Lecture 28
IEEE 1149.1 JTAG Boundary Scan Standard

- Motivation
- Bed-of-nails tester
- System view of boundary scan hardware
- Elementary scan cell
- Test Access Port (TAP) controller
- Boundary scan instructions
- Summary

Motivation for Standard

- Bed-of-nails printed circuit board tester gone
  - We put components on both sides of PCB & replaced DIPs with flat packs to reduce inductance
    - Nails would hit components
  - Reduced spacing between PCB wires
    - Nails would short the wires
  - PCB Tester must be replaced with built-in test delivery system -- JTAG does that
  - Need standard System Test Port and Bus
  - Integrate components from different vendors
    - Test bus identical for various components
    - One chip has test hardware for other chips

April 20, 2001
Bed-of-Nails Tester Concept

[Diagram showing a bed-of-nails tester concept]

April 20, 2001

Bed-of-Nails Tester

[Diagram showing the bed-of-nails tester setup]

April 20, 2001
Purpose of Standard

- Lets test instructions and test data be serially fed into a component-under-test (CUT)
  - Allows reading out of test results
  - Allows RUNBIST command as an instruction
  - Too many shifts to shift in external tests
- JTAG can operate at chip, PCB, & system levels
- Allows control of tri-state signals during testing
- Lets other chips collect responses from CUT
- Lets system interconnect be tested separately from components
- Lets components be tested separately from wires

System Test Logic
Instruction Register Loading with JTAG

System View of Interconnect
Boundary Scan Chain View

Elementary Boundary Scan Cell
Serial Board / MCM Scan

Parallel Board / MCM Scan
Tap Controller Signals

- **Test Access Port (TAP)** includes these signals:
  - **Test Clock Input (TCK)** -- Clock for test logic
    - Can run at different rate from system clock
  - **Test Mode Select (TMS)** -- Switches system from functional to test mode
  - **Test Data Input (TDI)** -- Accepts serial test data and instructions -- used to shift in vectors or one of many test instructions
  - **Test Data Output (TDO)** -- Serially shifts out test results captured in boundary scan chain (or device ID or other internal registers)
  - **Test Reset (TRST)** -- Optional asynchronous TAP controller reset
Tap Controller State Diagram

Tap Controller Timing

April 20, 2001
SAMPLE / PRELOAD
Instruction -- SAMPLE

Purpose:
1. Get snapshot of normal chip output signals
2. Put data on bound. scan chain before next instr.

SAMPLE / PRELOAD
Instruction -- PRELOAD

April 20, 2001
**EXTEST Instruction**

- **Purpose:** Test off-chip circuits and board-level interconnections

**INTEST Instruction**

- **Purpose:**
  1. Shifts external test patterns onto component
  2. External tester shifts component responses out
**INTEST Instruction Clocks**

- **Control of applied system clock during INTEST**

  - System Clock (from pin)
  - Controller State
  - TCK

- **Use of TCK for on-chip system logic clock**

  - System Clock (in INTEST)
  - Controller State
  - TCK

---

**RUNBIST Instruction**

- **Purpose:** Allows you to issue BIST command to component through JTAG hardware
- **Optional instruction**
- **Lets test logic control state of output pins**
  1. Can be determined by pin boundary scan cell
  2. Can be forced into high impedance state
- **BIST result (success or failure) can be left in boundary scan cell or internal cell**
  - Shift out through boundary scan chain
- **May leave chip pins in an indeterminate state (reset required before normal operation resumes)**

---

**April 20, 2001**
**CLAMP Instruction**

- **Purpose:** Forces component output signals to be driven by boundary-scan register
- Bypasses the boundary scan chain by using the one-bit *Bypass Register*
- Optional instruction
- May have to add RESET hardware to control on-chip logic so that it does not get damaged (by shorting 0’s and 1’s onto an internal bus, etc.)

April 20, 2001

---

**IDCODE Instruction**

- **Purpose:** Connects the component device identification register serially between *TDI* and *TDO*
  - In the *Shift-DR TAP controller state*
- Allows board-level test controller or external tester to read out component ID
- Required whenever a JEDEC identification register is included in the design

April 20, 2001
Device ID Register -- JEDEC Code

<table>
<thead>
<tr>
<th>MSB</th>
<th>31 28</th>
<th>27 12</th>
<th>11 1</th>
<th>LSB</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (4 bits)</td>
<td>Part Number (16 bits)</td>
<td>Manufacturer Identity (11 bits)</td>
<td>‘1’ (1 bit)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

USERCODE Instruction

- **Purpose:** Intended for user-programmable components (FPGA's, EEPROMs, etc.)
  - Allows external tester to determine user programming of component
- **Selects the device identification register** as serially connected between TDI and TDO
- User-programmable ID code loaded into device identification register
  - On rising TCK edge
- Switches component test hardware to its system function
- Required when Device ID register included on user-programmable component
**HIGHZ Instruction**

- **Purpose:** Puts all component output pin signals into high-impedance state
- **Control chip logic to avoid damage in this mode**
- **May have to reset component after HIGHZ runs**
- **Optional instruction**

![HIGHZ Instruction Diagram](image)

---

**BYPASS Instruction**

- **Purpose:** Bypasses scan chain with 1-bit register

![BYPASS Instruction Diagram](image)
Optional / Required Instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYPASS</td>
<td>Mandatory</td>
</tr>
<tr>
<td>CLAMP</td>
<td>Optional</td>
</tr>
<tr>
<td>EXTEST</td>
<td>Mandatory</td>
</tr>
<tr>
<td>HIGHZ</td>
<td>Optional</td>
</tr>
<tr>
<td>IDCODE</td>
<td>Optional</td>
</tr>
<tr>
<td>INTEST</td>
<td>Optional</td>
</tr>
<tr>
<td>RUNBIST</td>
<td>Optional</td>
</tr>
<tr>
<td>SAMPLE / PRELOAD</td>
<td>Mandatory</td>
</tr>
<tr>
<td>USERCODE</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Summary

- Boundary Scan Standard has become absolutely essential --
  - No longer possible to test printed circuit boards with bed-of-nails tester
  - Not possible to test multi-chip modules at all without it
  - Supports BIST, external testing with Automatic Test Equipment, and boundary scan chain reconfiguration as BIST pattern generator and response compacter
  - Now getting widespread usage
Lecture 30
IEEE 1149.4 JTAG
Analog Test Access Port and Standard

- Motivation
- Bus overview
- Hardware faults
- Test Bus Interface Circuit (TBIC)
- Analog Boundary Module (ABM)
- Instructions
- Specialized Bus Circuits
- Summary

Purpose of Analog JTAG Standard

- For a System-on-a-Chip (SOC):
  - Cannot assume that we are interconnecting pre-tested modules
  - Internal module probing is impractical
  - Solution: Use boundary scan structure to partition analog, digital, and memory subsystems in SOC and test each separately
- Analog JTAG test capability:
  - Oriented towards measuring external component values or internal impedances (shorts, opens, wrong components)
  - Not intended for DSP type analog tests
Analog Test Bus

**PROs:**
- Usable with digital JTAG boundary scan
- Adds analog testability – both controllability and observability
- Eliminates large area needed for analog test points

**CONs:**
- May have a 5% measurement error
- C-switch sampling devices couple all probe points capacitively, even with test bus off – requires more elaborate (larger) switches
- Stringent limit on how far data can move through the bus before it must be digitized to retain accuracy

---

**Analog Test Bus Diagram**

![Diagram of Analog Test Bus](image)

- ATAP
- TAP
- Test control block
- Digital I/O
- Mixed-signal circuitry
- Test bus interface circuit
- Control signals
- TDI
- TCK
- TMS
- Analog I/O
- VH
- VL
- VG
- VH
- VL
- VG
- TDO
- Test Access Port
- Analog Test Access Port
- TCB -- 1149.1 TAP, Instruction reg., & decode
Analog Boundary Module

From TDI-TDO chain

From analog circuit

1-bit digitizer

Uncommitted, usable for ABM test

Control decoder logic

To TDI-TDO chain

From TAP controller

Analog pin

SB2

SB1

SG

SL

SH

SD

AB1

AB2

VH

SL

SG

SB1

SV

U

C

B1

B2

Analog Defects and Faults

Extended Interconnect

Short

Misloaded Device

Open

Wrong Value

Simple Interconnect

Short

Differential Interconnect

(Analog or Digital)

Single-Ended Transmission Point

Single-Ended Reception Point
Need for Discrete Components

- Impedance matching of transmission lines necessary – merchant ICs will not have built-in impedance matching resistances
- Discrete resistors use much power – may prevent them from being on-chip
- Impossible to make high-valued, accurate inductors or transformers on chip
- Integrated $R$, $C$, $L$ components are never as precise as external ones
- Some ICs can be extended to more functions if external $R$, $C$, or $L$ value can be changed

Measurement Limitations with 1149.4

- Must test device with power on
- Multiplexing done with silicon devices, not relays
- Introduces unwanted impedances during testing
- Has additional current leakages to ground
- CMOS silicon switches non-linear over larger signal swings – may also be slow
- 1149.4 bus has less than 1 MHz bandwidth
## Switch Limitations

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Relay</th>
<th>CMOS</th>
<th>Bipolar</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-resistance</td>
<td>$10^{-2}$ Ω</td>
<td>$10^2$ to $10^3$ Ω</td>
<td>Varies $10^{10}$ Ω</td>
</tr>
<tr>
<td>Off-resistance</td>
<td>$10^{12}$ Ω</td>
<td>$10^{12}$ Ω</td>
<td>No</td>
</tr>
<tr>
<td>Bidirectional ?</td>
<td>Yes</td>
<td>Yes</td>
<td>&lt; 1 μs</td>
</tr>
<tr>
<td>Switching time</td>
<td>≥ 500 μs</td>
<td>&lt; 1 μs</td>
<td>100 to 5000</td>
</tr>
<tr>
<td>Area μm²</td>
<td>96.7 x $10^6$</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

## Chaining of 1149.4 ICs

- **Digital Inputs:** TDI, TCK, TMS
- **Analog Inputs:** AT1, AT2
- **Outputs:** U1, U2, TDO

---

April 20, 2001
**Analog Test Access Port**

- *TDI, TDO, TCK, TMS* signals from Digital standard are required
- *TRST* signal from Digital standard is optional
- New required analog signals:
  - *AT1* – for analog stimulus
  - *AT2* – for sending analog response to ATE
  - *AT1* and *AT2* can be partitioned
- Digital part same as before, except:
  - New *Test Bus Interface Circuit* (TBIC)
  - Multiple digital pin cells grouped into *Digital Boundary Module* (DBM)
  - Set of cells required to control analog pin grouped into *Analog Boundary Module* (ABM)

---

**Test Bus Interface Circuit**

[Diagram showing Test Bus Interface Circuit with labels for AT1, AT2, S1-S10, V_H, V_L, V_TH, and switch configurations for interconnect and bus-to-bus switching.]
TBIC Functions

- Connect or isolate analog measurement buses $AB1$ and $AB2$ within chip to or from external $AT1$ and $AT2$ signals
- Perform 1149.1 interconnect tests on $AT1$ and $AT2$ pins
  - Support coarse digitization relative to threshold $V_{TH}$
- Support analog characterization measurements
  - Clamp busses not being driven

TBIC Switching Patterns

<table>
<thead>
<tr>
<th>$P$ #</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$ATn$ disconnect (high Z), clamp $ABn$</td>
</tr>
<tr>
<td>1</td>
<td>Connect $AT2$ &amp; $AB2$</td>
</tr>
<tr>
<td>2</td>
<td>Connect $AT1$ &amp; $AB1$</td>
</tr>
<tr>
<td>3</td>
<td>Connect $ATn$ &amp; $ABn$</td>
</tr>
<tr>
<td>4</td>
<td>$AT1/2$ drive 00 out</td>
</tr>
<tr>
<td>5</td>
<td>$AT1/2$ drive 01 out</td>
</tr>
<tr>
<td>6</td>
<td>$AT1/2$ drive 10 out</td>
</tr>
<tr>
<td>7</td>
<td>$AT1/2$ drive 11 out</td>
</tr>
<tr>
<td>8</td>
<td>For characterization</td>
</tr>
<tr>
<td>9</td>
<td>For characterization</td>
</tr>
</tbody>
</table>

Switch state $S1$-$S10$ for patterns given in book

- $P0$ & $P4$ -- $P7$ for 1149.1 interconnect test
- $P1$ -- $P3$ for analog measurement
TBIC Switch Controls

Analog Boundary Module Has Four Control Cells

- Work in conjunction with TBIC and various 1149.4 bus modes to set state for one analog pin:
  - *Calibrate* (Ca)
  - *Control* (Co)
  - *Data1* (D1)
  - *Data2* (D2)

- Test mode determined by 4 ABM digital pins and by TBIC switches *S1*-*S10*
### ABM Switch Patterns

#### Switch states for the pattern given in book

<table>
<thead>
<tr>
<th>Pattern #</th>
<th>Pin State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Completely isolated</td>
</tr>
<tr>
<td>1</td>
<td>Monitored (mon.) by AB2</td>
</tr>
<tr>
<td>2</td>
<td>Connected (conn.) to AB1</td>
</tr>
<tr>
<td>3</td>
<td>Conn. to AB1, mon. by AB2</td>
</tr>
<tr>
<td>4</td>
<td>Connected to VG</td>
</tr>
<tr>
<td>5</td>
<td>Conn. to VG, mon. by AB2</td>
</tr>
<tr>
<td>6</td>
<td>Conn. to VG &amp; AB1</td>
</tr>
<tr>
<td>7</td>
<td>Conn. to VTG &amp; AB1, mon. by AB2</td>
</tr>
<tr>
<td>8</td>
<td>Conn. to VL</td>
</tr>
<tr>
<td>9</td>
<td>Conn. to VL, mon. by AB2</td>
</tr>
<tr>
<td>10</td>
<td>Conn. to VL &amp; AB1</td>
</tr>
<tr>
<td>11</td>
<td>Conn. to VL &amp; AB1, mon. by AB2</td>
</tr>
<tr>
<td>12</td>
<td>Conn. to VH</td>
</tr>
<tr>
<td>13</td>
<td>Conn. to VH, mon. by AB2</td>
</tr>
<tr>
<td>14</td>
<td>Conn. to VH &amp; AB1</td>
</tr>
<tr>
<td>15</td>
<td>Conn. to VH &amp; AB1, mon. by AB2</td>
</tr>
<tr>
<td>16</td>
<td>Conn. to core, isolated from test</td>
</tr>
<tr>
<td>17</td>
<td>Conn. to core, mon. by AB2</td>
</tr>
<tr>
<td>18</td>
<td>Conn. to core &amp; AB1</td>
</tr>
<tr>
<td>19</td>
<td>Conn. to core &amp; AB1, mon. by AB2</td>
</tr>
</tbody>
</table>

### TBIC Patterns & ABM Values

#### 4 Cells

<table>
<thead>
<tr>
<th>4 Cells</th>
<th>EXTEST</th>
<th>CLAMP</th>
<th>RUNBIST</th>
<th>PROBE</th>
<th>INTTEST</th>
<th>HIGHZ</th>
<th>BYPASS, SAMPLE</th>
<th>PRELOAD, IDCODE</th>
<th>USERCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>P0</td>
<td></td>
<td></td>
<td>P0</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>P1</td>
<td></td>
<td></td>
<td>P1</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>P2</td>
<td></td>
<td></td>
<td>P2</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>P3</td>
<td></td>
<td></td>
<td>P3</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td>P4</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td>P5</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>P6</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>P7</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>P0</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>P8</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>P9</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>P0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analog Boundary Module Functions

- One-bit digitizer captures pin voltage and interprets it as digital
- Simultaneously provides one more more of these functions at an analog pin:
  - Connect pin to $V_L$
  - Connect pin to $V_H$
  - Connect pin to $V_G$ (reference quality)
  - Connect pin to $AB1$ (provides current)
  - Connect pin to $AB2$ (monitors voltage)

Electro-Static Discharge Protection for ABM

(a) Ordinary pin                       (b) ABM pin
**EXTEST** Instruction

- Can disable or enable each of these connections for each analog pin:
  - Core-disconnect state (disconnected from internal analog circuitry)
  - Connect to $V_L$
  - Connect to $V_H$

- Had to be individually pin programmable, because bias voltage pins can never be disconnected, and low impedance $R$’s or $L$’s often cannot be disconnected.

- Core-disconnect state often not implemented with a transistor, since that can reduce driver performance.

---

**ATE External Impedance Measurement with **EXTEST**

![ATE External Impedance Measurement Diagram](image)
1149.4 Measurement of External Impedance

- (a) Pin 1 voltage measurement

Pin 2 Voltage Measurement

\[ Z = \frac{V_{Pin1} - V_{Pin2}}{I} \]
CLAMP and HIGHZ Instructions

- **CLAMP** – Disconnects all pins from cores and freezes analog pins in present state
  - Freezes TBIC in present state
  - Keeps circuit quiescent, while \( V \) and \( I \) are measured in other parts
- **HIGHZ** – Opens core disconnect switch \( SB \)
  - Disconnects all test circuits
  - Disables TBIC

New **PROBE** Instruction

- Required
- Works similarly to digital SAMPLE instruction
- Operates on both digital and analog pins
- Allows continuous time sampling while analog core is functioning
  - Can only sample 1 analog pin at a time (only 1 set of \( ABn \) wires exists)
  - Sets all Analog and Digital Boundary Modules to connect all pins to cores
- \( AB \) switch may add parasitic element into circuit
- Most useful for noise measurements
- Can make \( f \) measurements only up to 1 kHz
**INTEST Instruction**

At any time, only 1 analog pin can be stimulated and only 1 analog pin can be read.

**RUNBIST and SAMPLE / PRELOAD Instructions**

- **RUNBIST** – operates exactly as in 1149.1 digital standard
  - Analog pins can either mimic **HIGHZ** or **CLAMP** instructions
- **SAMPLE / PRELOAD** – for Analog pins
  - Digitizes the analog pin voltage
    - Stored as ‘1’ if \( V_{TH} \) otherwise as ‘0’
    - Stored in boundary register
Differential Interconnect

- Greatly improves common-mode noise rejection
- Can still work, even when single lines or \( R' \)s are opened or shorted

**Partitioned \( AB \) Busses**
Isolation of Analog and Digital Cores

- 1149.4 standard requires that a digital boundary module be on each digital line between digital and analog core
  - Only when \textit{INTEST} or \textit{RUNBIST} instructions supported, otherwise can eliminate DBM
- Can use analog boundary module to test digital pins & interconnect with 1149.4

Analog Switch to Reduce Coupling

![Analog Switch Diagram](a) \quad \textit{Enable}

![Analog Switch Diagram](b) \quad \textit{Enable}
Guarding Between Signals

Summary

- Analog test bus allows static analog tests
- Non-static or feedback circuits are hard to test
- Good for locating shorts, opens, and wrong external component values
  - $V_H$ and $V_L$ switches in ABM must be able to survive large voltage differences
- Needs customizing digitizing receiver for digitizing analog bus – inverter not suitable
- Can eliminate separate process monitor transistors and resistors on wafers – saves area
- Needs large, low-resistance transistor switches to avoid common mode measurement errors
Lecture - Logic Simulation

- What is simulation?
- Design verification
- Circuit modeling
- True-value simulation algorithms
  - Compiled-code simulation
  - Event-driven simulation
- Summary

Simulation Defined

- Definition: Simulation refers to modeling of a design, its function and performance.
- A software simulator is a computer program; an emulator is a hardware simulator.
- Simulation is used for design verification:
  - Validate assumptions
  - Verify logic
  - Verify performance (timing)
- Types of simulation:
  - Logic or switch level
  - Timing
  - Circuit
  - Fault
Simulation for Verification

- Specification
- Design (netlist)
- True-value simulation
- Input stimuli
- Computed responses
- Design changes
- Response analysis

Modeling for Simulation

- Modules, blocks or components described by
  - Input/output (I/O) function
  - Delays associated with I/O signals
  - Examples: binary adder, Boolean gates, FET, resistors and capacitors
- Interconnects represent
  - ideal signal carriers, or
  - ideal electrical conductors
- Netlist: a format (or language) that describes a design as an interconnection of modules. Netlist may use hierarchy.
Example: A Full-Adder

HA: inputs: a, b; outputs: c, f; AND: A1, (a, b); (c); AND: A2, (d, e); (f); OR: O1, (a, b); (d); NOT: N1, (c); (a); HA1, (A, B), (D, E); HA2, (E, C), (F, Sum); OR: O2, (D, F), (Carry);

Logic Model of MOS Circuit

Pa and Pb are interconnect or propagation delays
DC is inertial delay of gate

Ca, Cb and Cc are parasitic capacitances
**Options for Inertial Delay**
(simulation of a NAND gate)

- **Signals States**
  - Two-states (0, 1) can be used for purely combinational logic with zero-delay.
  - Three-states (0, 1, X) are essential for timing hazards and for sequential logic initialization.
  - Four-states (0, 1, X, Z) are essential for MOS devices. See example below.
  - Analog signals are used for exact timing of digital logic and for analog circuits.
### Modeling Levels

<table>
<thead>
<tr>
<th>Modeling level</th>
<th>Circuit description</th>
<th>Signal values</th>
<th>Timing</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Programming language-like HDL</td>
<td>0, 1</td>
<td>Clock boundary</td>
<td>Architectural and functional verification</td>
</tr>
<tr>
<td>Logic</td>
<td>Connectivity of Boolean gates, flip-flops and transistors</td>
<td>0, 1, X and Z</td>
<td>Zero-delay unit-delay, multiple-delay</td>
<td>Logic verification and test</td>
</tr>
<tr>
<td>Switch</td>
<td>Transistor size and connectivity, node capacitances</td>
<td>0, 1 and X</td>
<td>Zero-delay</td>
<td>Logic verification</td>
</tr>
<tr>
<td>Timing</td>
<td>Transistor technology data, connectivity, node capacitances</td>
<td>Analog voltage</td>
<td>Fine-grain timing</td>
<td>Timing verification</td>
</tr>
<tr>
<td>Circuit</td>
<td>Tech. Data, active/passive component connectivity</td>
<td>Analog voltage, current</td>
<td>Continuous time</td>
<td>Digital timing and analog circuit verification</td>
</tr>
</tbody>
</table>

### True-Value Simulation Algorithms

- **Compiled-code simulation**
  - Applicable to zero-delay combinational logic
  - Also used for cycle-accurate synchronous sequential circuits for logic verification
  - Efficient for highly active circuits, but inefficient for low-activity circuits
  - High-level (e.g., C language) models can be used

- **Event-driven simulation**
  - Only gates or modules with input events are evaluated (*event means a signal change*)
  - Delays can be accurately simulated for timing verification
  - Efficient for low-activity circuits
  - Can be extended for fault simulation
Compiled-Code Algorithm

- Step 1: Levelize combinational logic and encode in a compilable programming language
- Step 2: Initialize internal state variables (flip-flops)
- Step 3: For each input vector
  - Set primary input variables
  - Repeat (until steady-state or max. iterations)
    - Execute compiled code
  - Report or save computed variables

Event-Driven Algorithm (Example)

```
a = 1
c = 1
b = 1
g = 1
d = 0
e = 1
f = 0

g = 0
c = 0
d, e
f, g
```

Time, t
**Time Wheel (Circular Stack)**

Current time pointer

- max
- \( t=0 \)
- 1
- 2
- 3
- 4
- 5
- 6
- 7

Event link-list

---

**Efficiency of Event-driven Simulator**

- Simulates events (value changes) only
- Speed up over compiled-code can be ten times or more; in large logic circuits about 0.1 to 10% gates become active for an input change

Steady 0

\[ \ldots \ldots 0, 1, 0, 1, \ldots \]

Large logic block without activity

Steady 0

(no event)
Summary

- Logic or true-value simulators are essential tools for design verification.
- Verification vectors and expected responses are generated (often manually) from specifications.
- A logic simulator can be implemented using either compiled-code or event-driven method.
- Per vector complexity of a logic simulator is approximately linear in circuit size.
- Modeling level determines the evaluation procedures used in the simulator.

Fault Simulation

- Problem and motivation
- Fault simulation algorithms
  - Serial
  - Parallel
  - Deductive
  - Concurrent
- Random Fault Sampling
- Summary
Problem and Motivation

- **Fault simulation Problem:** Given
  - A circuit
  - A sequence of test vectors
  - A fault model
  - Determine
    - Fault coverage - fraction (or percentage) of modeled faults detected by test vectors
    - Set of undetected faults

- **Motivation**
  - Determine test quality and in turn product quality
  - Find undetected fault targets to improve tests

Fault simulator in a VLSI Design Process

1. Verified design netlist
2. Verification input stimuli
3. Fault simulator
4. Modeled fault list
5. Remove tested faults
6. Test compact or
7. Test generator
8. Fault coverage
   - Low
   - Adequate
   - Stop
9. Add vectors
10. Delete vectors
Fault Simulation Scenario

- Circuit model: mixed-level
  - Mostly logic with some switch-level for high-impedance (Z) and bidirectional signals
  - High-level models (memory, etc.) with pin faults

- Signal states: logic
  - Two (0, 1) or three (0, 1, X) states for purely Boolean logic circuits
  - Four states (0, 1, X, Z) for sequential MOS circuits

- Timing:
  - Zero-delay for combinational and synchronous circuits
  - Mostly unit-delay for circuits with feedback

Fault Simulation Scenario (continued)

- Faults:
  - Mostly single stuck-at faults
  - Sometimes stuck-open, transition, and path-delay faults; analog circuit fault simulators are not yet in common use
  - Equivalence fault collapsing of single stuck-at faults
  - Fault-dropping -- a fault once detected is dropped from consideration as more vectors are simulated; fault-dropping may be suppressed for diagnosis
  - Fault sampling -- a random sample of faults is simulated when the circuit is large
Fault Simulation Algorithms

- Serial
- Parallel
- Deductive
- Concurrent
- Differential

Serial Algorithm

- **Algorithm**: Simulate fault-free circuit and save responses. Repeat following steps for each fault in the fault list:
  - Modify netlist by injecting one fault
  - Simulate modified netlist, vector by vector, comparing responses with saved responses
  - If response differs, report fault detection and suspend simulation of remaining vectors

- **Advantages**:
  - Easy to implement; needs only a true-value simulator, less memory
  - Most faults, including analog faults, can be simulated
Serial Algorithm (Cont.)

- Disadvantage: Much repeated computation; CPU time prohibitive for VLSI circuits
- Alternative: Simulate many faults together

```
Test vectors
---------
Fault-free circuit
         | Comparator
         v
Circuit with fault f1
         | Comparator
         v
Circuit with fault f2
         | Comparator
         v
Circuit with fault fn
         | Comparator
         v
f1 detected?
---------
f2 detected?
---------
f_n detected?
---------
```

Parallel Fault Simulation

- Compiled-code method; best with two-states (0,1)
- Exploits inherent bit-parallelism of logic operations on computer words
- Storage: one word per line for two-state simulation
- Multi-pass simulation: Each pass simulates \( w-1 \) new faults, where \( w \) is the machine word length
- Speed up over serial method \( \approx w-1 \)
- Not suitable for circuits with timing-critical and non-Boolean logic
Parallel Fault Sim. Example

Bit 0: fault-free circuit
Bit 1: circuit with c s-a-0
Bit 2: circuit with f s-a-1

Deductive Fault Simulation

- One-pass simulation
- Each line $k$ contains a list $L_k$ of faults detectable on $k$
- Following true-value simulation of each vector, fault lists of all gate output lines are updated using set-theoretic rules, signal values, and gate input fault lists
- PO fault lists provide detection data
- Limitations:
  - Set-theoretic rules difficult to derive for non-Boolean gates
  - Gate delays are difficult to use
Deductive Fault Sim. Example

Notation: $L_k$ is fault list for line $k$

- $k_n$ is s-a-n fault on line $k$.

Faults detected by the input vector

Concurrent Fault Simulation

- Event-driven simulation of fault-free circuit and only those parts of the faulty circuit that differ in signal states from the fault-free circuit.
- A list per gate containing copies of the gate from all faulty circuits in which this gate differs. List element contains fault ID, gate input and output values and internal states, if any.
- All events of fault-free and all faulty circuits are implicitly simulated.
- Faults can be simulated in any modeling style or detail supported in true-value simulation (offers most flexibility.)
- Faster than other methods, but uses most memory.
Fault Sampling

- A randomly selected subset (sample) of faults is simulated.
- Measured coverage in the sample is used to estimate fault coverage in the entire circuit.
- Advantage: Saving in computing resources (CPU time and memory.)
- Disadvantage: Limited data on undetected faults.
Motivation for Sampling

- Complexity of fault simulation depends on:
  - Number of gates
  - Number of faults
  - Number of vectors

- Complexity of fault simulation with fault sampling depends on:
  - Number of gates
  - Number of vectors

Random Sampling Model

All faults with a fixed but unknown coverage

- $N_P =$ total number of faults (population size)
- $C =$ fault coverage (unknown)

Detected fault

Undetected fault

Random picking

$N_s =$ sample size

$N_s << N_P$

$c =$ sample coverage (a random variable)
Probability Density of Sample Coverage, \( c \)

\[
\frac{(x-C)^2}{2\sigma^2} = \text{Prob}(x < c \frac{1}{\sqrt{2\pi\sigma^2}}) = \frac{1}{\sqrt{2\pi\sigma^2}}
\]

\[
\sigma^2 = \frac{\text{Variance}}{\text{Sampling Error}}
\]

\[
C \pm 3\sigma = \frac{3\sqrt{C(1-C)}}{\sqrt{N_s}} \quad \left(1 + 0.44 \frac{N_s}{x} \right)
\]

Solving the quadratic equation for \( C \), we get the 3-sigma (99.7% confidence) estimate:

\[
C = \frac{3\sqrt{C(1-C)}}{\sqrt{N_s}} \pm 4.5 \quad \left(1 + 0.44 \frac{N_s}{x} \right)
\]

Where \( N_s \) is sample size and \( x \) is the measured fault coverage in the sample.

Example: A circuit with 39,096 faults has an actual fault coverage of 87.1%. The measured coverage in
Summary

- Fault simulator is an essential tool for test development.
- Concurrent fault simulation algorithm offers the best choice.
- For restricted class of circuits (combinational and synchronous sequential with only Boolean primitives), differential algorithm can provide better speed and memory efficiency (Section 5.5.6.)
- For large circuits, the accuracy of random fault sampling only depends on the sample size (1,000 to 2,000 faults) and not on the circuit size. The method has significant advantages in reducing CPU time and memory needs of the simulator.