Issues in Microprocessor Testing

- Today’s microprocessors consist of m/billions of transistors operating at extraordinarily high speeds
- Large number of registers
- Large number of small buffers or queues
- Different sizes of memories
- Complex random logic (control path & datapath)
- Board level testing
- Test integration & scheduling
- CAD tool support

Part 1: Systems with Microprocessors

- A system is an organization of components (hardware/software parts and subsystems) with capability to perform useful functions
- Systems with a microprocessor can use it to implement testing strategies for the whole system. The uP can self-test itself too
- Functional testing: verifies integrity of system
  - 1. Without fault models (heuristic): simply exercise the functions of the system
     - Check existence and responsiveness of subsystems
     - Check system specifications
     - Check/execute selected critical functions of the system
  - 2. Using specific fault models, that attempt to represent the effect of physical faults on the operation of the functionally modeled system
     - Explicit model: should define a reasonably small fault universe
     - Implicit model: identifies classes of faults with “similar” properties

Part 2: Microprocessors Testing

- Structural testing
  - Faults defined in conjunction with a structural model: structural fault models. Main types of structural faults are shorts and opens and they are mapped into stuck-at and bridging faults
  - Test generation methods are based on the structural model of a system under test: produce tests for structural faults. Examples: PODEM, CONTEST, etc.
  - Test generation difficulty increases with the increase of processor complexity. Addressed partially by DFT techniques.
- Functional testing
  - Functional fault model at Register Transfer Level (RTL): represent the effect of physical faults on the operation of a functionally modeled system. Example: addressing fault affecting register-decoding
  - Difficult to automate

Functional Testing

- Functional testing reduces the complexity of the test generation problem by approaching it at higher levels of abstraction → higher efficiency in test time
- The process of test generation is difficult to automate. It is often a manual process – time consuming, prone to errors
- The applicability of a functional testing method is limited to systems described via a particular modeling technique
- 1. Testing without a fault model
   - Quality of the functional tests is unknown
   - Typically does not check that unintended operations do not occur (e.g., in addition to a correct transfer of data into register R1, the presence of a fault may cause the same data to be written into register R2)
- 2. Using specific fault models
   - We do not know the comprehensiveness of a functional fault model → functional fault coverage is not meaningful

1. Without fault models

- Develop test programs that can be executed on the processor
- Method:
  - Test each instruction
  - Test each subunit such as ALU
  - Test buses, register file and decoders
  - Test sequencing of instructions
- Key idea: Start small — test components and instructions that are easy to test and then use the tested parts to test other parts
2. Using specific fault models

- Graph model for microprocessors: based on architecture and instruction set
- Fault classes:
  - Addressing faults affecting the register-decoding function, instruction-decoding, and instruction sequencing
  - Faults in the data-storage, data-transfer, data-manipulation functions
- Fault model development
  - Determine which instructions are “easy” to execute — such as uses fewest resources, fewest cycles — easy to control and observe
  - Use such instructions to read and write register file to test register file and address decoding logic
  - Test buses by moving different types of data on buses
  - Test ALU by executing ALU related instructions such as ADD, SUB, ...
  - Buses: stuck-at and bridging faults
  - ALU, register file: stuck-at
- Algorithm development
  - Develop simple sub-programs for each sub-unit testing
  - Put them together

“Distributed Test” strategy developed by Intel

- Wafer Test
- Package Test

Content is distributed across these steps

Additional References

- Int. Workshop on Microprocessor Test and Verification (MTV 2010)
  - [http://mtvcon.org](http://mtvcon.org)
- IEEE Transactions on VLSI (TVLSI), 2007. Special Section on “Autonomous Silicon Validation and Testing of Microprocessors and Microprocessor-Based Systems”
- IEEE Design & Test, 2000. Special Issue on Microprocessor testing and verification
- Abramovici, et. al — book Chapter 8 (not up to date but good introduction)

“Snapshot” of Recent Microprocessor Testing Research Papers

- Intel
- AMD
- SUN Microsystems
- IBM

1. Intel

- Intel high performance 3GHz uProcessor, multiple clock domains, multi-cycle paths, domino logic
- Concerns: silicon area, leakage power, scan performance impact
- DFT uses a Hierarchical Scan Architecture (“divide and conquer” strategy)
  - Design partitioned into clusters (e.g., floating point execution cluster)
  - A cluster contains more units
  - Each cluster has one cluster test controller (CTC) and at least one unit test controller (UTC)
- Each CTC has 36 scan chains that allow testing of partitions formed by selected clusters, units or combinations

Intel

- Scan chains not in partition under test can be bypassed
- ATPG patterns are generated using the scan-based ATPG tools
- Skip scan methodologies
  - Skip scan technique or Data Path Interleaved Scan (DI-Scan)
  - Follow a set of Di-Scan rules: Di-Scan used only in datapath pipelines, control logic is full scan, etc.
- Cache/memory testing
  - Programmable built-in self-testing (PBIST)
  - Access to all portions of PBIST is available through the JTAG TAP controller
  - Direct access testing (DAT): 100 times faster production test
  - Programmable weak-write test mode (PWWTM): to detect stability types of defects in memory cells
Intel

- Integrated test controller (ITC) includes the TAP logic
  - Complies with JTAG (IEEE 1149.1)
  - Provides access to testability and debug features:
    - Micro-breakpoints
    - Control register bus access
    - Scan, Scanout, Signature mode
    - Thermal sensor control
    - Fuse programming, DAT mode
    - Boundary scan register
- Full chip ATPG methodology with a very low scan overhead
- On-chip weighted random patterns BIST structure including a test compression structure

2. AMD

- 33-element partitioning → 80% reduction in test time compared to a flat model
- Advantages of modular test:
  - Reduced ATPG run-time
  - Greater test reuse
  - Simplified verification and scan chain failure debug
  - Reduced test time
- Note that this is similar to the Intel approach as divide-and-conquer

AMD

- Partitioning in three major steps:
  - Disposition of partition boundaries – done by surrounding each test module with a core test wrapper
  - Connect partition module to the test resources – known as providing a test access mechanism (TAM). Use 40 scan chains
  - Test boundaries are selected considering:
    - Maximizing test coverage
    - Minimizing pattern count
    - Using the shortest possible scan chains
    - Minimizing routing overhead
    - Re-using existing scan registers at partition boundaries
    - Allowing parallel module testing if desired

AMD Athlon Chip

- Design partitioned into 10 top-level modules and 33 second-level modules
- Test time is reduced with 38% compared to the non-modular approach. Attributed to reduction in the scan chain lengths in each module compared to the length of 4000 in the flat case
- Number of flops increased with 5% (due to wrapper cells)
- The cumulative pattern count of the modules 370% higher

3. SUN

- Niagara2 SPARC:
  - 8 processor cores, 1.4GHz, 4MB on-chip L2 cache, 65nm technology
  - 8 clock domains, mixture of full custom, semi-custom, and ASIC design styles, 300 SRAMs
- Level sensitive scan architecture
- Every SRAM tested with at-speed MBIST
- Scan chains
  - More than 1 million flops are organized as 32 scan chains
  - 84 JTAG scan chain configurations, 2 manufacturing scan chains
  - 35 MBIST chains for rapid programming of MBIST configuration registers

SUN

- Stuck-at test coverage: 98.5%
- Transition test coverage: 82%
- Path delay testing
  - 15,000 paths in each core tested at-speed
- SRAM access
  - Each SRAM is equipped with scalable input flops
  - Micro Test: process used to access the SRAM during debug from the JTAG port
- Memory BIST: at-speed testing
  - 80 MBIST engines
  - March C forms the basis of the test algorithm
  - Read after write worst case (RAWWC) test
- FIFO memories (200Kbits): equipped with custom clock MUX
- CAM, Double-pumped memories (network interface unit): March tests
- Direct Memory Observe test: combination of MBIST with direct pin access to facilitate embedded SRAM bitmapping
- Support for JTAG 1149.1 boundary scan testing
4. IBM

- **Timing uncertainty** comprised of:
  - PLL jitter
  - Clock distribution skew
  - Across chip variations
  - Power supply noise
- On-chip measurement macro called SKITTER (skew + jitter): measures timing uncertainty from all combined sources; 5-8 ps resolution
- Very sensitive monitor of power supply noise, as dominant factor of timing uncertainty

IBM

- SKITTER used in IBM microprocessors: PPC970MP, XBOX360, CELL broadband engine, POWER6

IBM

- Changing location of the edges in the Skitter sampling latches is a good indicator of the variations in chip timing
  - The Skitter itself can be self-monitoring and trigger a readout if an edge is detected in a bin where it is not expected
  - Measurements can be converted from bin counts into picoseconds
  - The shift in an edge bin can be converted into mV of VDD noise
- Duty cycle measurements useful
- Multiple Skitters on chip: for each core, at different locations on chip, in the front side bus, etc.

Summary

- Functional testing
- Design hierarchy & circuit partitioning - “Divide-and-conquer” seems to be a successful testing strategy
- (M)BIST for large memories/arrays
- Special BIST for small buffers
- Scan for random logic
- Boundary Scan for test control and board level testing
- Full chip testing employs multiple scan chains, MBIST, boundary scan, transition and path delay tests
- Other design for testability and debug/diagnosis: Skitter, software based defect detection and diagnosis
- Fault tolerance techniques: self repairing microprocessor arrays