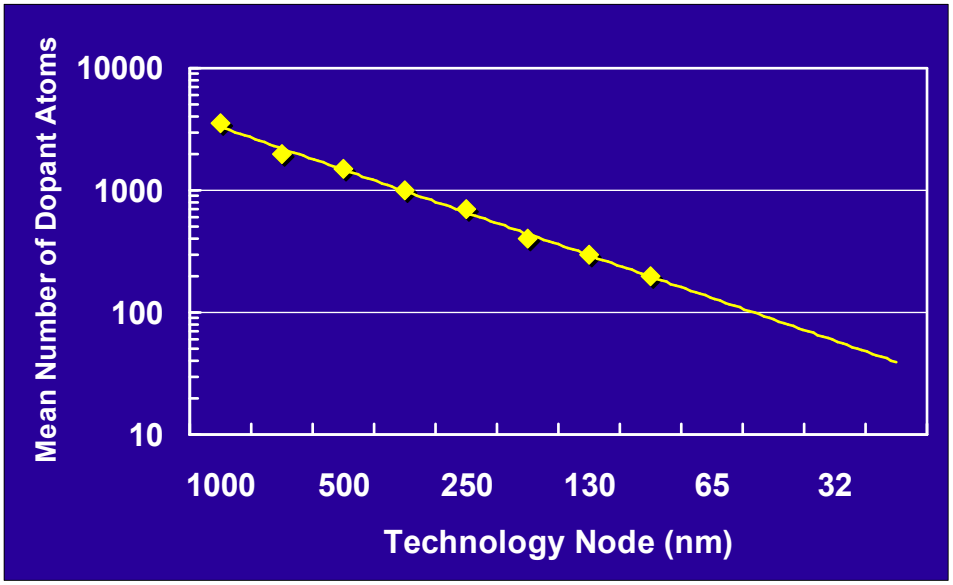


Features Smaller than Wavelengths



Source: Raul Camposano, Synopsys

Random Dopant Fluctuations

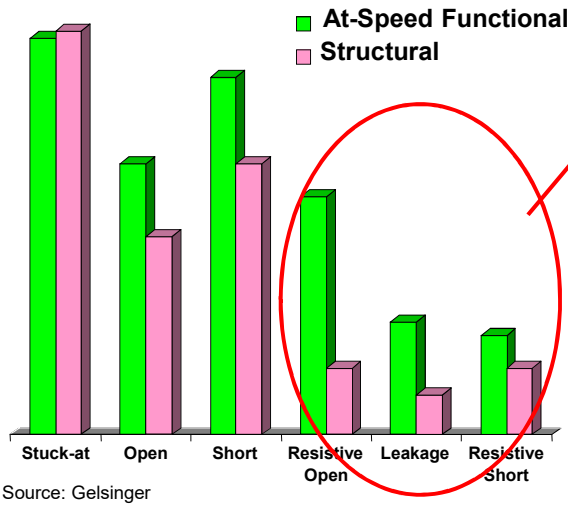


Defects in DSM Technologies

- **Experiments on real chips (e.g., Stanford)**
  - Stuck-at tests do not detect some defects unless they are applied at speed
- **Resistive opens comprise the bulk of test escapes in one production line**
  - Likely in copper interconnect – cause delay faults
- **Delay faults identified as the cause of most test escapes on another line**
  - Speed differences of up to a factor of 1.5 can exist between fast and slow devices - problems with "speed binning"
- **Increasing possibility of *shorts* and *crosstalk***

Effects on Chip?

- **Change in delays of paths**
- **Effects could be distributed across paths**



Source: Gelsinger

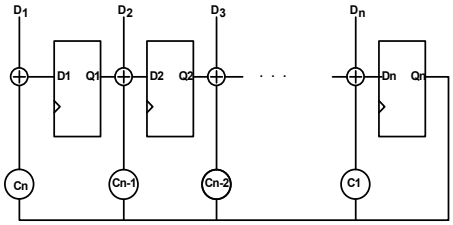
At-speed functional tests better for delay defects

Solution:

- **At-Speed tests**
  - Tester Cost?
- **Apply "Native Mode"?**
  - Can use low-cost testers

Native-Mode Built-In Self Test

- **Functional capabilities of processors can be used to replace BIST hardware – [UT Austin, ITC'1998]**
  - Application to self-test of processors at Intel – FRITS method applied to Pentium 4, Itanium [ITC'2002]



Hardware for MISR

```
for each data value Di {
   Shift_Right_Through_Carry(S);
   if (Carry) S = XOR(S, polynomial);
   S = XOR(S, Di);
}
```

Software implementation of MISR

Native-Mode Self Test for Processors

- **Random instructions can be run from cache and results compressed into a signature**
- **Implementation in Intel FRITS system showed benefits for real chips (Pentium 4, Itanium)**
- **Technique can be used for self-test of an embedded processor in a System-on-Chip**

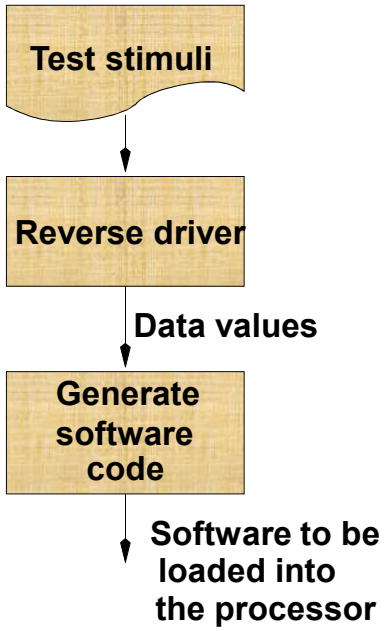
➢ **Is it possible to now use this processing capability to test other modules (digital, analog/mixed-signal and RF) on the SoC?**
  - First, can the processor test be improved to detect realistic defects, e.g., small delays?

Are Random Tests Sufficient?

- **Intel implementation involved code in the cache which generated random instruction sequences**
- **Interest in generating instructions targeting faults**
  - Possible to generate instruction sequences which will test for an internal stuck-at fault in a module [Gurumurthy, Vasudevan and Abraham, ITC 2006]
- **In order to deal with defects in DSM technologies, need to target small delay defects**
  - Recent work: automatically generate instruction sequences which will target small delay defects in an internal module [Gurumurthy, Vemu, Abraham and Saab, European Test Symposium (ETS) 2007]

Approach to Testing Cores

- **Uses functional TAM**
- **Uses pre-existing vectors**
- **Generates software to be loaded on to the embedded processor**
  - Reverse driver that produces given test vectors for core



[Gurumurthy, Sambamurthy and Abraham, Int'l Test Synthesis Workshop (ITSW) 2008]

Pre-Existing Vectors

- **If using a core bought from vendor**
  - Vectors might also be provided by the vendor
- **Reusing a core**
  - Vectors from the previous use
- **Newly designed core**
  - Validation vectors
- **Only constraint: these vectors must be functional test patterns for the core**

Reverse Driver

- **Parses the vector sequence to generate the data set to be sent to the core being tested**
- **Is specific to each core – as many as the number of driver programs**
- **Only overhead involved**
- **Generates the output in a format readable by the driver program**

Reverse Driver Illustration

- **Peripheral core communicating with external environment (send/receive 32-bit data)**
- **Five 8-bit registers addresses 0 – 4**
  - Register 0 – Control
  - Registers 1 to 4 – Data

Address	Data
0x00	0x07
0x01	0x54
0x02	0xDF
0x03	0x71
0x04	0x78

Reverse Driver → Send at speed rate 1 Data 0x0754DF7178

Software Generation

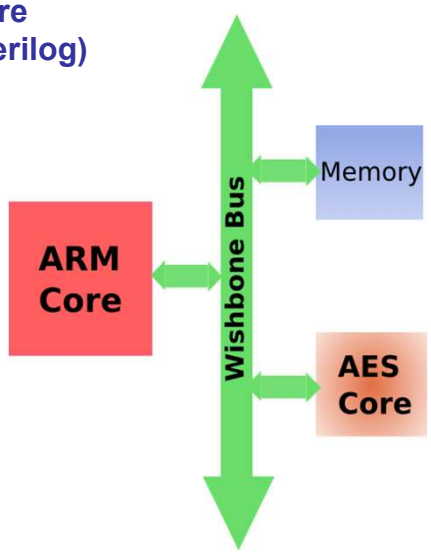
- **Use the driver program associated with each core being tested**
- **Driver programs**
  - Software code that actually talks with the non-processor cores
  - Know about the bus protocol
  - Generally able to take in the data to be sent to the core or read back data from the core
  - Developed as part of designing the SOC

Coverage Measurement

- **Simulate the SOC using the software generated**
  - Platform used SOC validation can be used
- **Monitor the core boundaries to capture the pin data**
- **Fault simulate the core with the captured data**

Experiment

- **Implemented a SOC containing ARM core, AES cryptographic core and a Wishbone bus interface (Verilog)**
  - AES 128-bit data/key encryption/decryption from www.opencores.org



Validation vectors:
Set of random values encrypted and decrypted

Results

Details about the synthesized AES core

No. of inputs	69
No. of outputs	33
No. of sequential elements	9225
No. of combinational elements	1119
No. of stuck-at faults	64070

Results

	Size (bytes)	Fault coverage	Original coverage	No. of cycles	Original cycles
Test1	7808	90.01	90.26	6700	6373
Test2	9128	90.15	90.35	7816	7435
Test3	10432	90.20	90.44	8932	8496

Summary

- **SoC Manufacturing Test**
  - Scan chains
  - JTAG Boundary Scan
  - Test wrappers for cores
  - Built-in self test (BIST)
- **Advanced SoC test topics**
  - Analog/Mixed-Signal (AMS) test
  - RF test
  - Micro-Electro-Mechanical Systems (MEMS) test