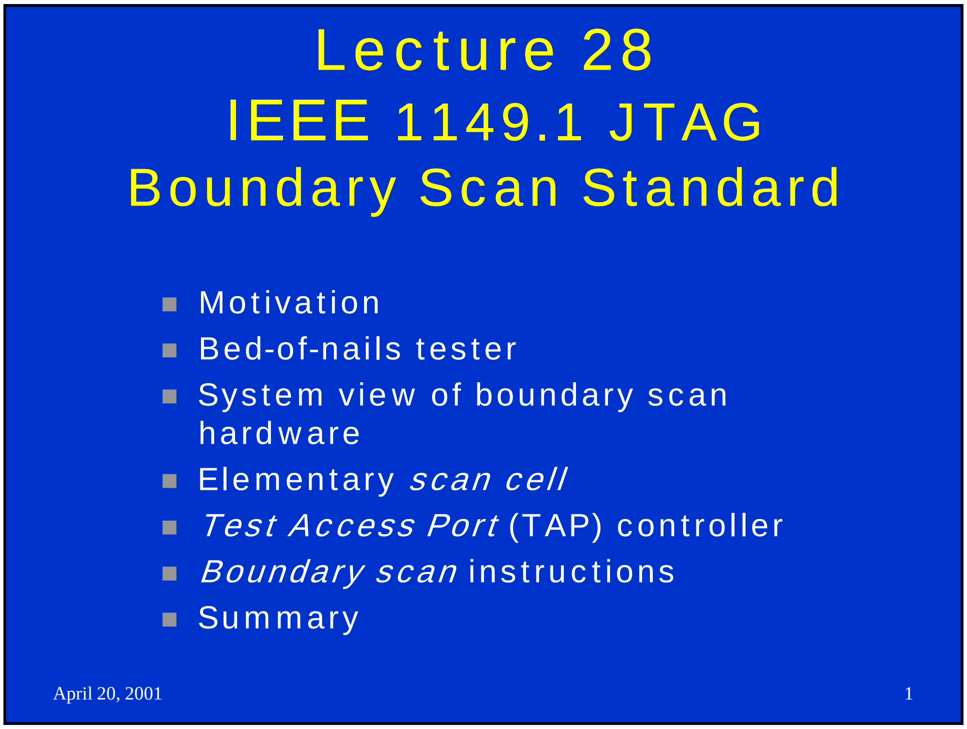
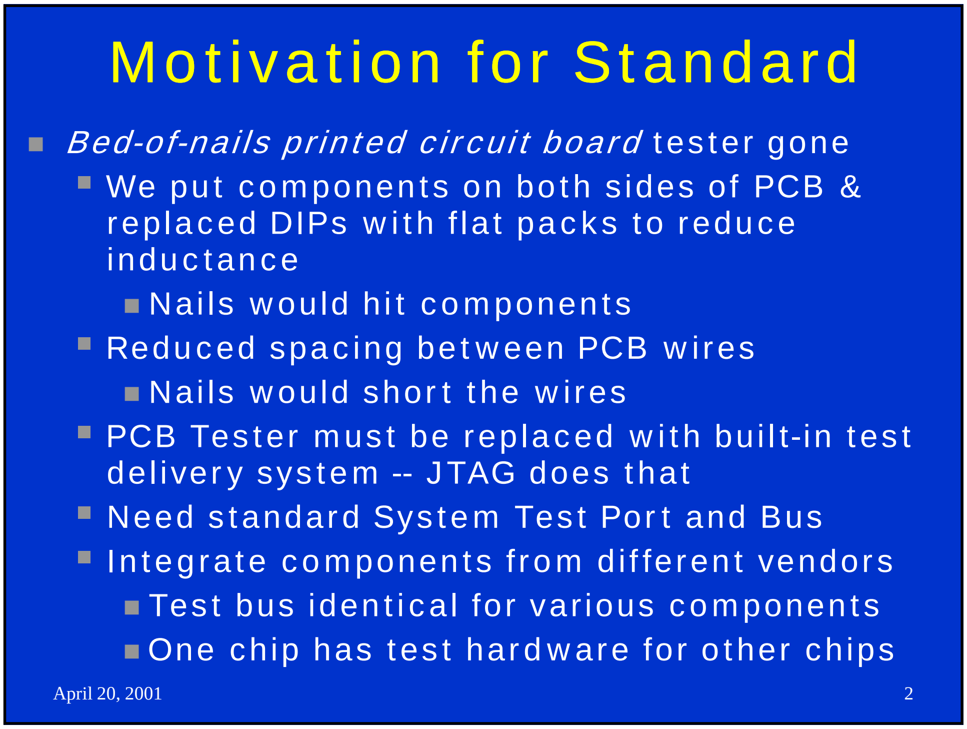


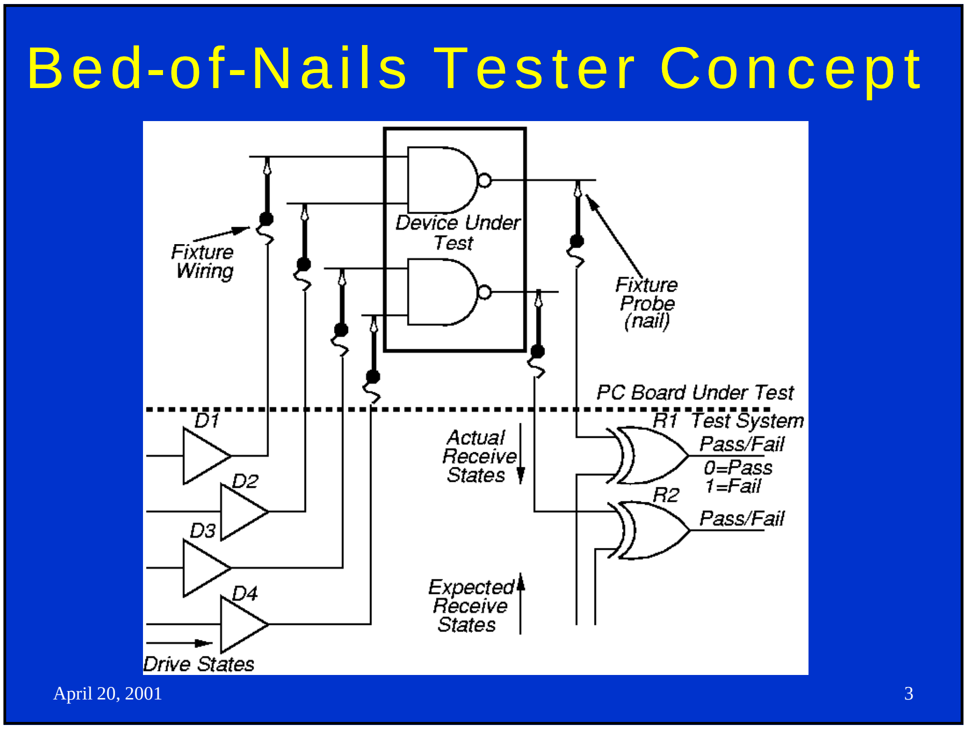
Lecture 28
IEEE 1149.1 JTAG
Boundary Scan Standard

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



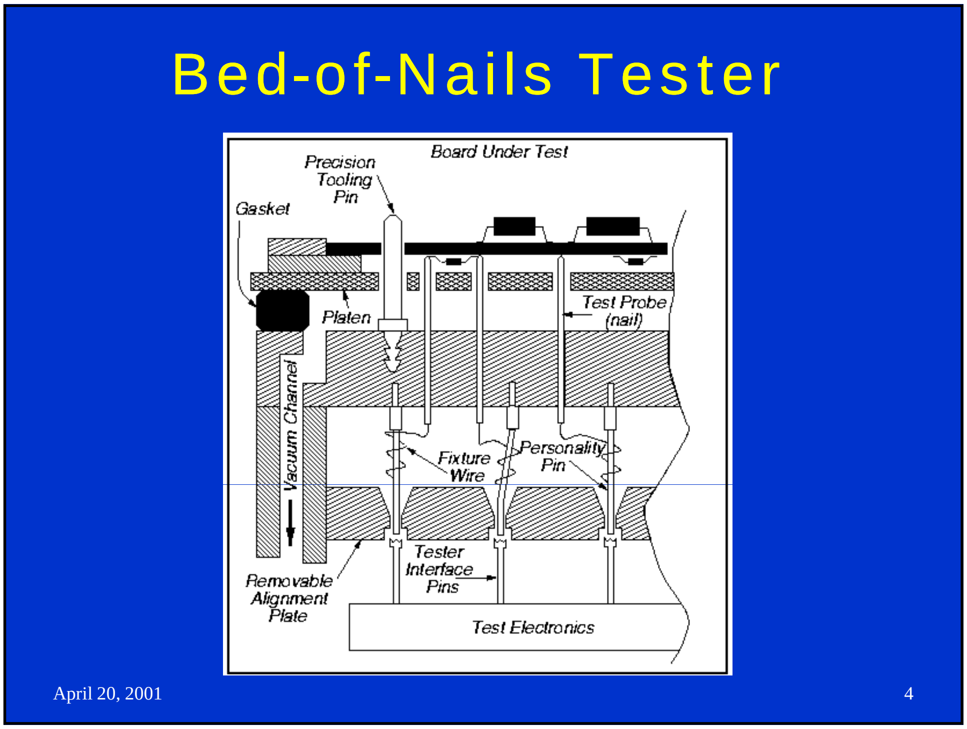
Motivation for Standard

- *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



Bed-of-Nails Tester Concept

April 20, 2001 3



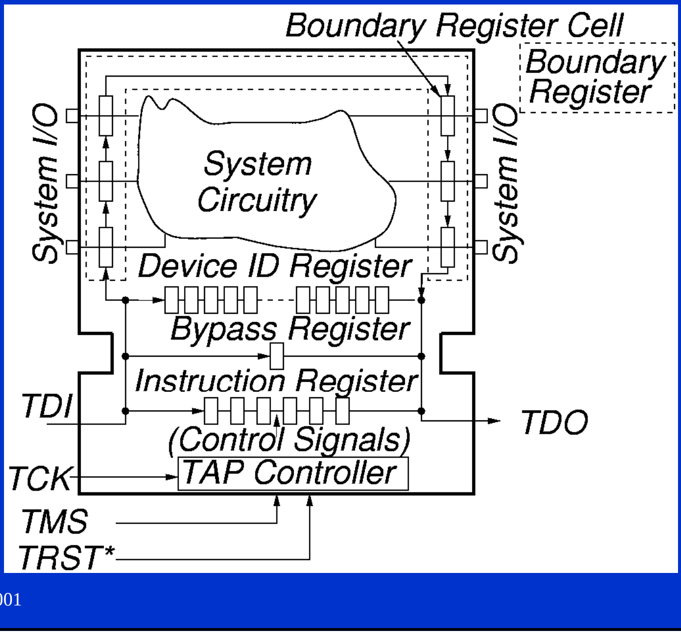
Bed-of-Nails Tester

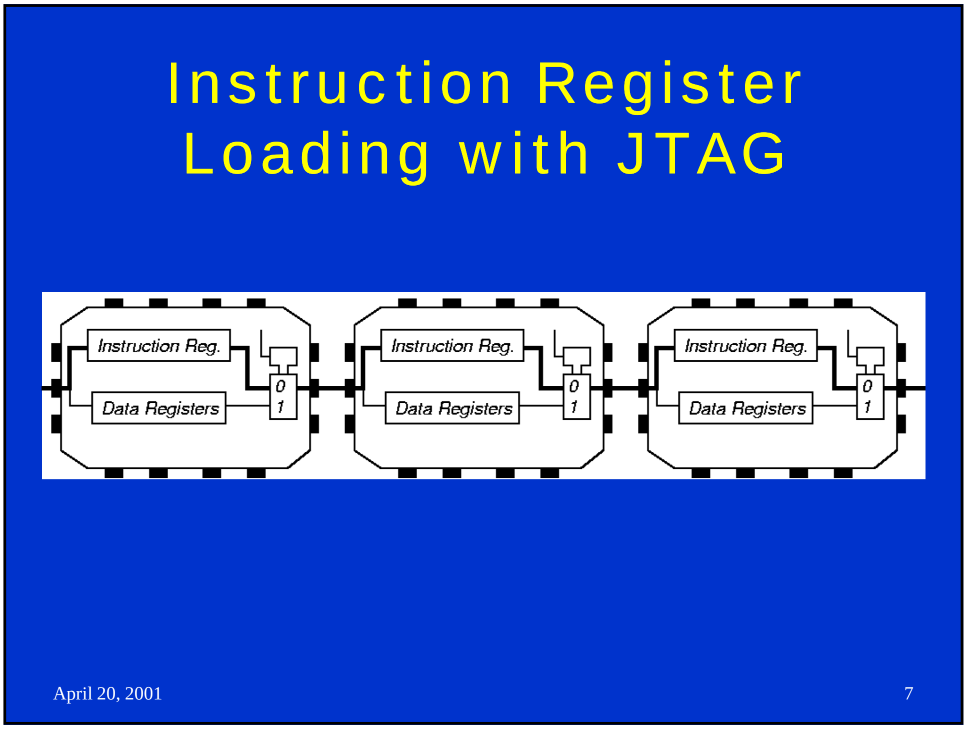
April 20, 2001 4

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

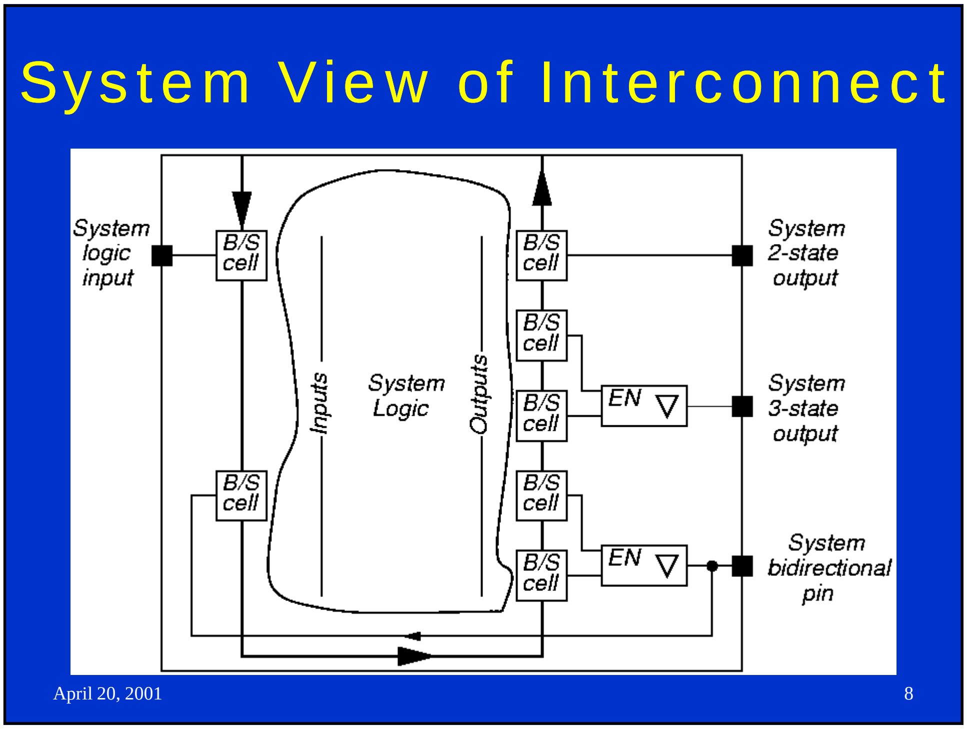
System Test Logic





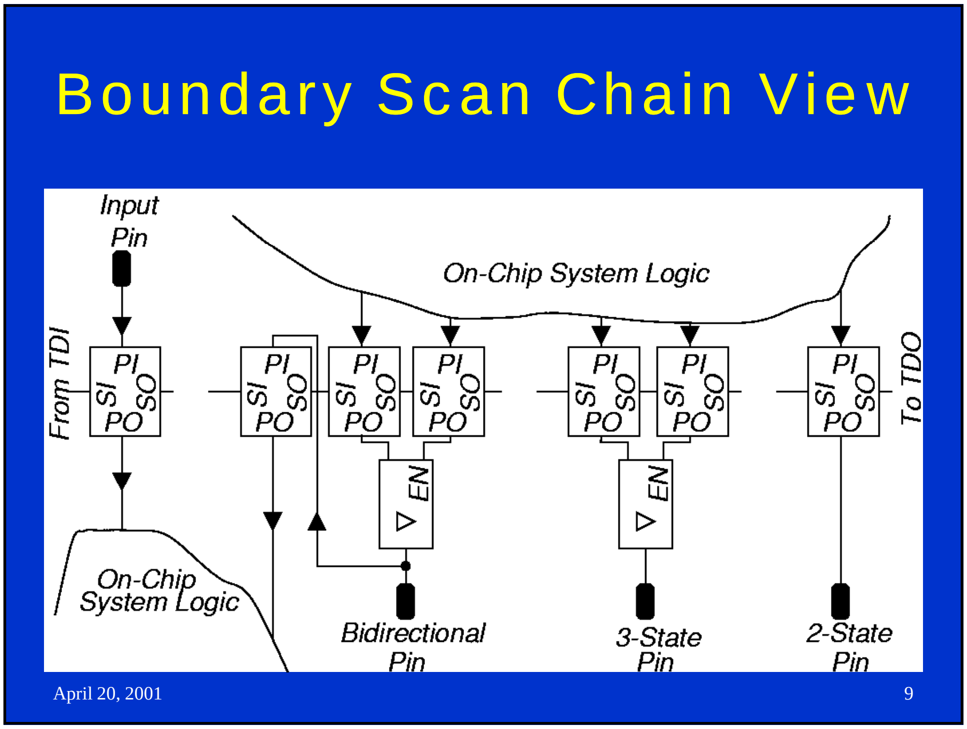
Instruction Register Loading with JTAG

April 20, 2001 7



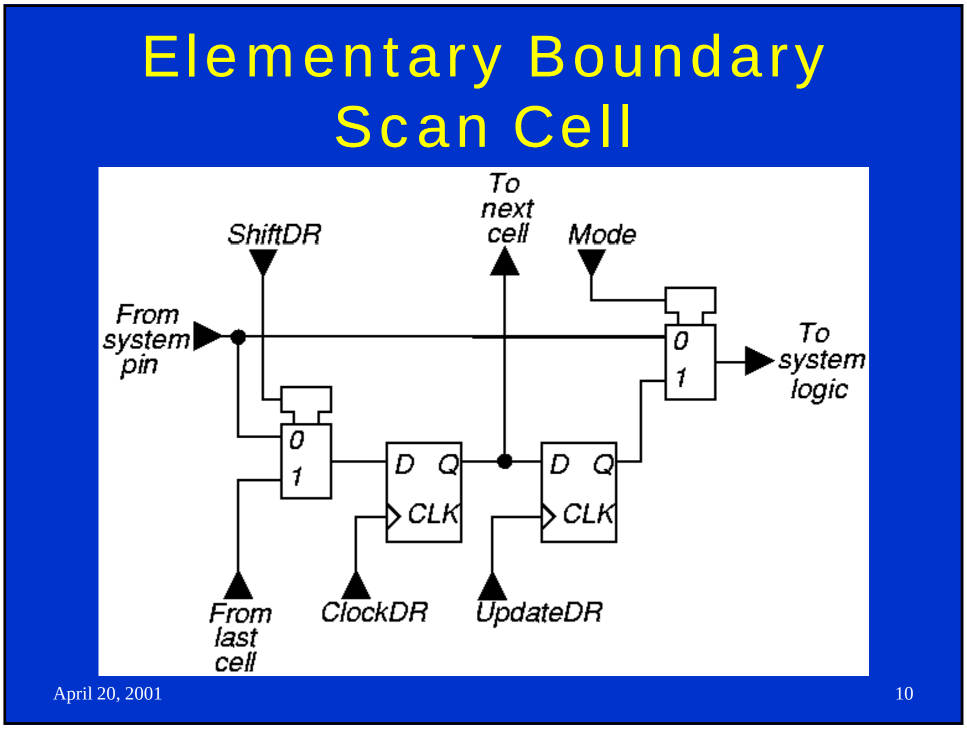
System View of Interconnect

April 20, 2001 8



Boundary Scan Chain View

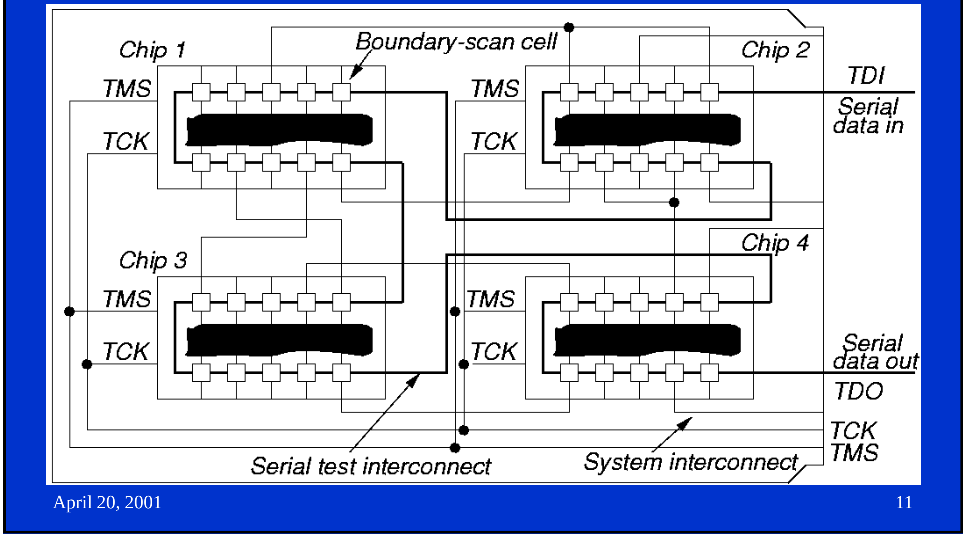
April 20, 2001 9



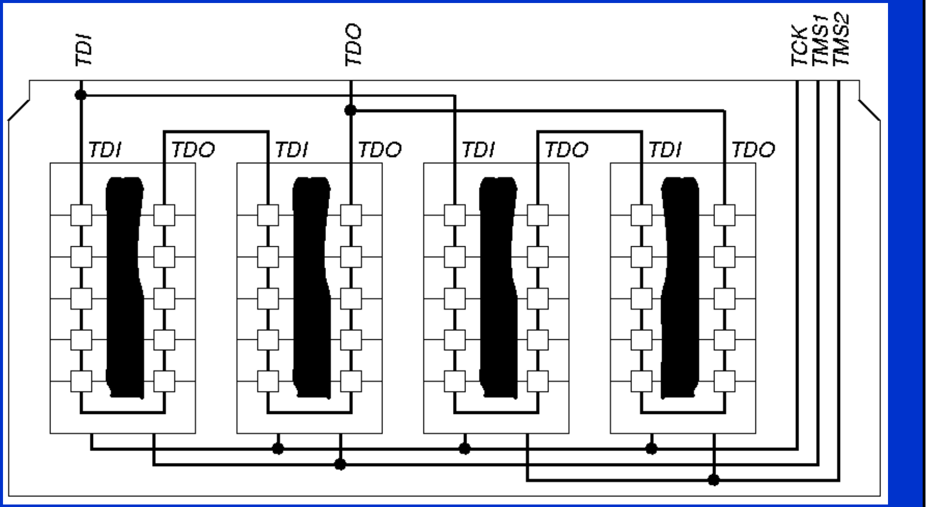
Elementary Boundary Scan Cell

April 20, 2001 10

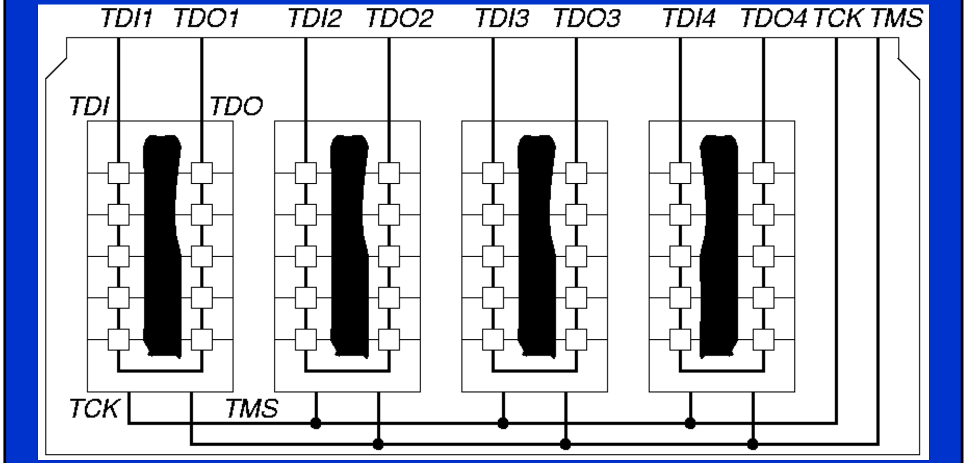
Serial Board / MCM Scan



Parallel Board / MCM Scan

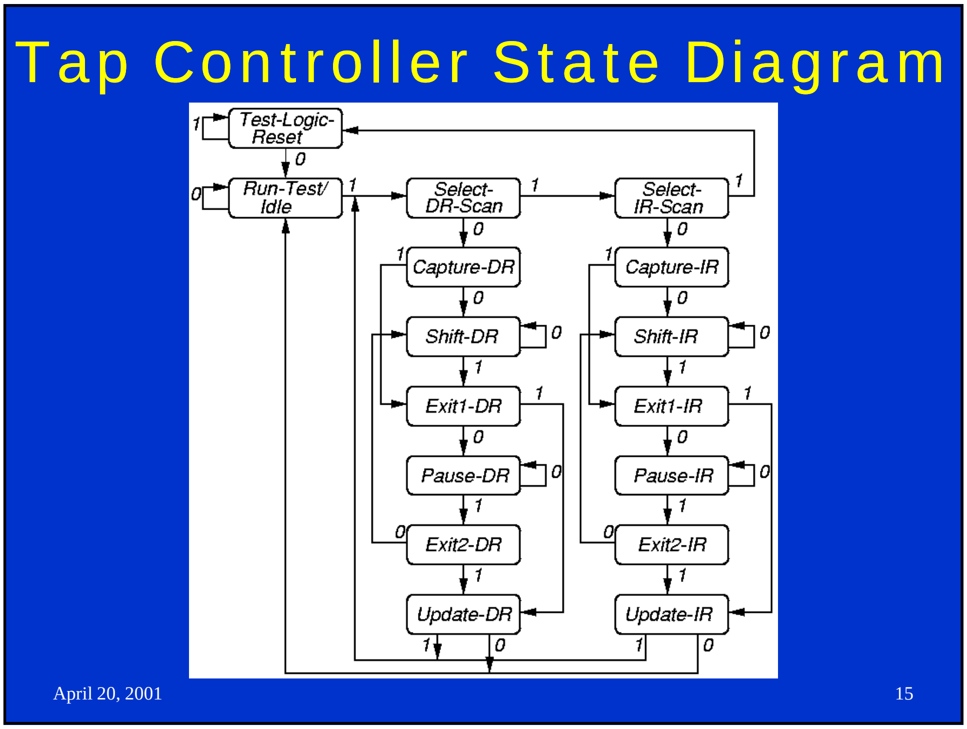


Independent Path Board / MCM Scan



Tap Controller Signals

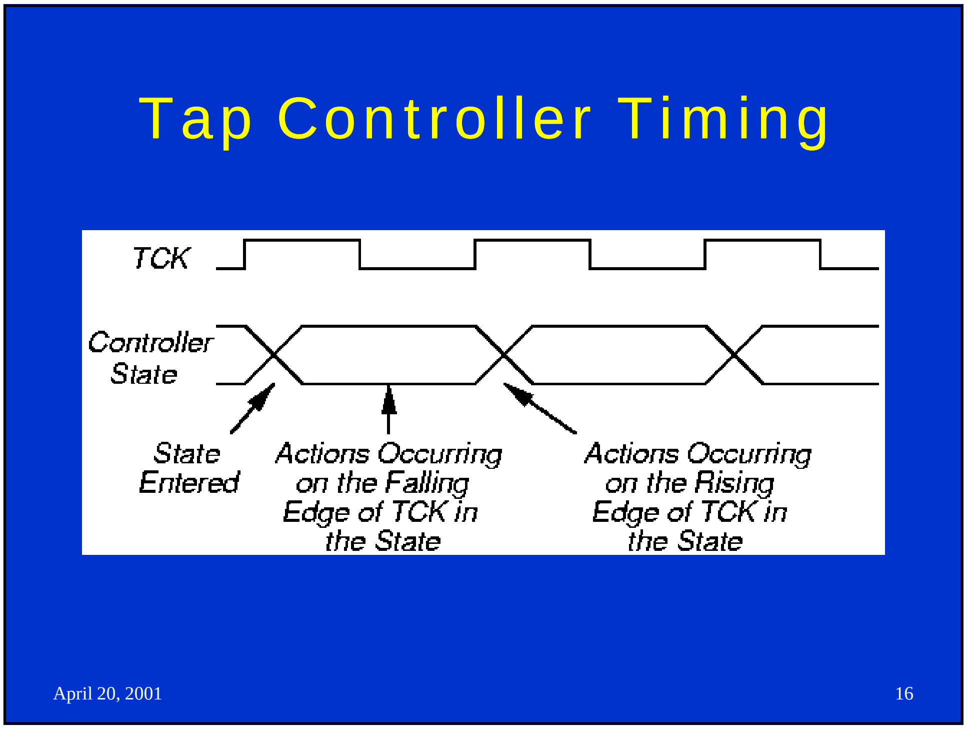
- *Test Access Port* (TAP) includes these signals:
  - *Test Clock Input* (*TCK*) -- Clock for test logic
    - Can run at different rate from system clock
  - *Test Mode Select* (*TMS*) -- Switches system from functional to test mode
  - *Test Data Input* (*TDI*) -- Accepts serial test data and instructions -- 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

April 20, 2001

15



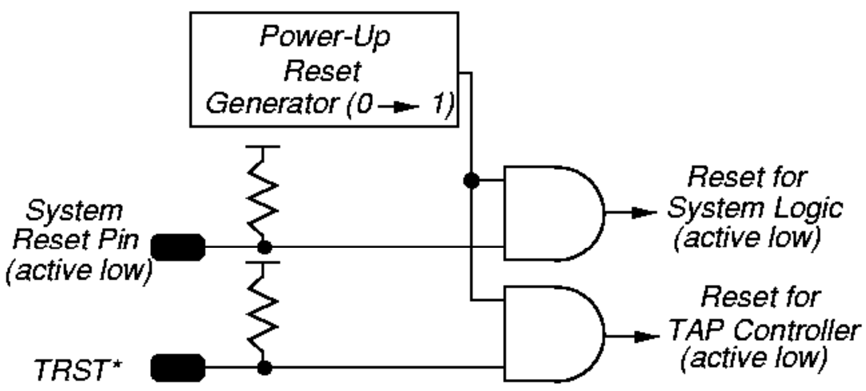
Tap Controller Timing

TCK
Controller State
State Entered
Actions Occurring on the Falling Edge of TCK in the State
Actions Occurring on the Rising Edge of TCK in the State

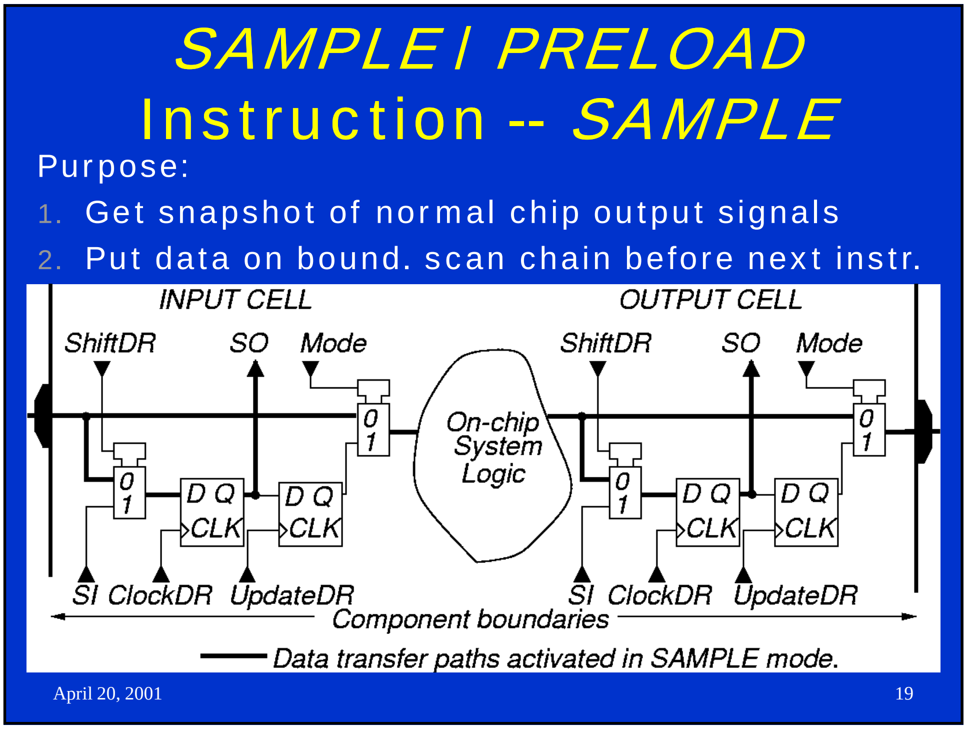
April 20, 2001

16

TAP Controller Power-Up Reset Logic



Boundary Scan Instructions

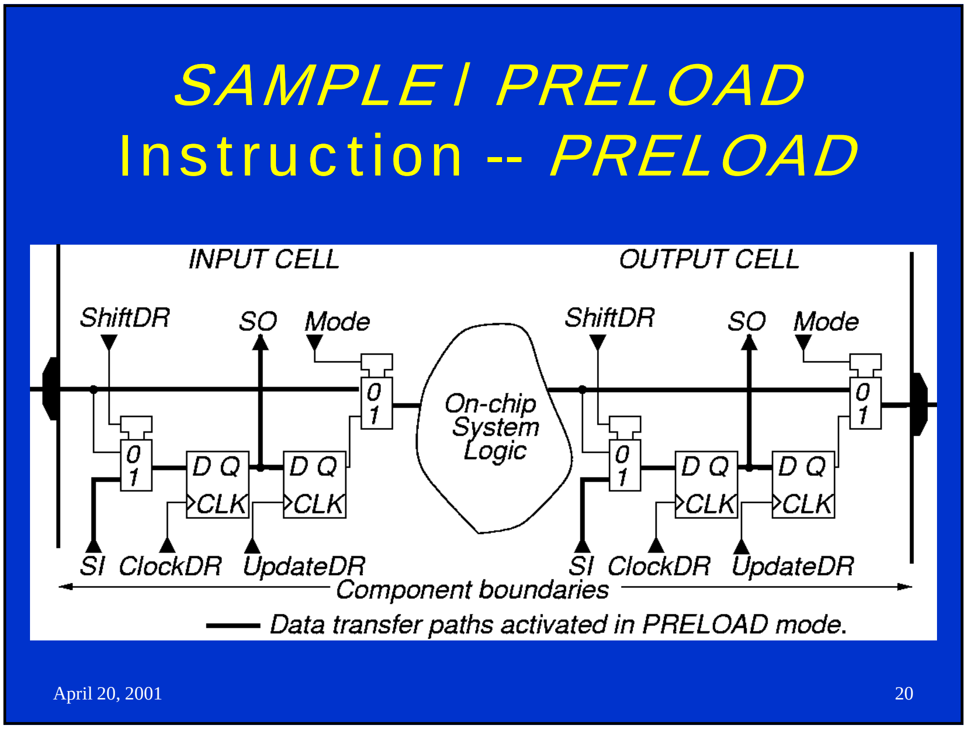


SAMPLE / PRELOAD Instruction -- *SAMPLE*

Purpose:

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

April 20, 2001 19

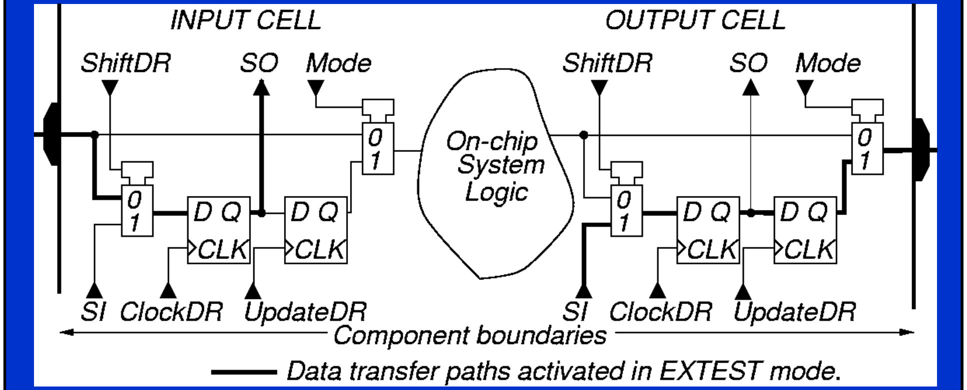


SAMPLE / PRELOAD Instruction -- *PRELOAD*

April 20, 2001 20

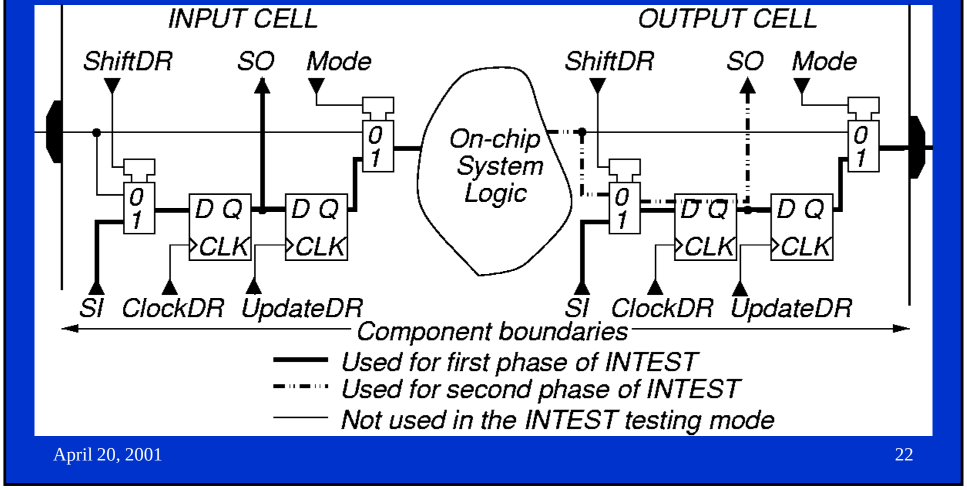
EXTEST Instruction

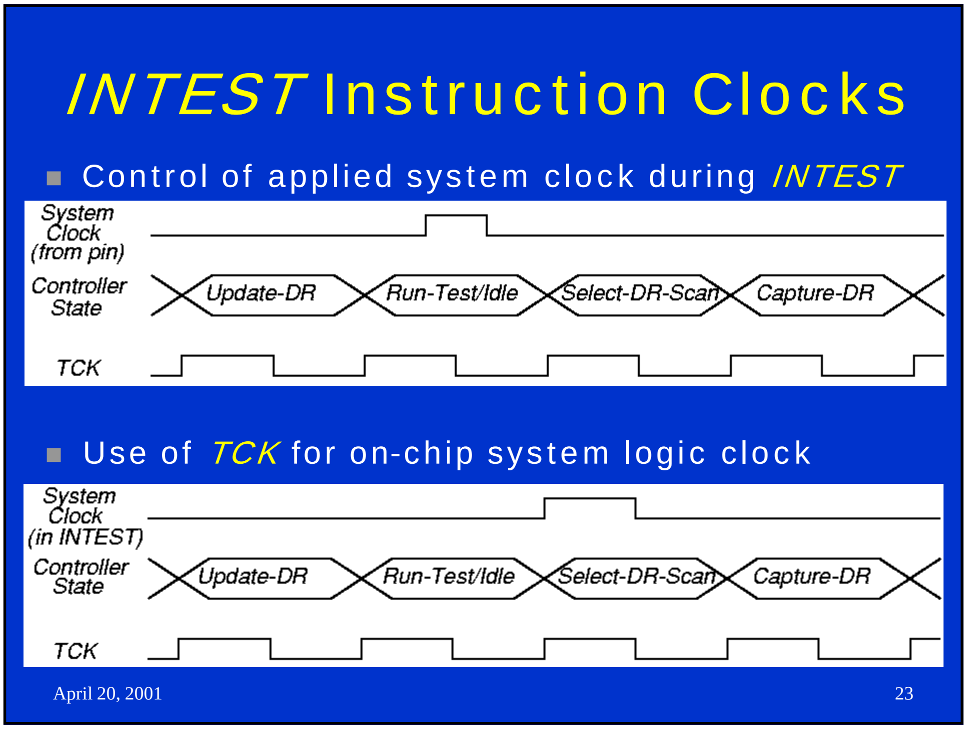
- Purpose: Test off-chip circuits and board-level interconnections



INTEST Instruction

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

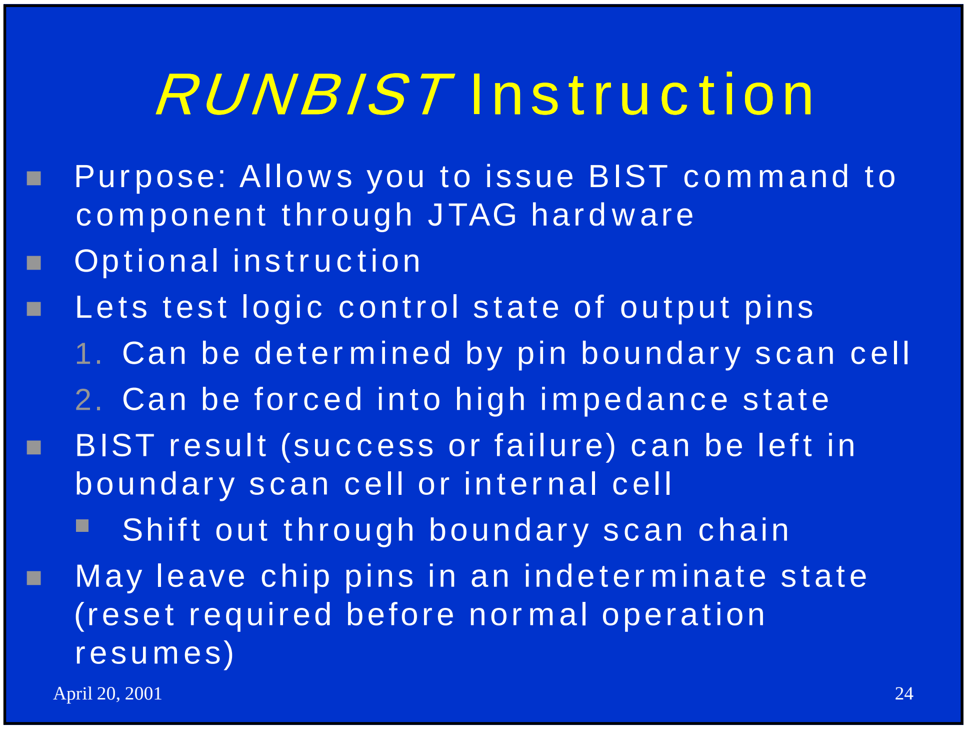




INTEST Instruction Clocks

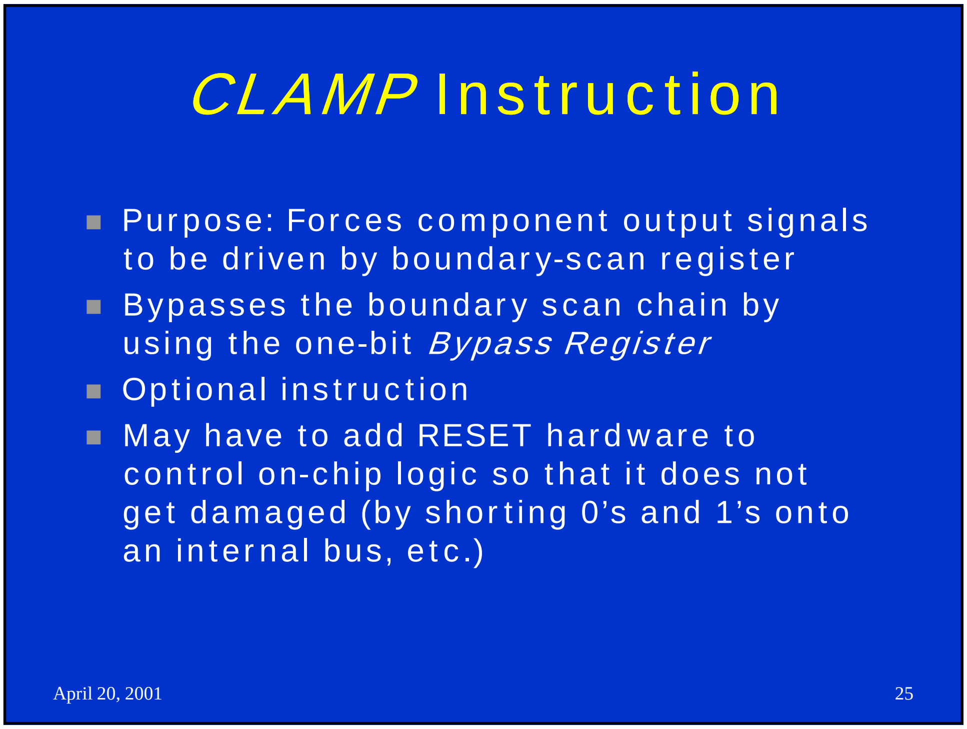
- Control of applied system clock during *INTEST*

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



RUNBIST Instruction

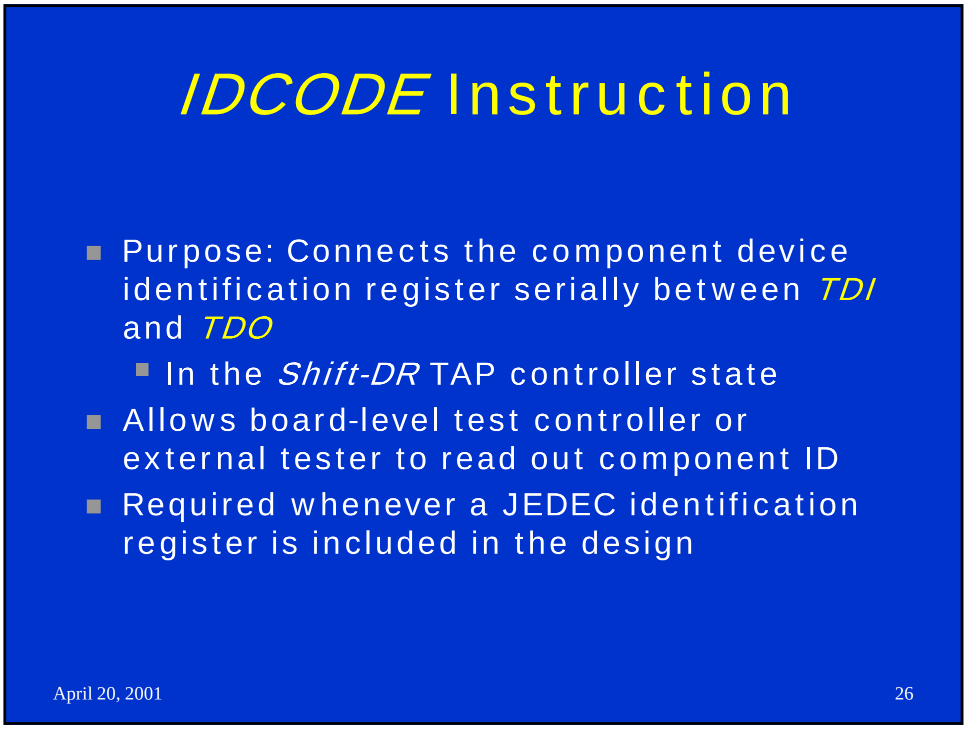
- Purpose: Allows you to issue BIST command to component through JTAG hardware
- Optional instruction
- Lets test logic control state of output pins
  1. Can be determined by pin boundary scan cell
  2. Can be forced into high impedance state
- BIST result (success or failure) can be left in boundary 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)



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.)

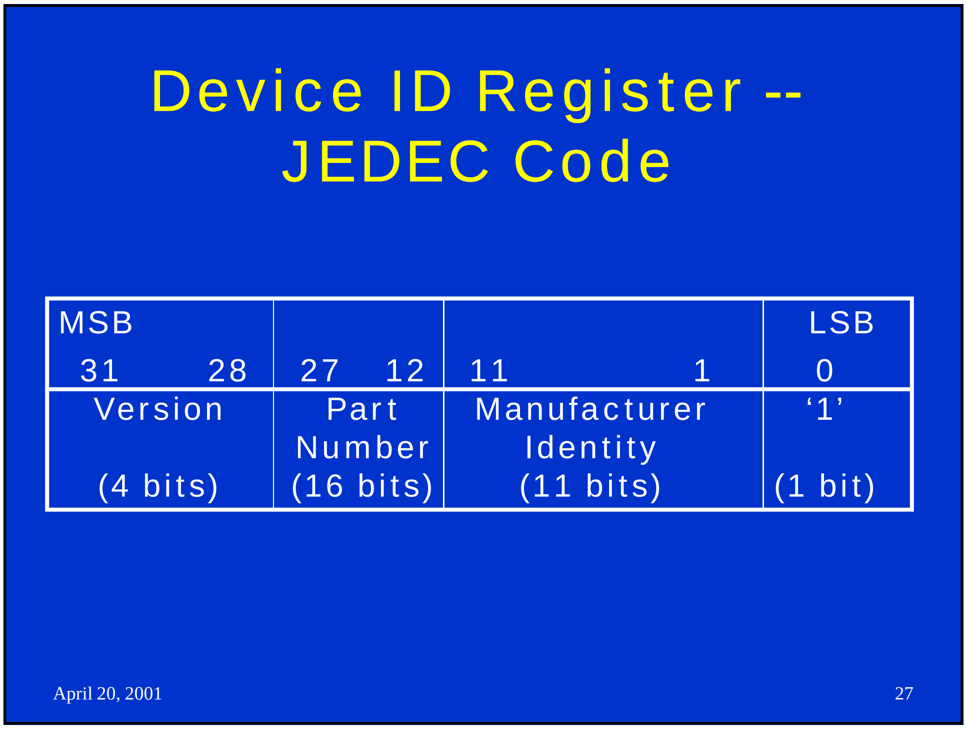
April 20, 2001 25



IDCODE Instruction

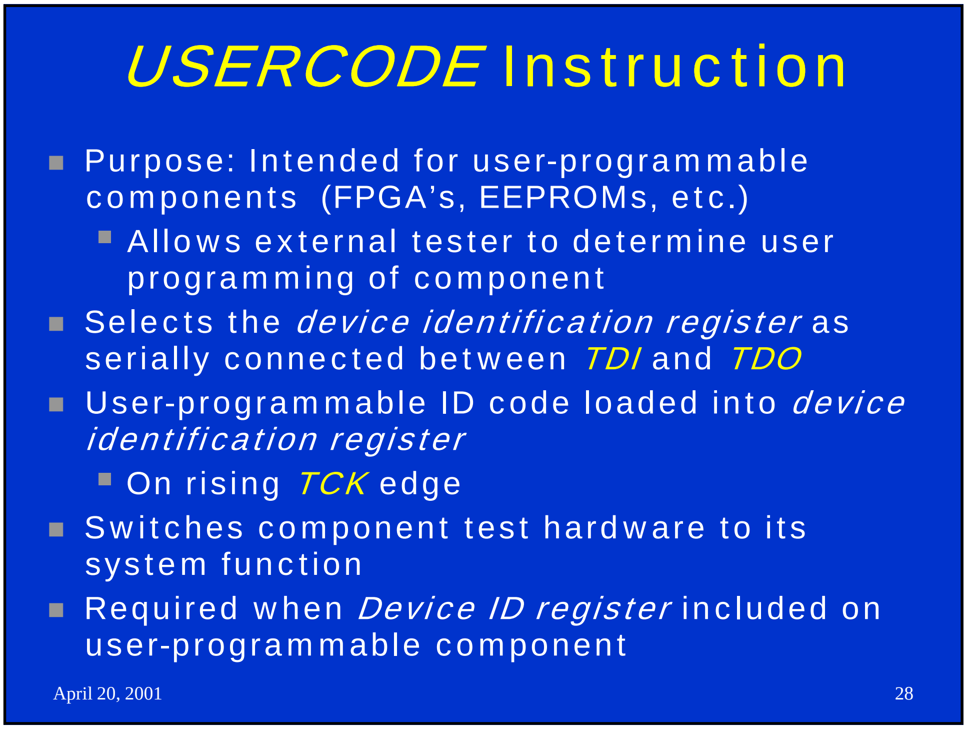
- Purpose: Connects the component device identification register serially between *TDI* and *TDO*
  - In the *Shift-DR* TAP controller state
- Allows board-level test controller or external tester to read out component ID
- Required whenever a JEDEC identification register is included in the design

April 20, 2001 26



Device ID Register -- JEDEC Code

MSB 31 28	27 12	11 1	LSB 0
Version (4 bits)	Part Number (16 bits)	Manufacturer Identity (11 bits)	‘1’ (1 bit)

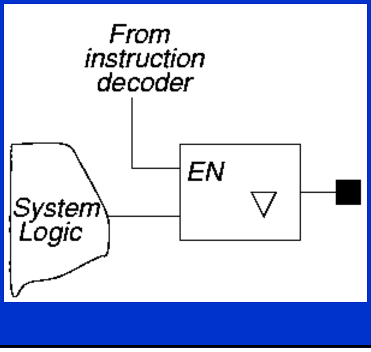


USERCODE Instruction

- 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

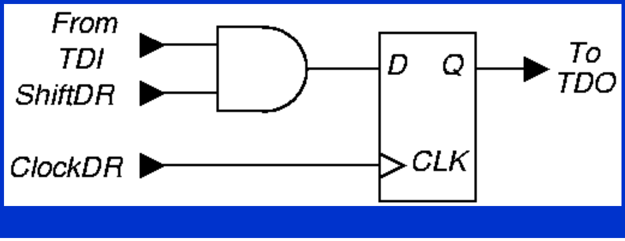
HIGHZ Instruction

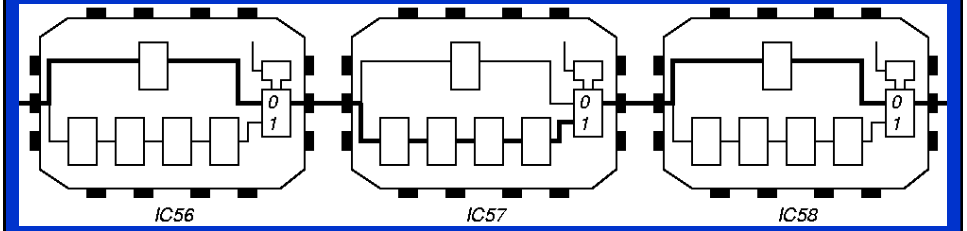
- Purpose: Puts all component output pin signals into high-impedance state
- Control chip logic to avoid damage in this mode
- May have to reset component after *HIGHZ* runs
- Optional instruction

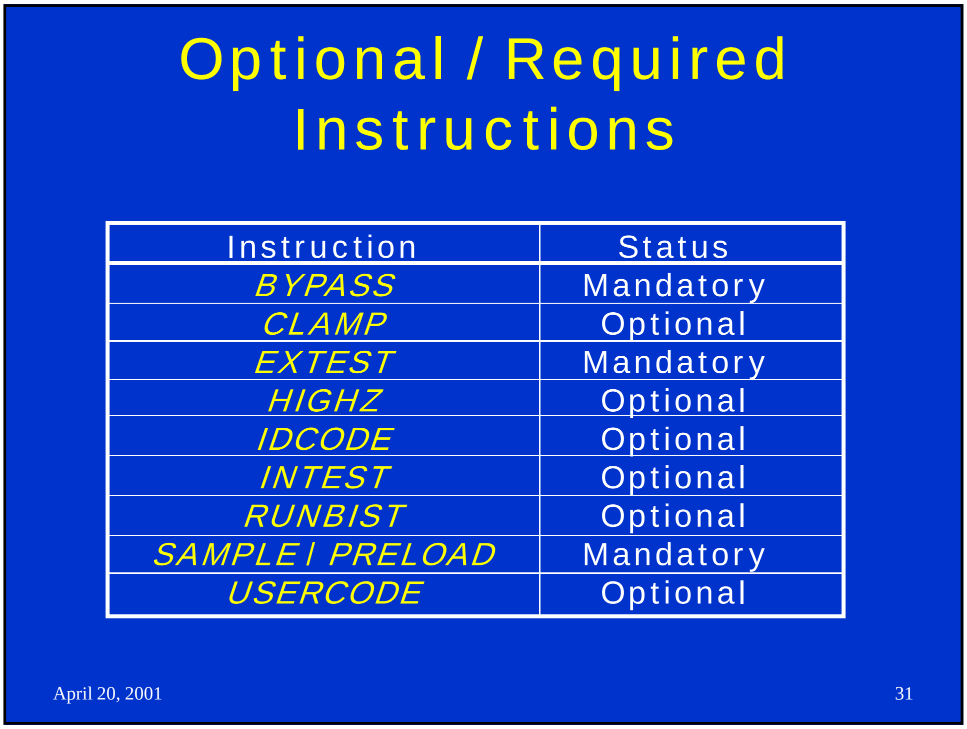


BYPASS Instruction

- Purpose: Bypasses scan chain with 1-bit register

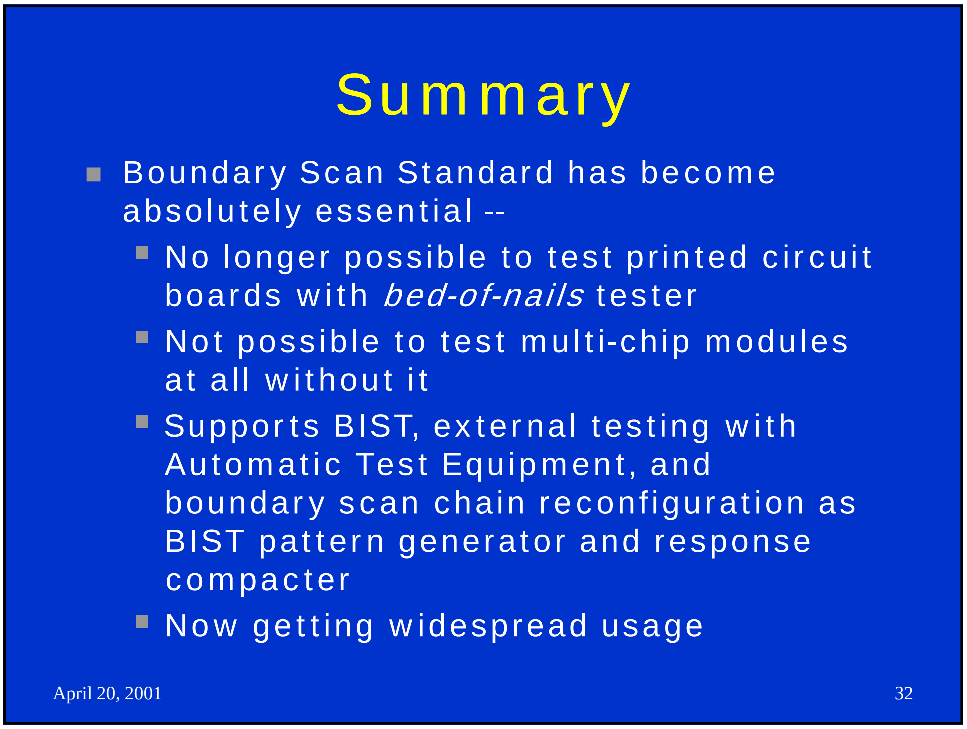






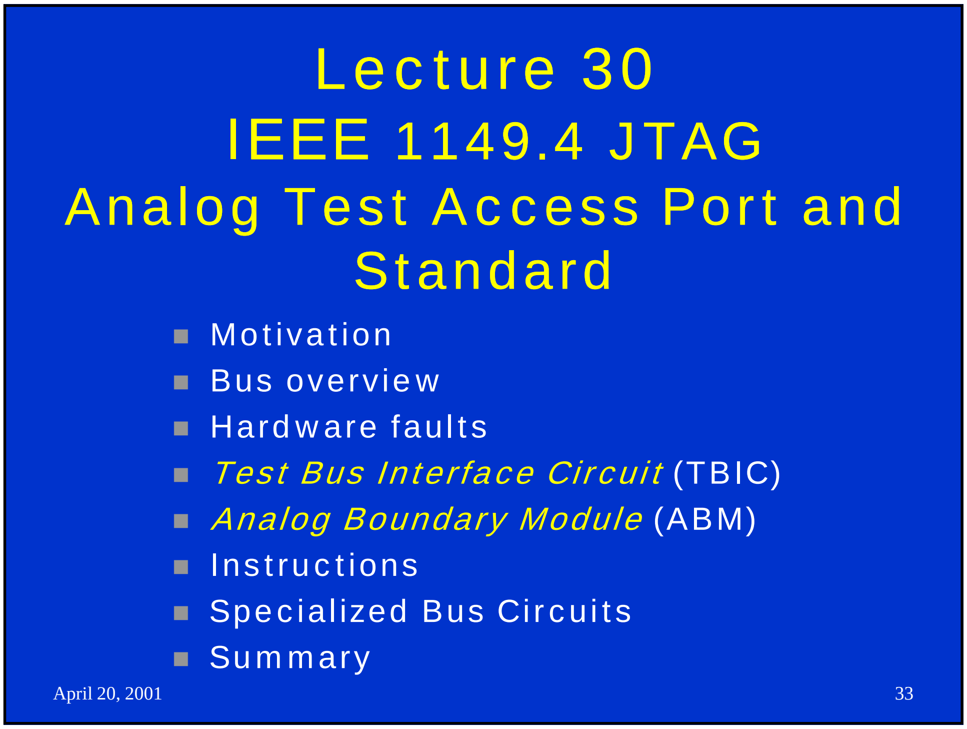
Optional / Required Instructions

Instruction	Status
BYPASS	Mandatory
CLAMP	Optional
EXTEST	Mandatory
HIGHZ	Optional
IDCODE	Optional
INTEST	Optional
RUNBIST	Optional
SAMPLE / PRELOAD	Mandatory
USERCODE	Optional



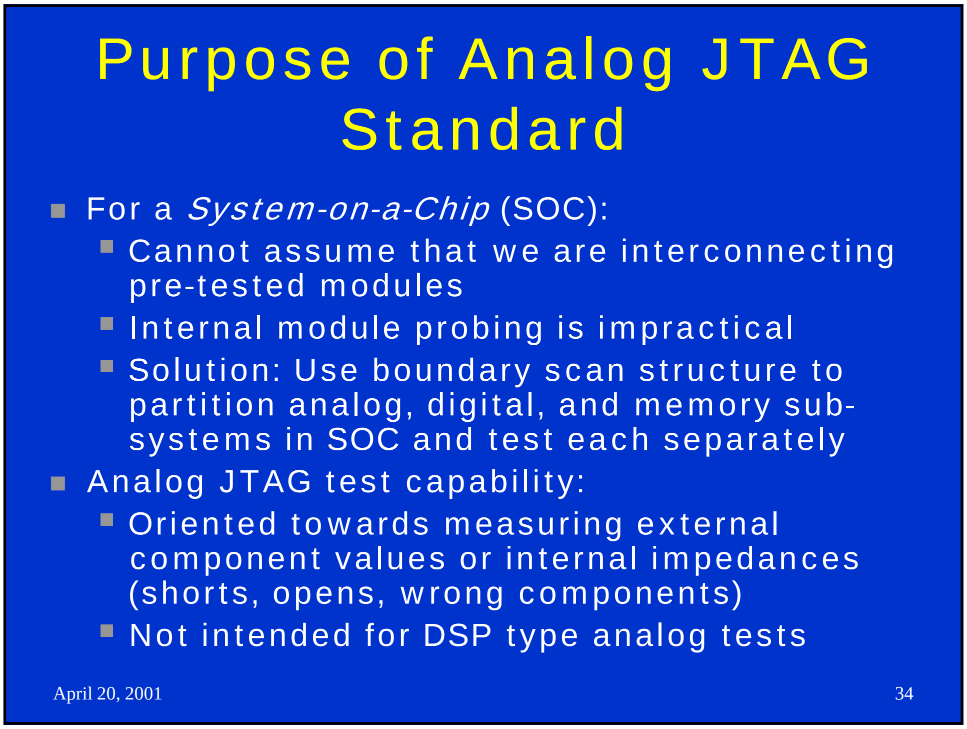
Summary

- 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



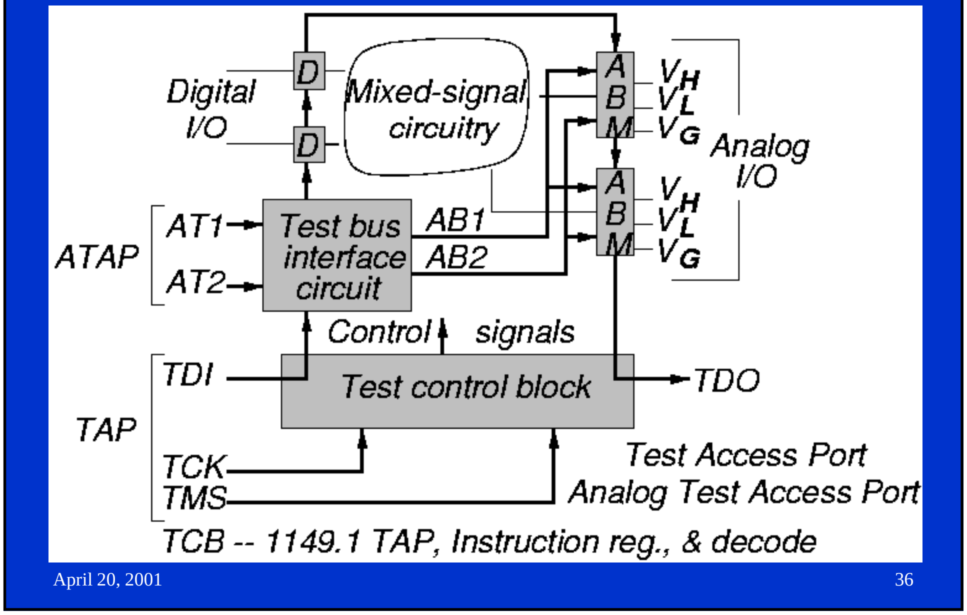
Purpose of Analog JTAG Standard

- 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 sub-systems 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

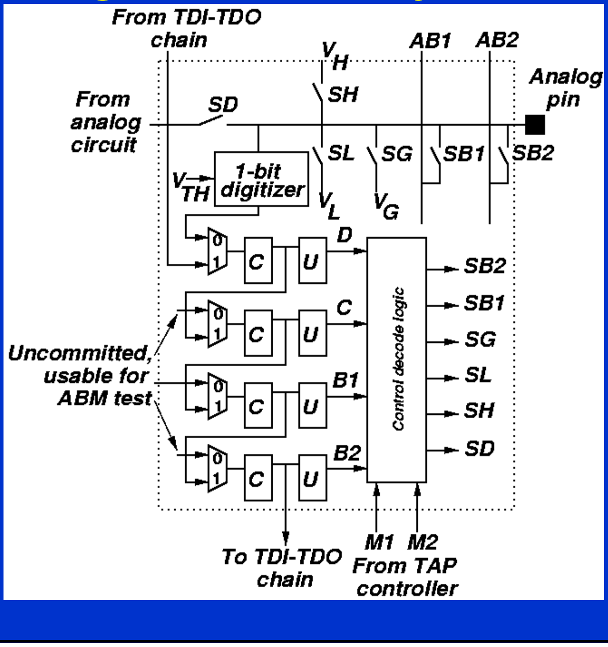
- PROs:
  - Usable with digital JTAG boundary scan
  - Adds analog testability – both controllability and observability
  - Eliminates large area needed for analog test points
- CONs:
  - May have a 5 % measurement error
  - C-switch sampling devices couple all probe points capacitively, even with test 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



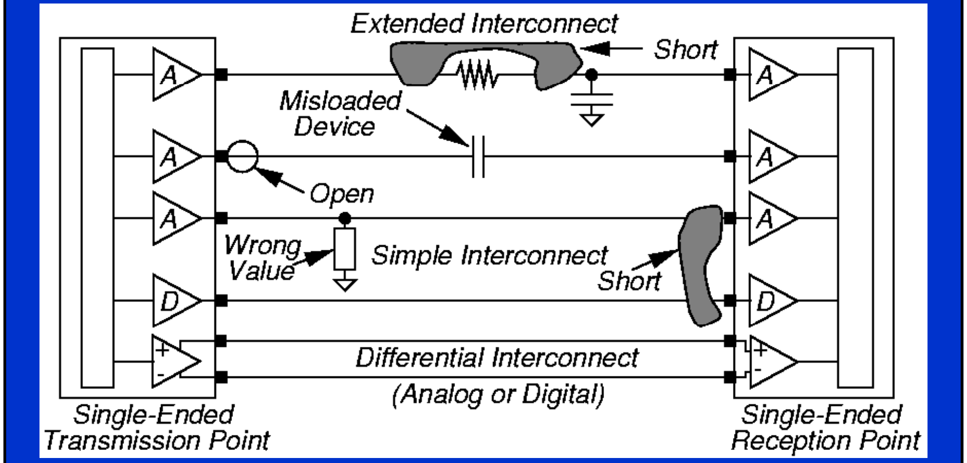
TCB -- 1149.1 TAP, Instruction reg., & decode

Analog Boundary Module

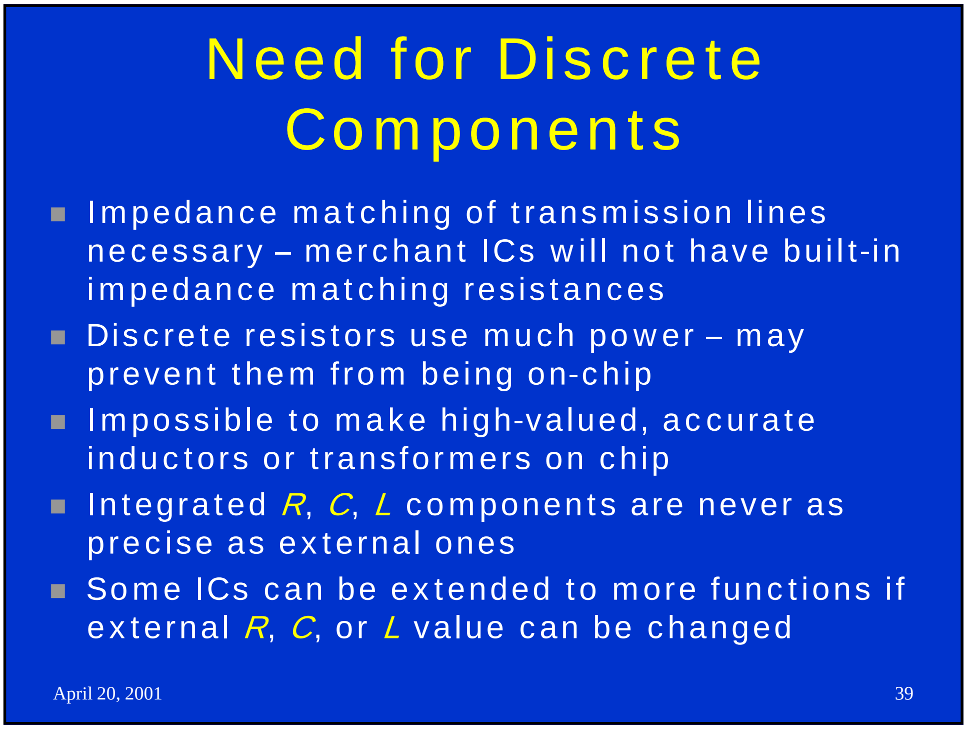


April 20, 2001 37

Analog Defects and Faults

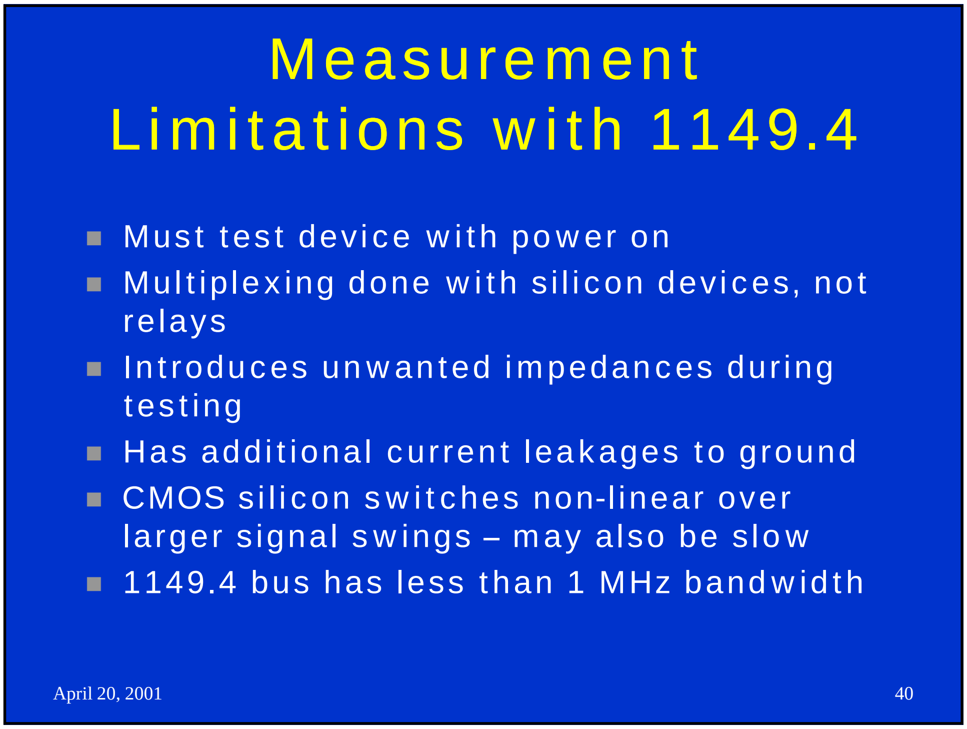


April 20, 2001 38



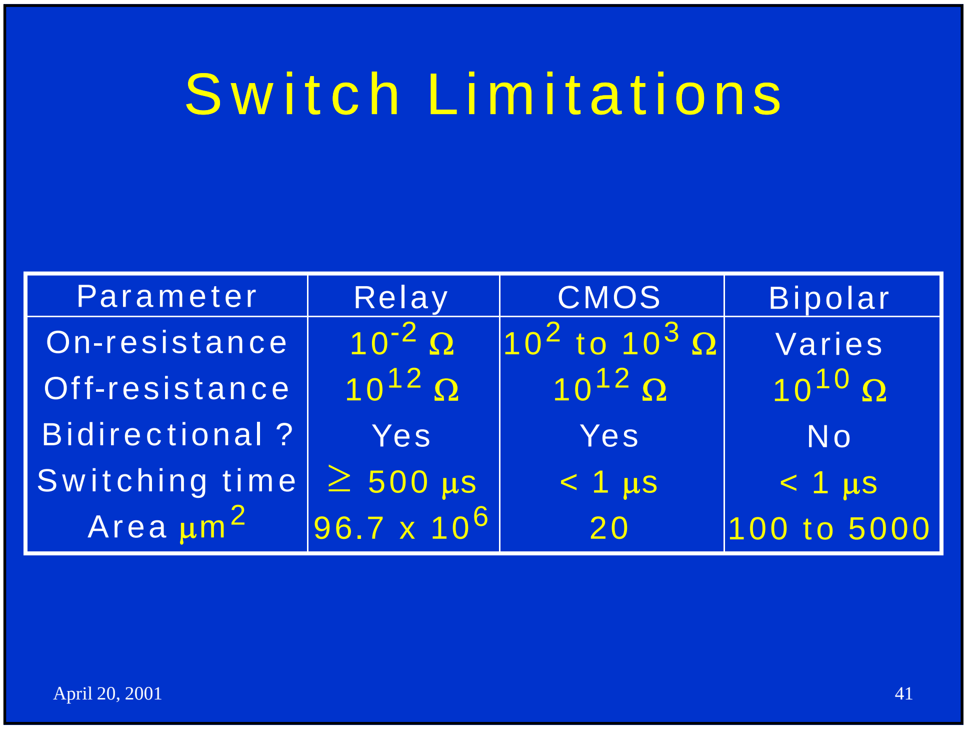
Need for Discrete Components

- 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



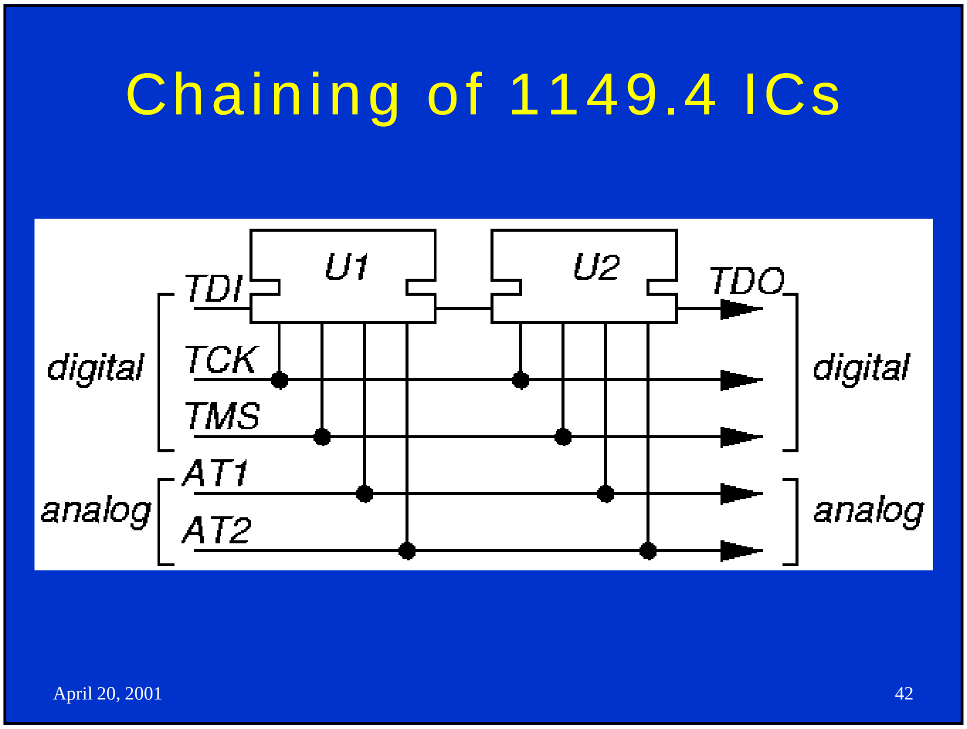
Measurement Limitations with 1149.4

- 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

Parameter	Relay	CMOS	Bipolar
On-resistance	$10^{-2}\ \Omega$	10^{2} to $10^{3}\ \Omega$	Varies
Off-resistance	$10^{12}\ \Omega$	$10^{12}\ \Omega$	$10^{10}\ \Omega$
Bidirectional ?	Yes	Yes	No
Switching time	$\geq$ 500 μs	< 1 μs	< 1 μs
Area μm^2	96.7×10^{6}	20	100 to 5000

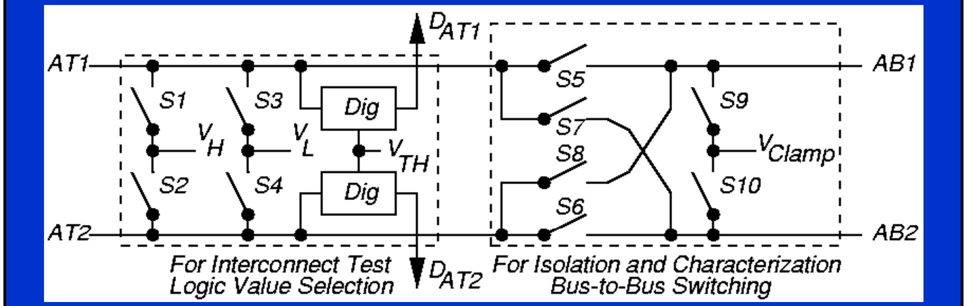


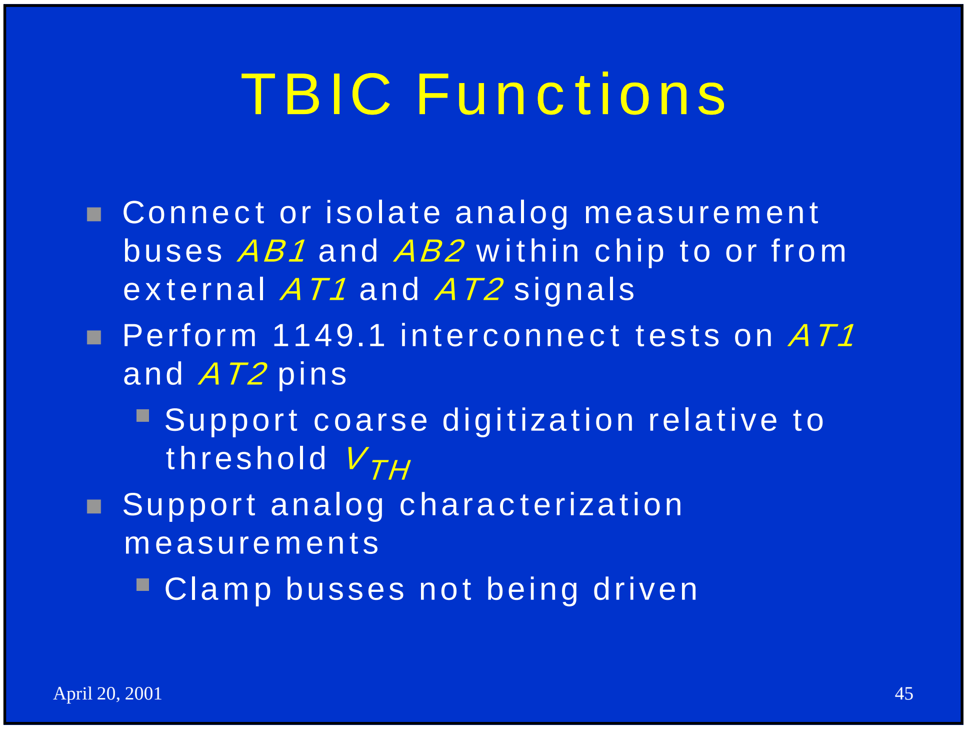
Chaining of 1149.4 ICs

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)

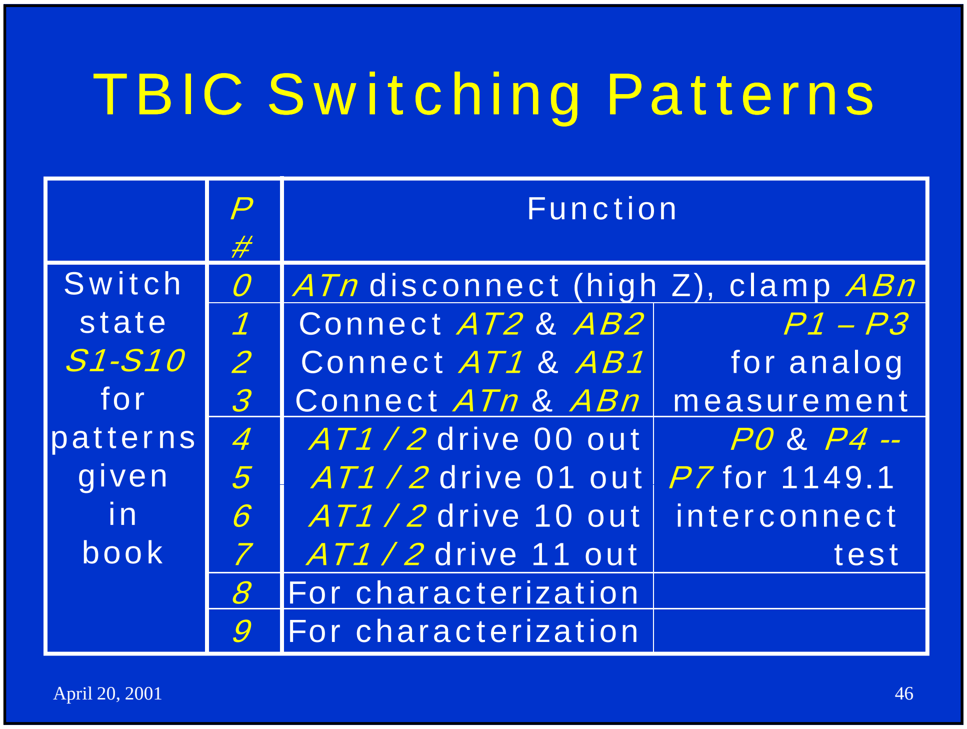
Test Bus Interface Circuit





TBIC Functions

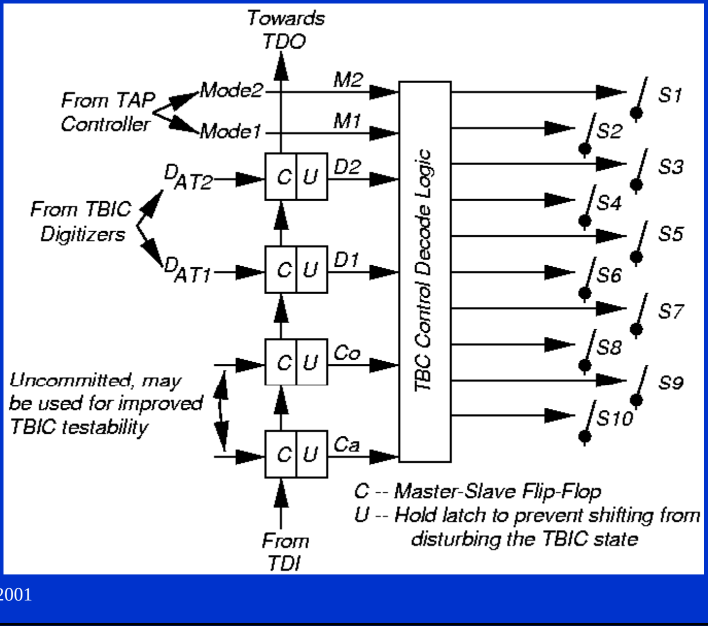
- Connect or isolate analog measurement buses *AB1* and *AB2* within chip to or from external *AT1* and *AT2* signals
- Perform 1149.1 interconnect tests on *AT1* and *AT2* pins
  - Support coarse digitization relative to threshold V_{TH}
- Support analog characterization measurements
  - Clamp busses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



TBIC Switching Patterns

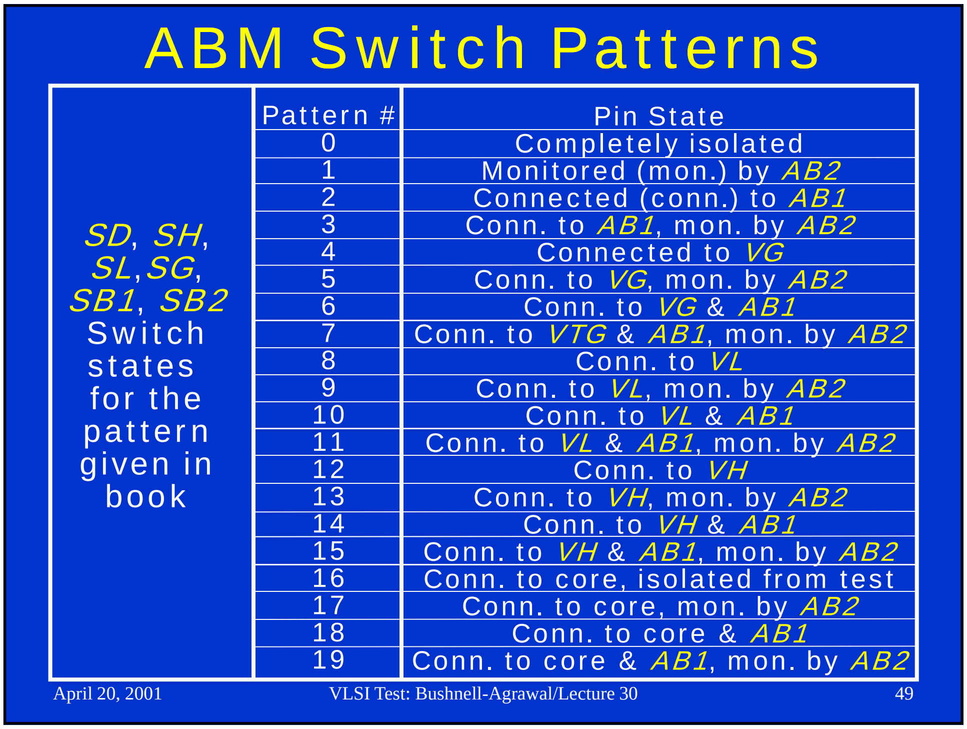
	P #	Function	
Switch state *S1-S10* for patterns given in book	*0*	*ATn* disconnect (high Z), clamp *ABn*	
	1	Connect *AT2* & *AB2*	*P1 – P3* for analog measurement
	2	Connect *AT1* & *AB1*	
	3	Connect *ATn* & *ABn*	
	4	*AT1 / 2* drive 00 out	*P0* & *P4* -- *P7* for 1149.1 interconnect test
	5	*AT1 / 2* drive 01 out	
	6	*AT1 / 2* drive 10 out	
	7	*AT1 / 2* drive 11 out	
	8	For characterization	
	9	For characterization	

TBIC Switch Controls



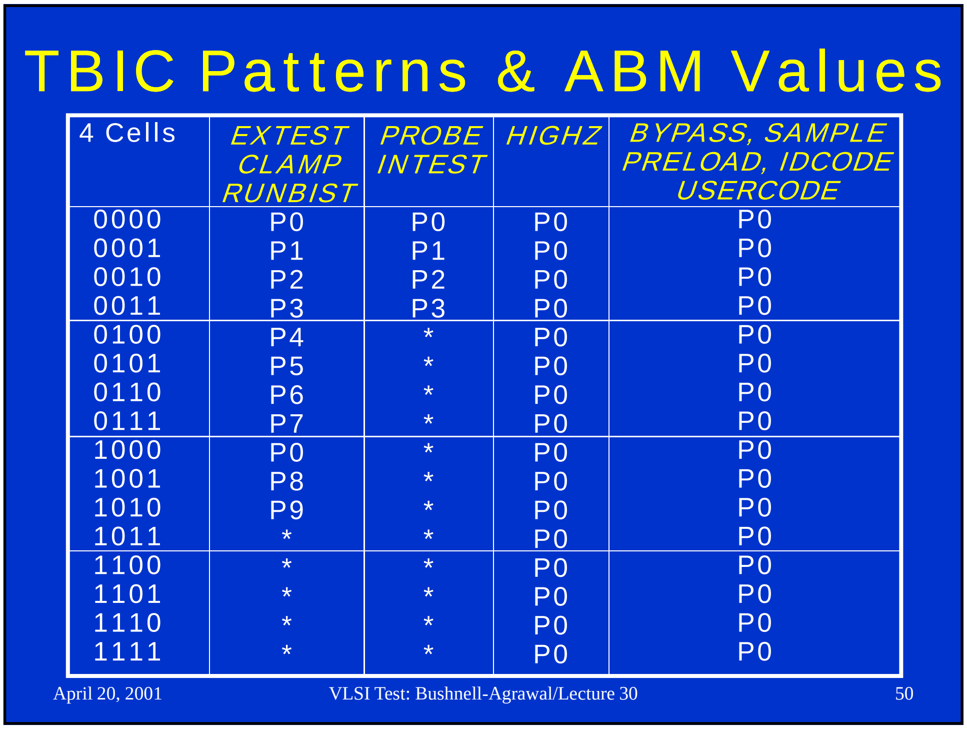
Analog Boundary Module Has Four Control Cells

- 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

SD, *SH*, *SL*,*SG*, *SB1*, *SB2* Switch states for the pattern given in book	Pattern #	Pin State
	0	Completely isolated
	1	Monitored (mon.) by *AB2*
	2	Connected (conn.) to *AB1*
	3	Conn. to *AB1*, mon. by *AB2*
	4	Connected to *VG*
	5	Conn. to *VG*, mon. by *AB2*
	6	Conn. to *VG* & *AB1*
	7	Conn. to *VTG* & *AB1*, mon. by *AB2*
	8	Conn. to *VL*
	9	Conn. to *VL*, mon. by *AB2*
	10	Conn. to *VL* & *AB1*
	11	Conn. to *VL* & *AB1*, mon. by *AB2*
	12	Conn. to *VH*
	13	Conn. to *VH*, mon. by *AB2*
	14	Conn. to *VH* & *AB1*
	15	Conn. to *VH* & *AB1*, mon. by *AB2*
	16	Conn. to core, isolated from test
	17	Conn. to core, mon. by *AB2*
	18	Conn. to core & *AB1*
	19	Conn. to core & *AB1*, mon. by *AB2*



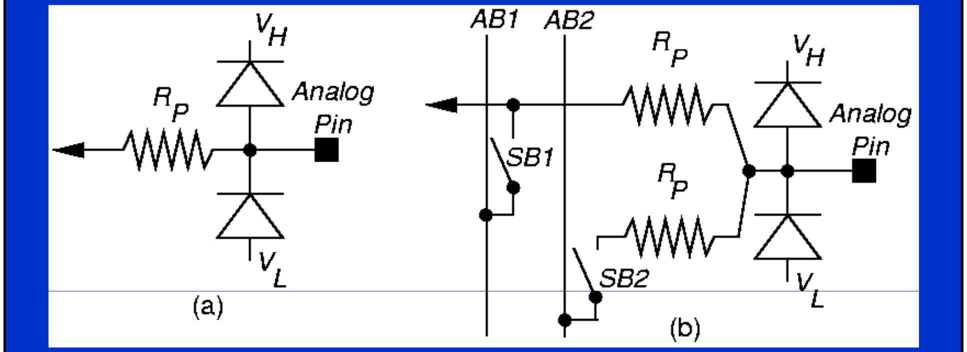
TBIC Patterns & ABM Values

4 Cells	*EXTEST CLAMP RUNBIST*	*PROBE INTEST*	*HIGHZ*	*BYPASS, SAMPLE PRELOAD, IDCODE USERCODE*
0000	P0	P0	P0	P0
0001	P1	P1	P0	P0
0010	P2	P2	P0	P0
0011	P3	P3	P0	P0
0100	P4	*	P0	P0
0101	P5	*	P0	P0
0110	P6	*	P0	P0
0111	P7	*	P0	P0
1000	P0	*	P0	P0
1001	P8	*	P0	P0
1010	P9	*	P0	P0
1011	*	*	P0	P0
1100	*	*	P0	P0
1101	*	*	P0	P0
1110	*	*	P0	P0
1111	*	*	P0	P0

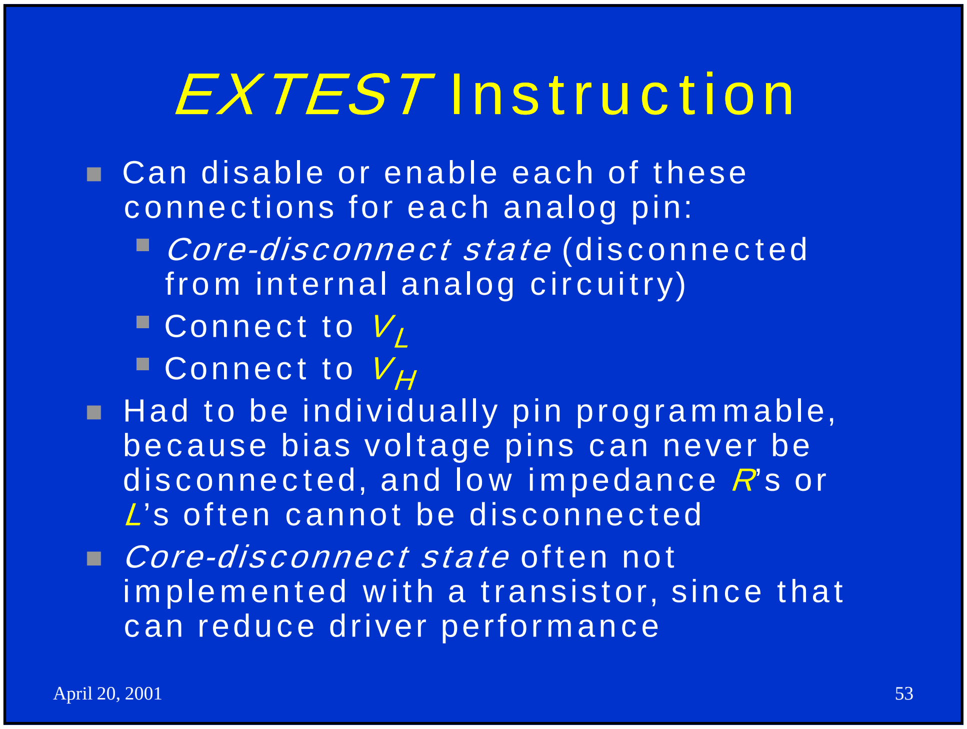
Analog Boundary Module Functions

- 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

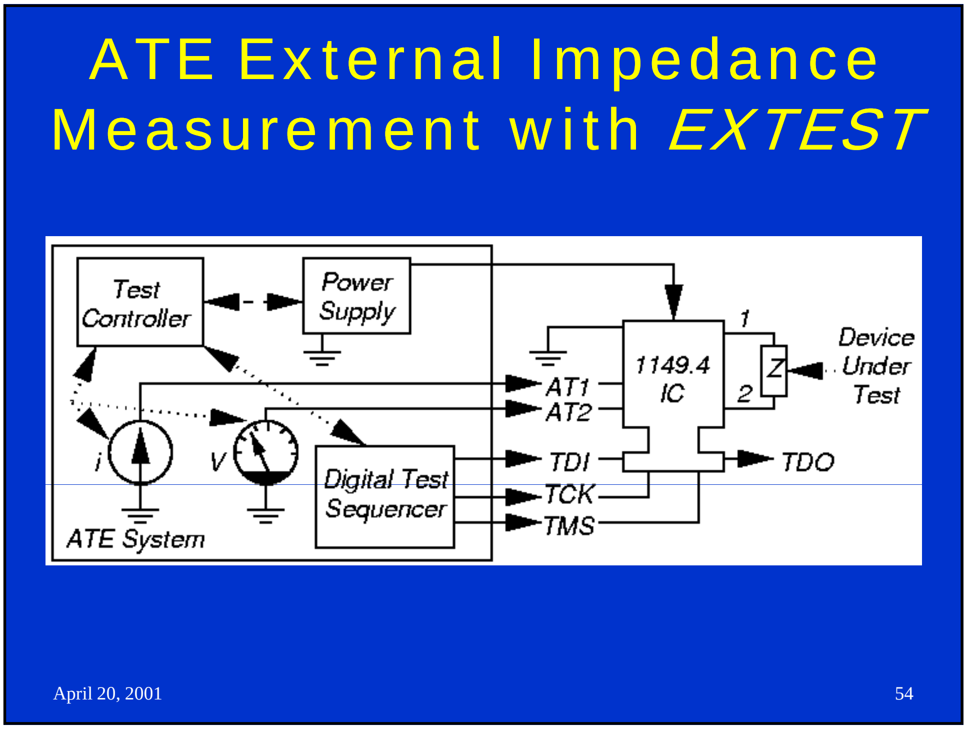


(a) Ordinary pin (b) ABM pin



EXTEST Instruction

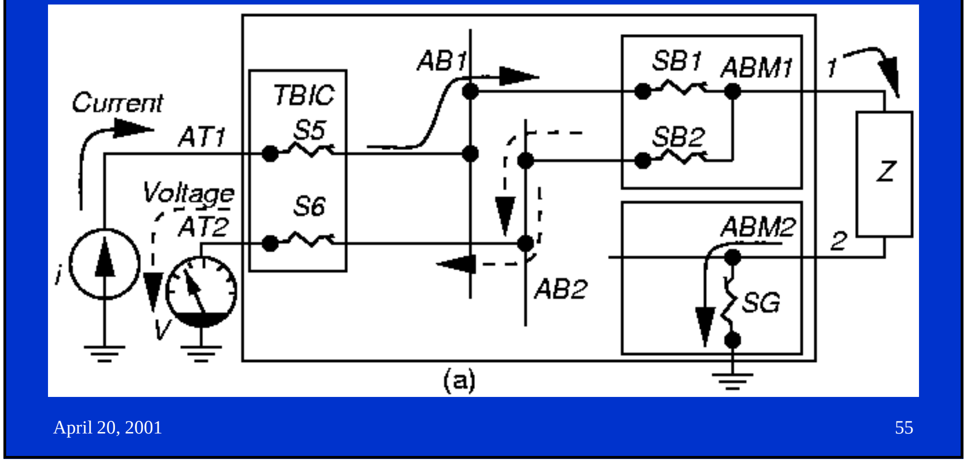
- Can disable or enable each of these connections for each analog pin:
  - *Core-disconnect state* (disconnected from internal analog circuitry)
  - Connect to V_L
  - Connect to V_H
- Had to be individually pin programmable, because bias voltage pins can never be disconnected, and low impedance *R*'s 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*

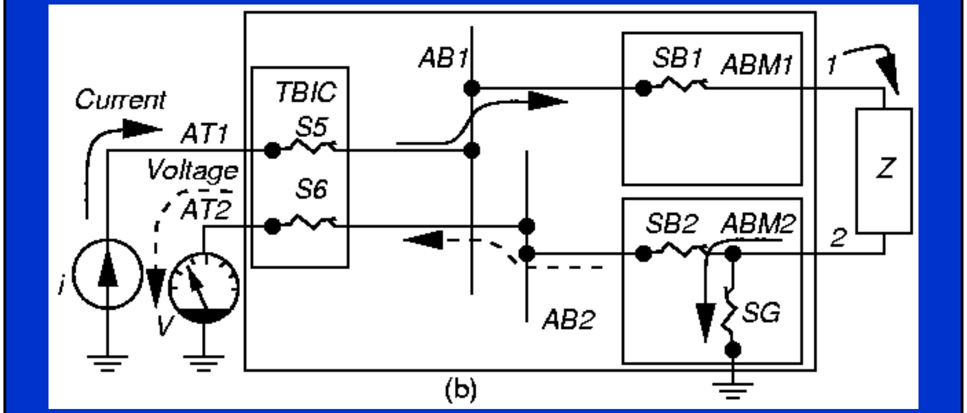
1149.4 Measurement of External Impedance

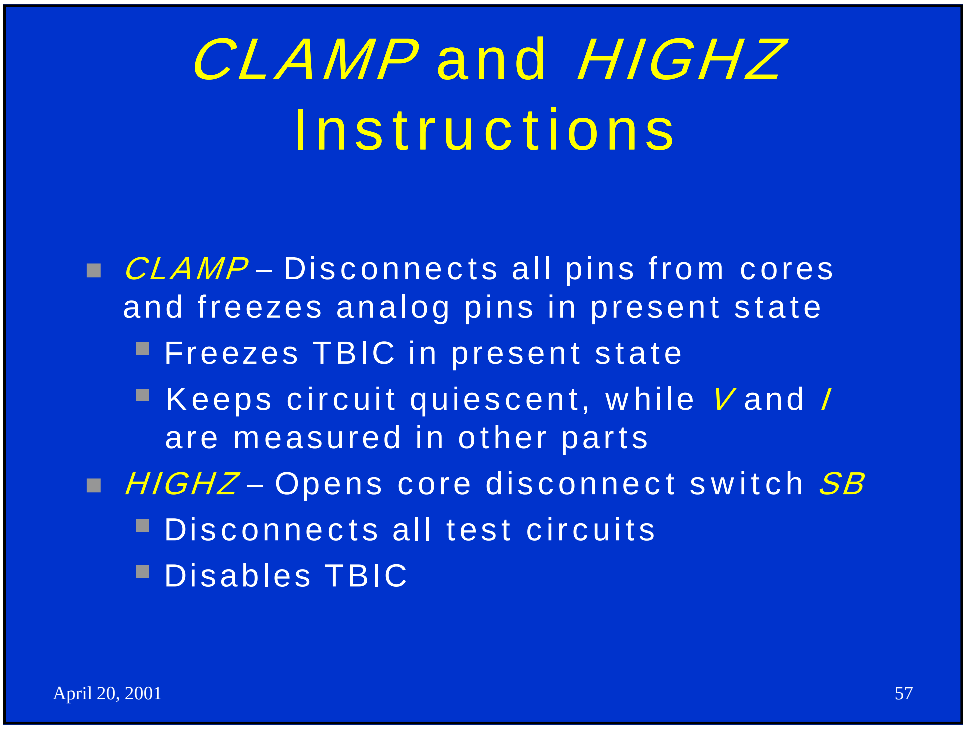
- (a) Pin 1 voltage measurement



Pin 2 Voltage Measurement

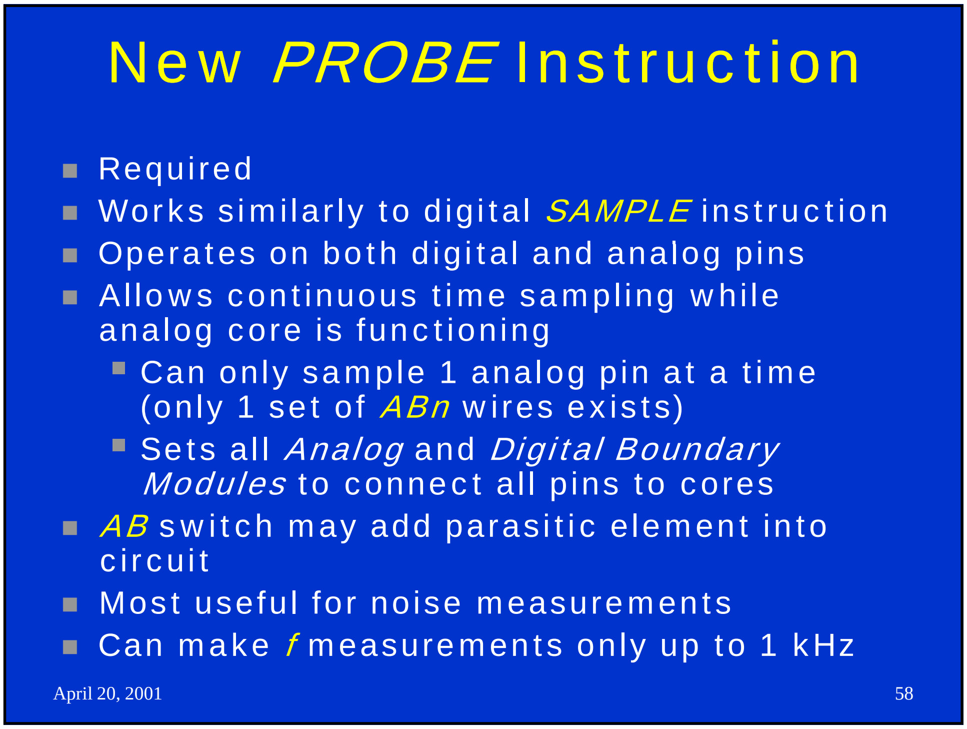
$$Z = \frac{V_{Pin1} - V_{Pin2}}{I}$$





CLAMP and *HIGHZ* Instructions

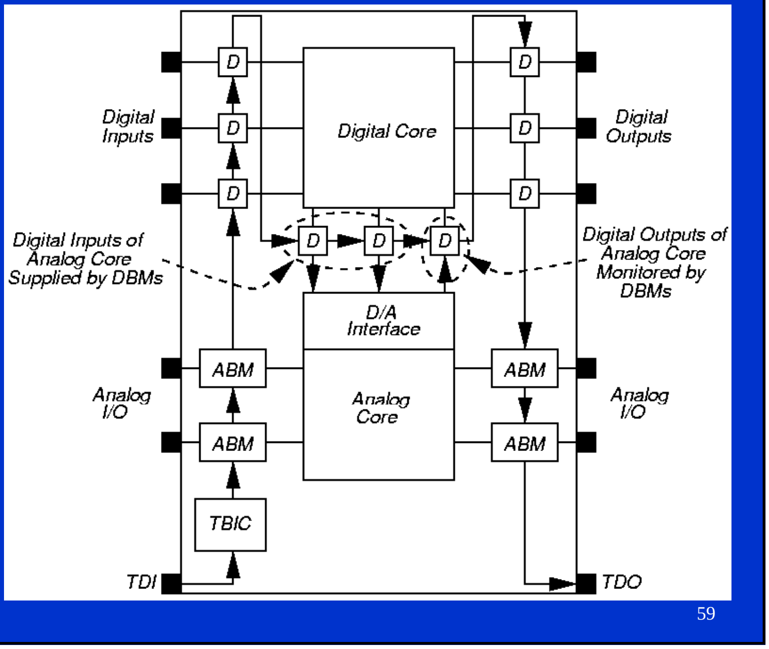
- *CLAMP* – Disconnects all pins from cores and freezes analog pins in present state
  - Freezes TBIC in present state
  - Keeps circuit quiescent, while *V* and *I* are measured in other parts
- *HIGHZ* – Opens core disconnect switch *SB*
  - Disconnects all test circuits
  - Disables TBIC



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

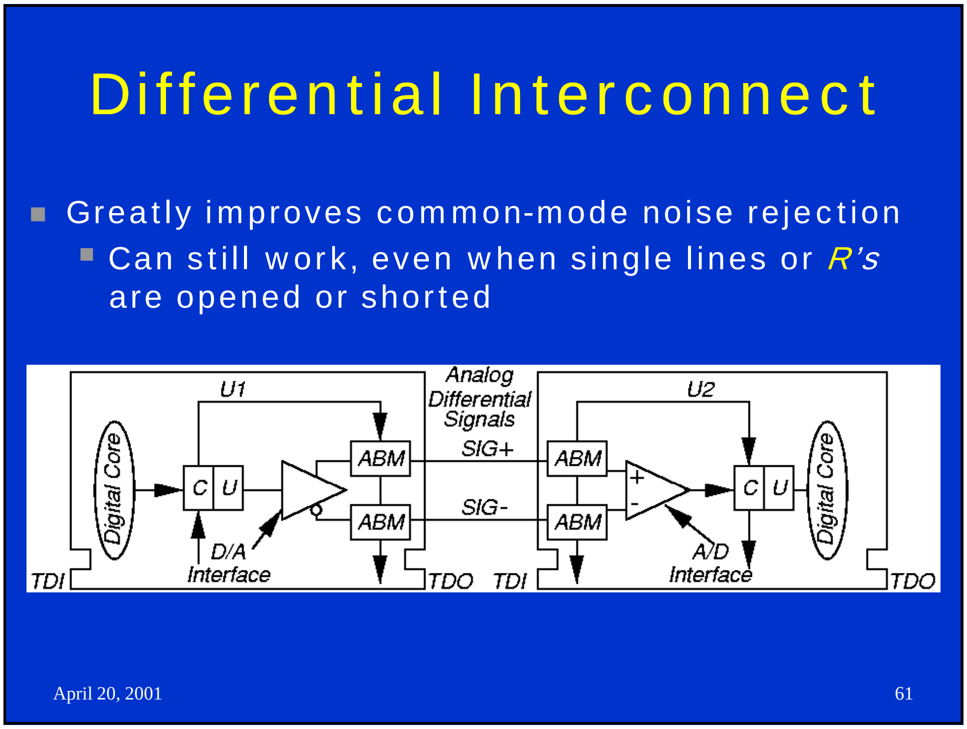
INTEST Instruction



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

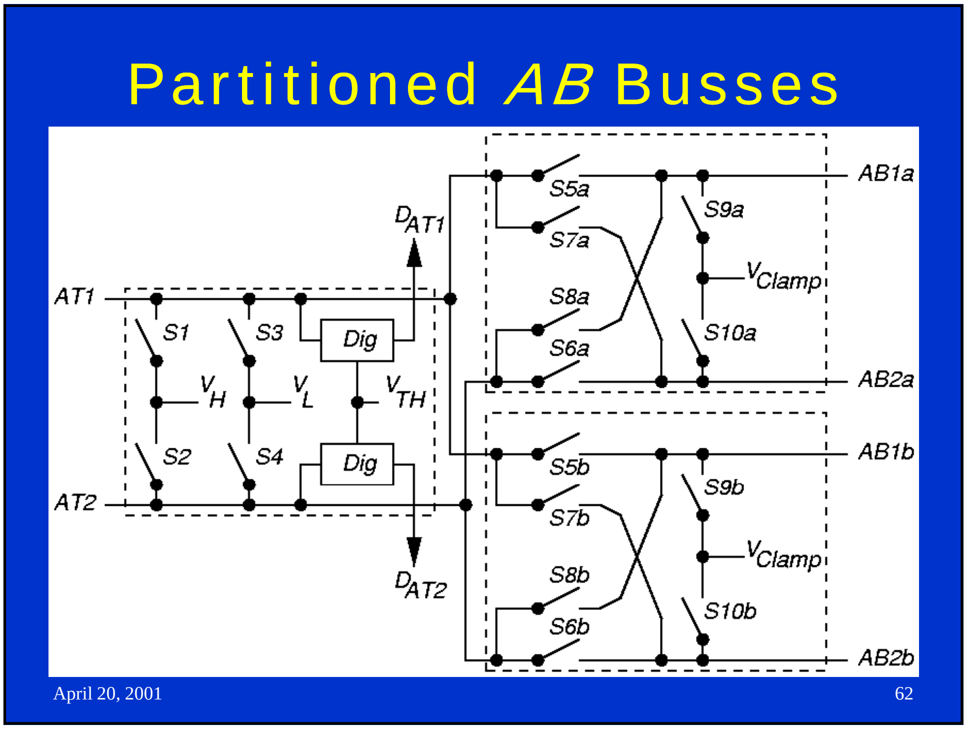
RUNBIST and SAMPLE / PRELOAD Instructions

- *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

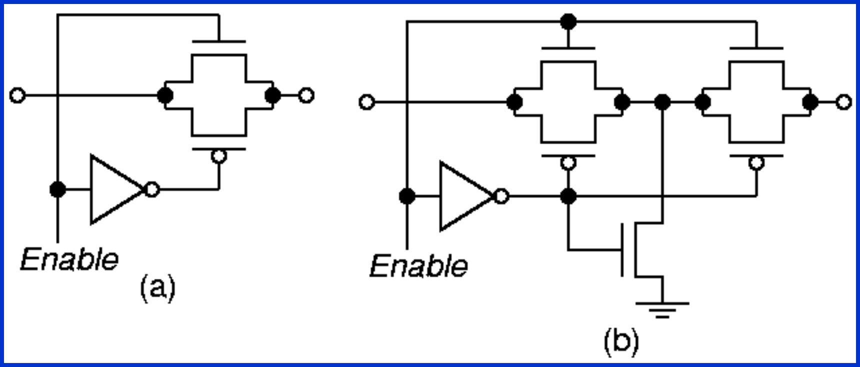


Partitioned *AB* Busses

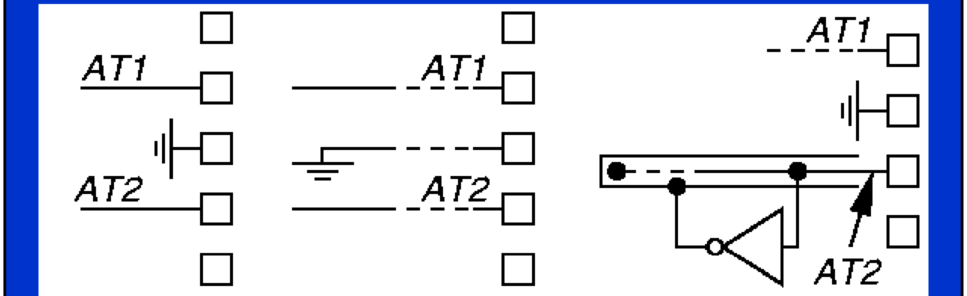
Isolation of Analog and Digital Cores

- 1149.4 standard requires that a digital boundary module be on each digital line between digital and analog core
  - Only when *INTEST* or *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



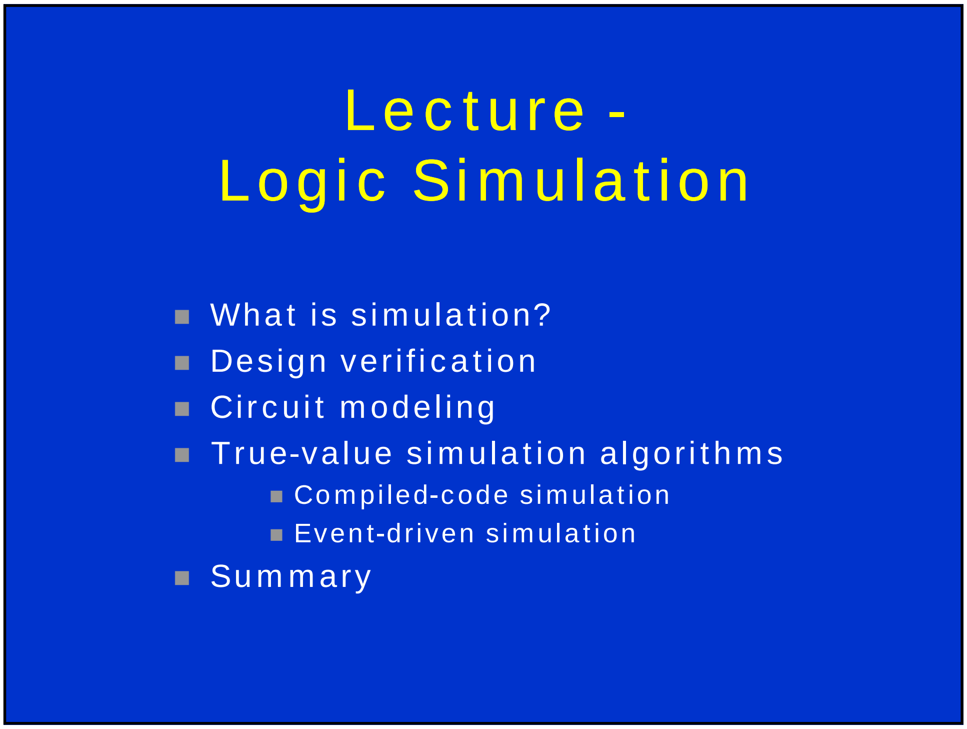
Guarding Between Signals



(a) Pin guard. (b) Guard wire. (c) Driven guard.

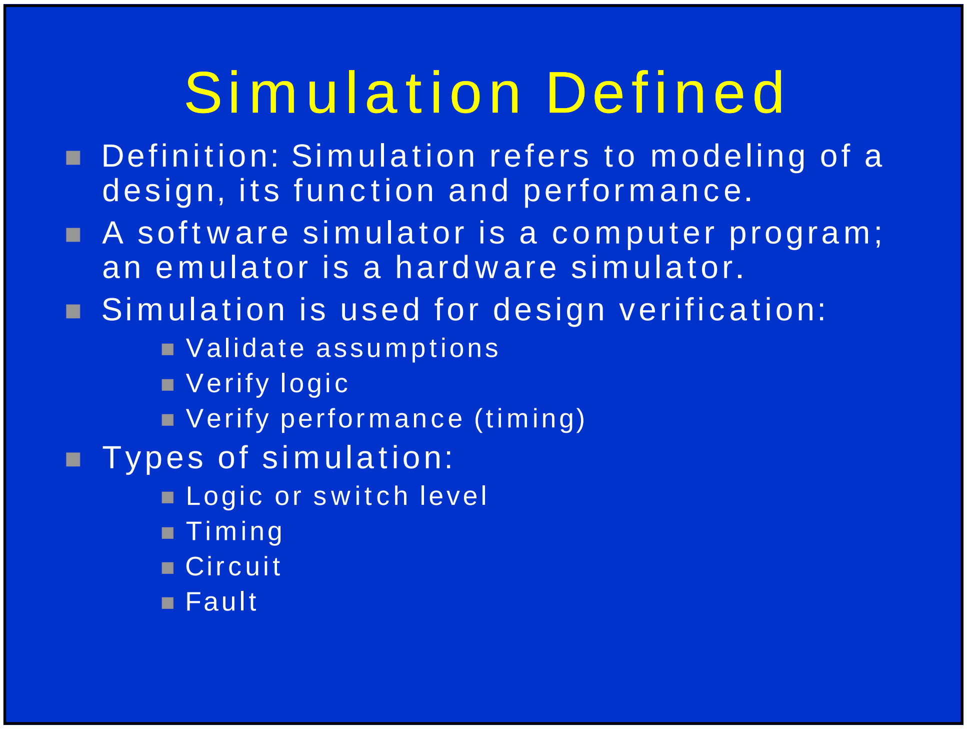
Summary

- 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



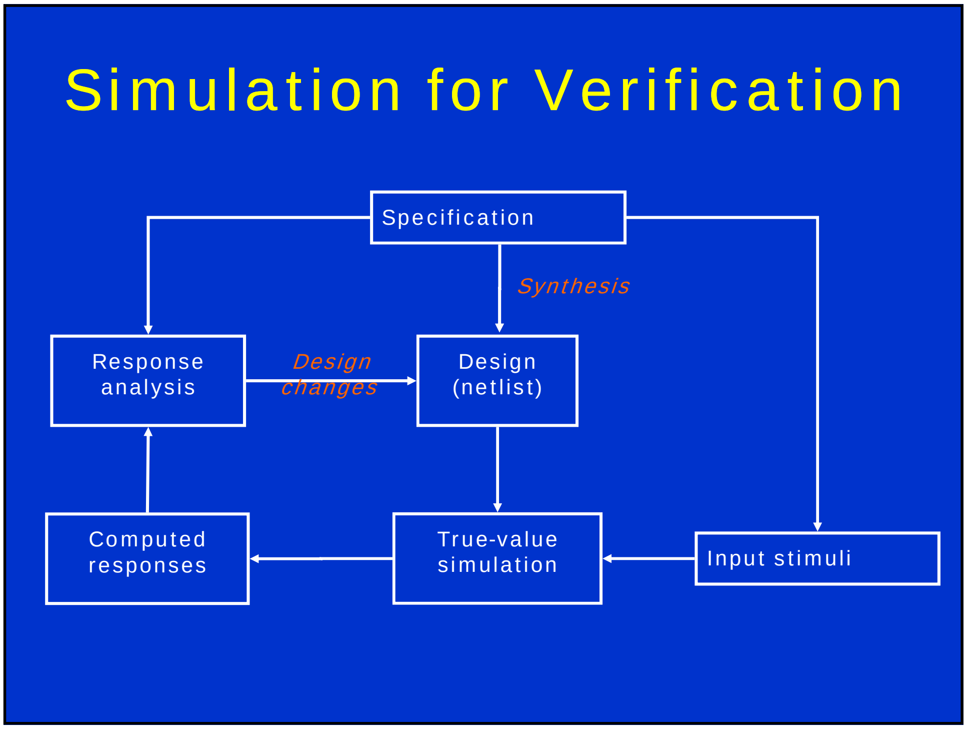
Lecture - Logic Simulation

- 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

Synthesis

Response analysis

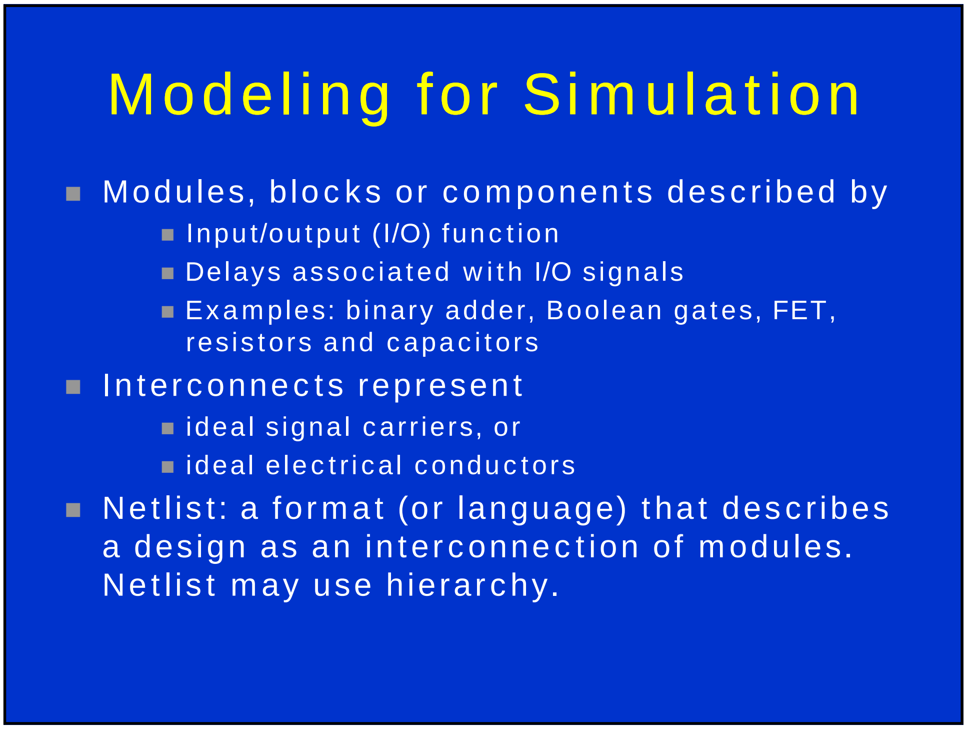
Design changes

Design (netlist)

Computed responses

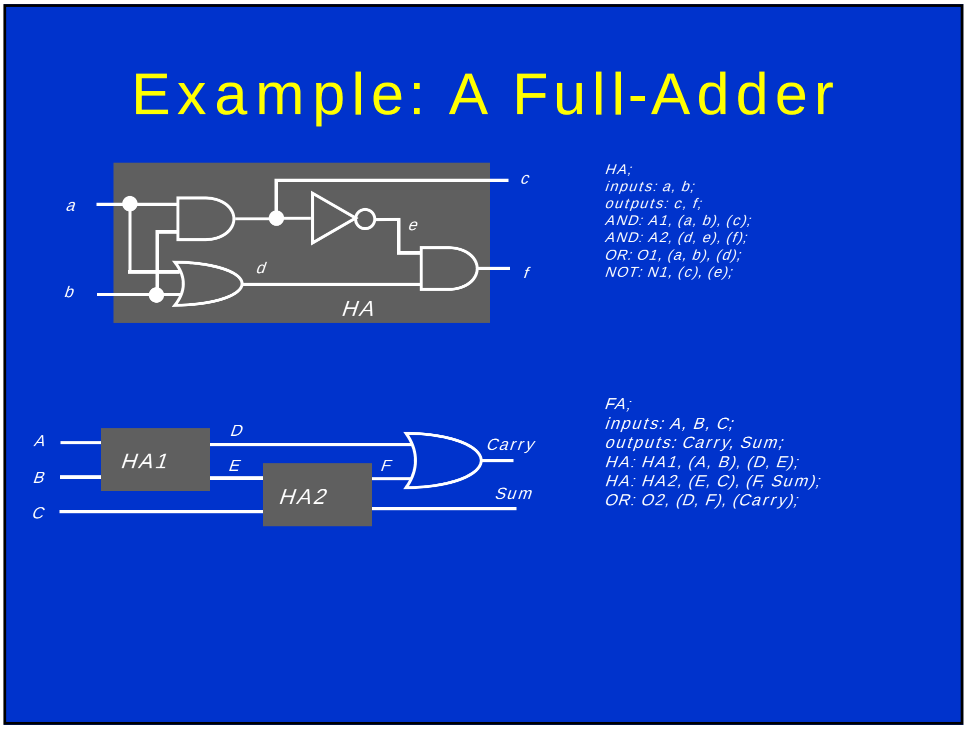
True-value simulation

Input stimuli



Modeling for Simulation

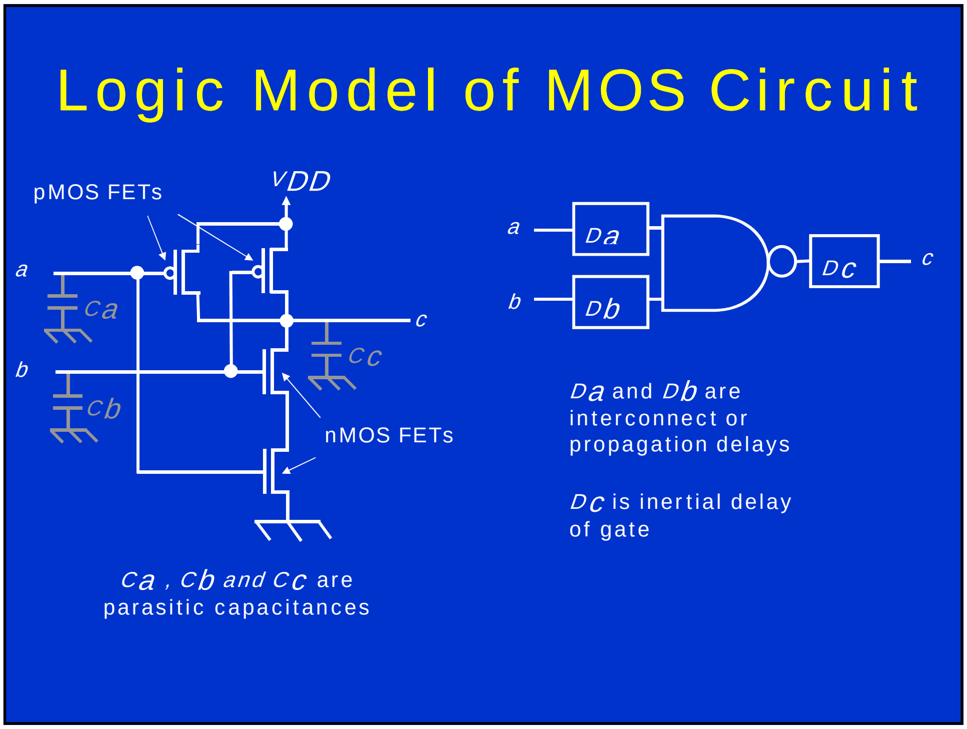
- Modules, blocks or components described by
  - Input/output (I/O) function
  - Delays associated with I/O signals
  - Examples: binary adder, Boolean gates, FET, resistors and capacitors
- Interconnects represent
  - ideal signal carriers, or
  - ideal electrical conductors
- Netlist: a format (or 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), (e);

FA;
inputs: A, B, C;
outputs: Carry, Sum;
HA: HA1, (A, B), (D, E);
HA: HA2, (E, C), (F, Sum);
OR: O2, (D, F), (Carry);

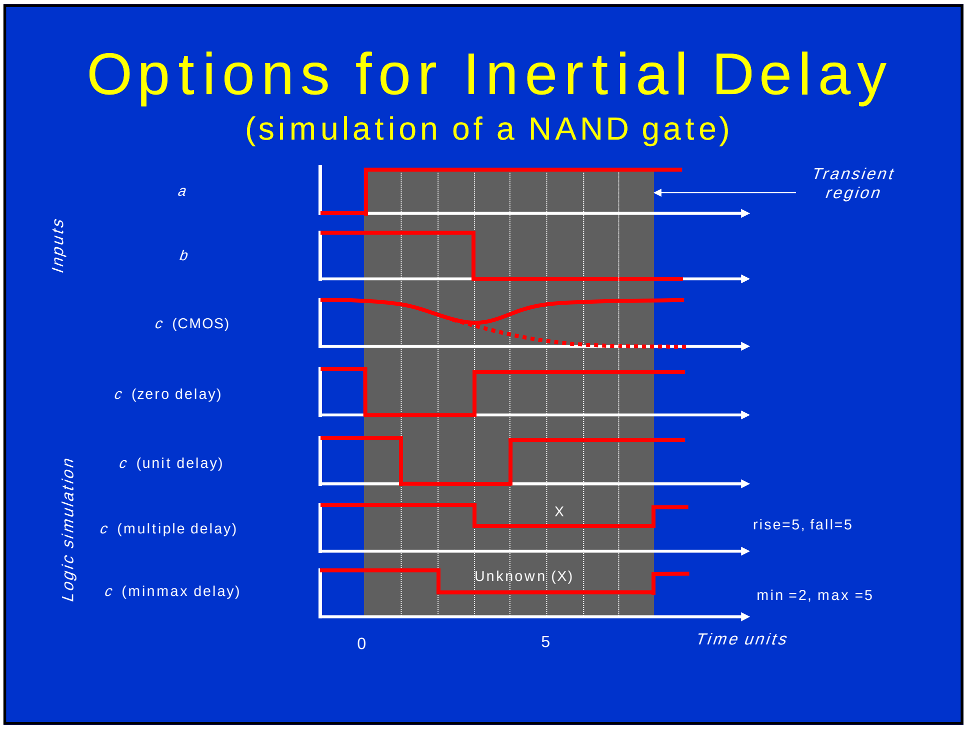


Logic Model of MOS Circuit

Ca , *Cb* and *Cc* are parasitic capacitances

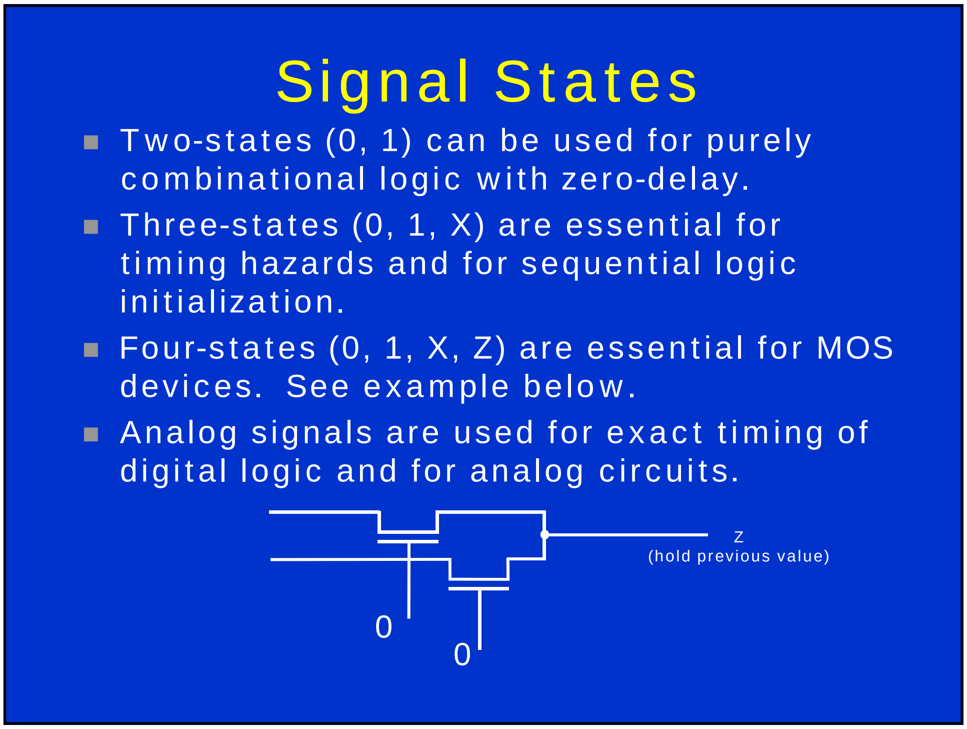
Da and *Db* are interconnect or propagation delays

Dc is inertial delay of gate



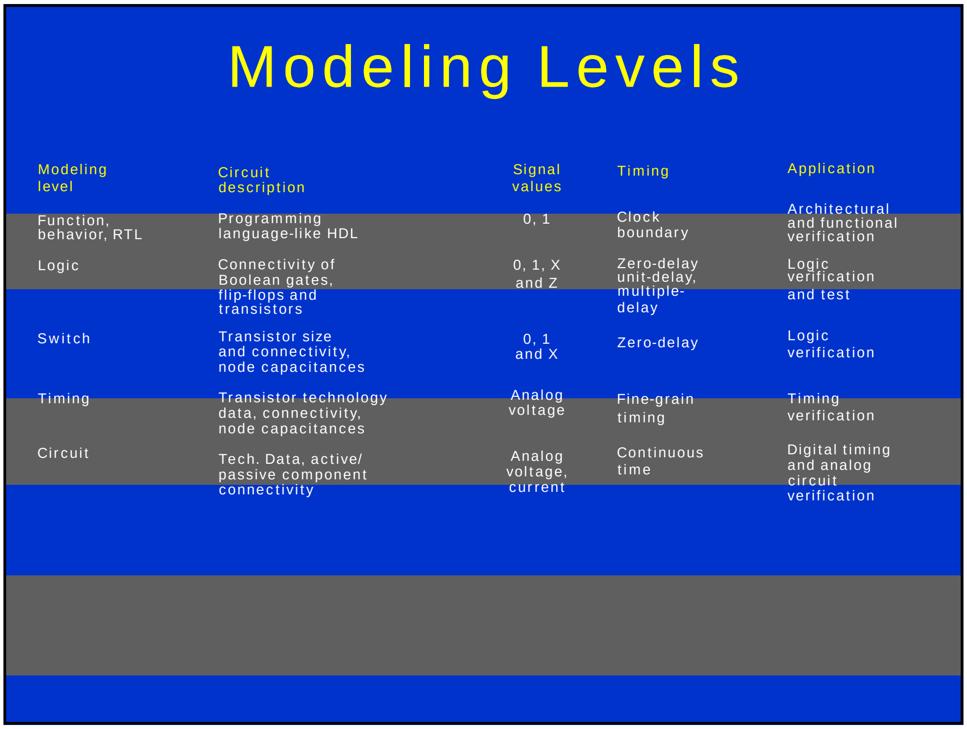
Options for Inertial Delay

(simulation of a NAND gate)



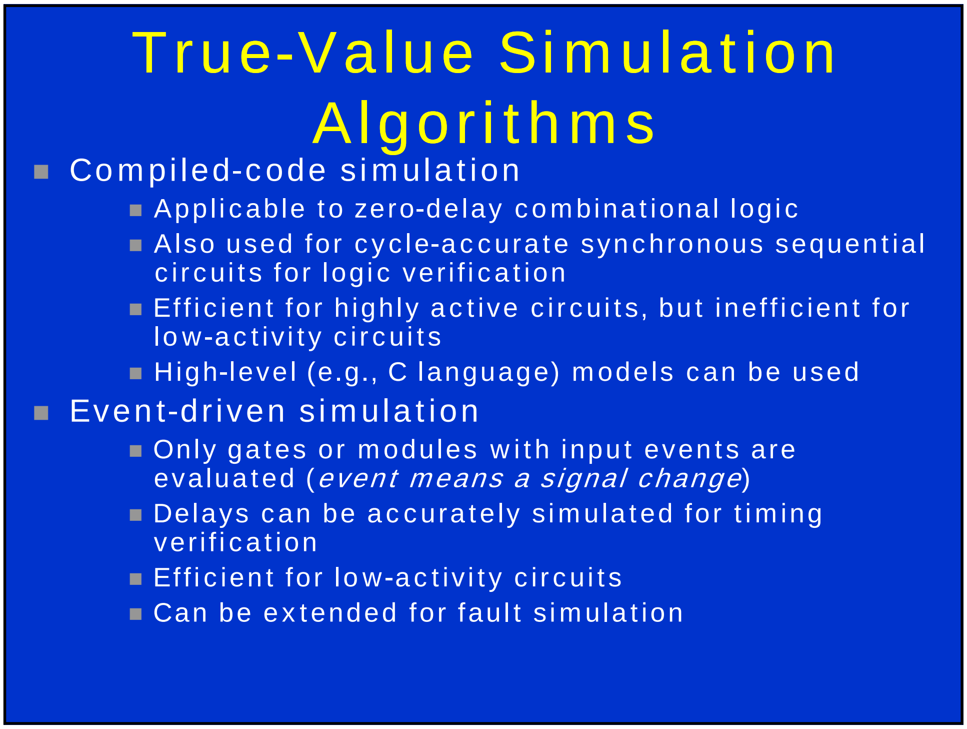
Signal States

- 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.



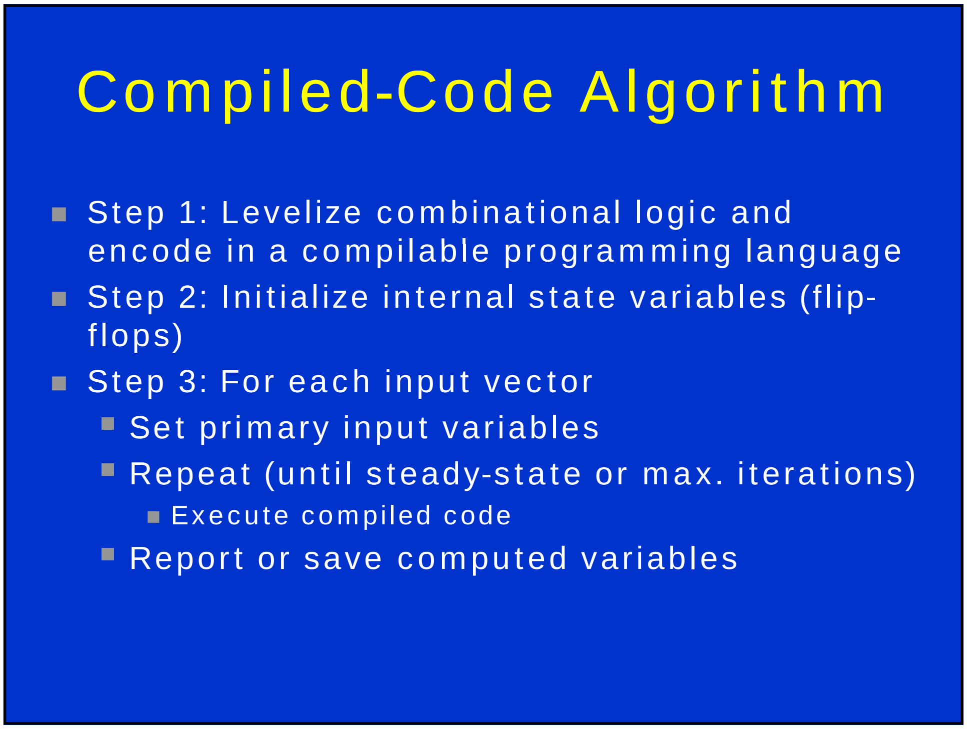
Modeling Levels

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



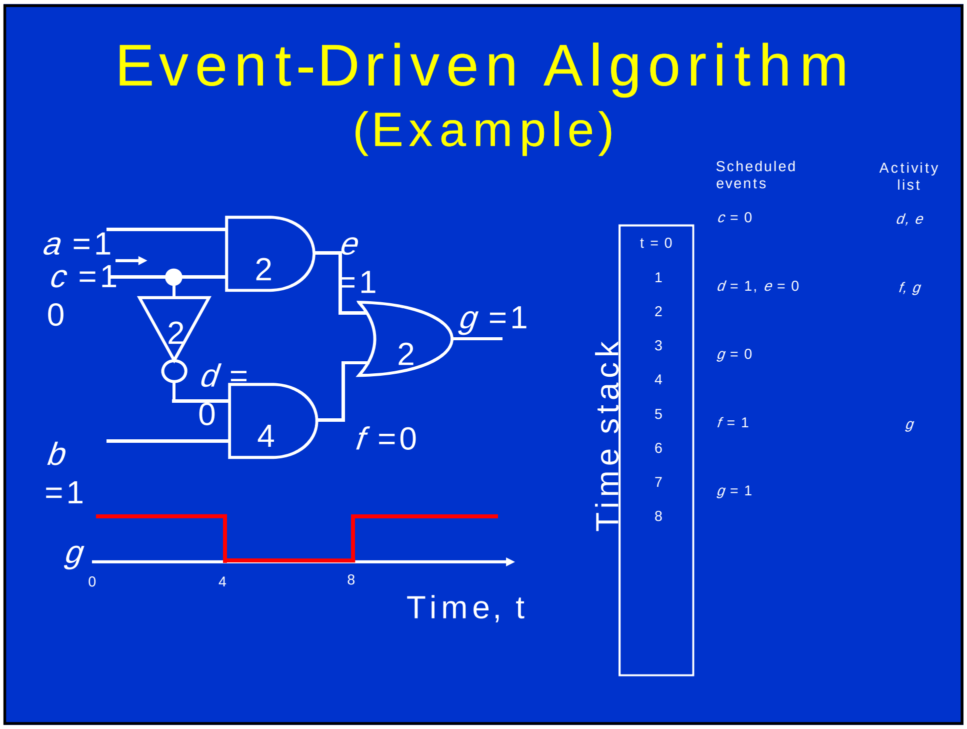
True-Value Simulation Algorithms

- 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



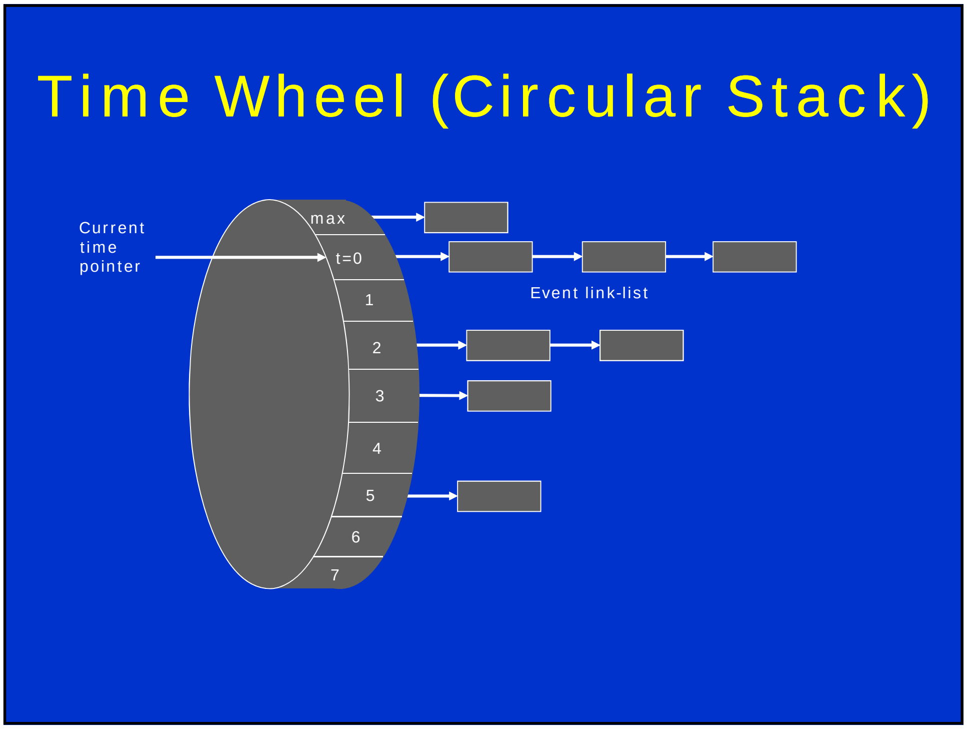
Compiled-Code Algorithm

- 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

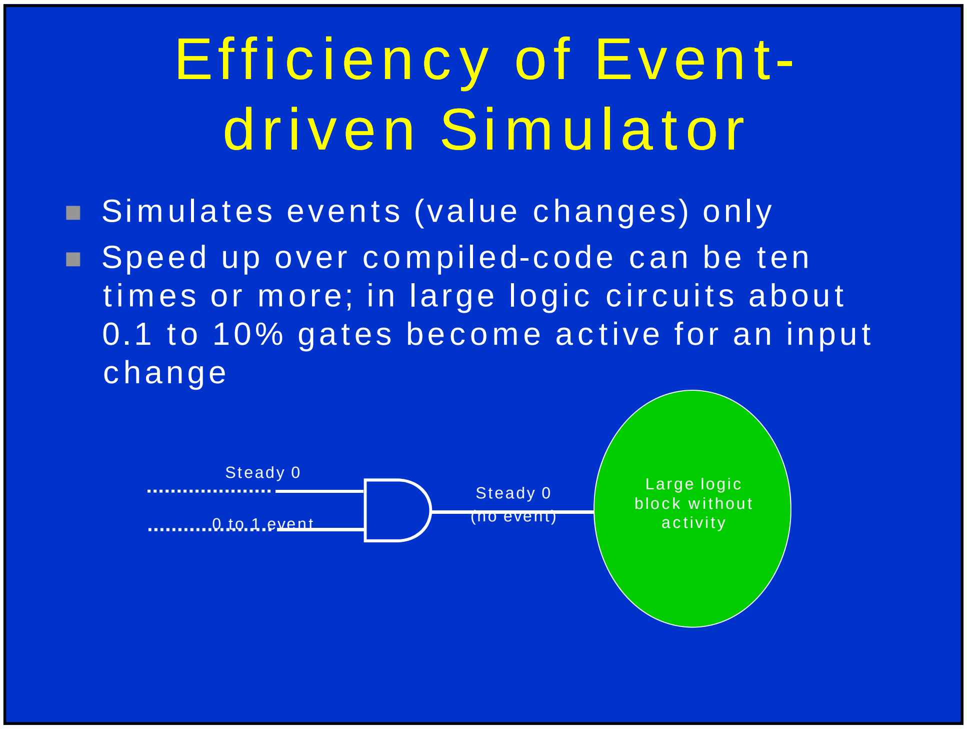


Event-Driven Algorithm (Example)

Time stack	Scheduled events	Activity list
t = 0	*c* = 0	*d, e*
1		
2	*d* = 1, *e* = 0	*f, g*
3		
4	*g* = 0	
5		
6	*f* = 1	*g*
7		
8	*g* = 1	

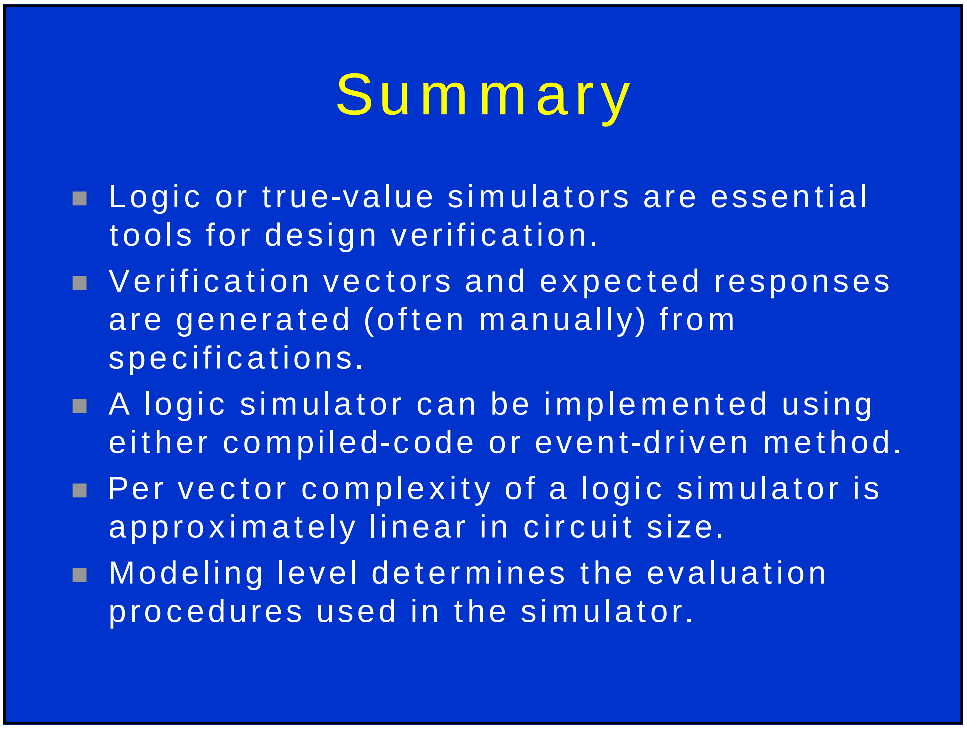


Time Wheel (Circular Stack)



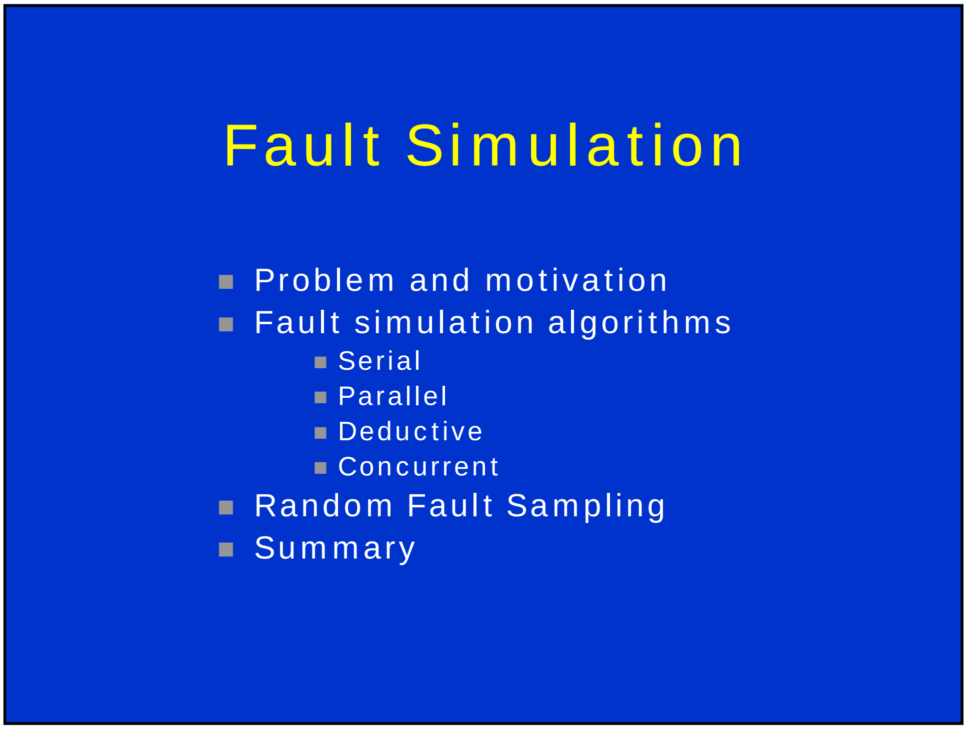
Efficiency of Event-driven Simulator

- 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



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.



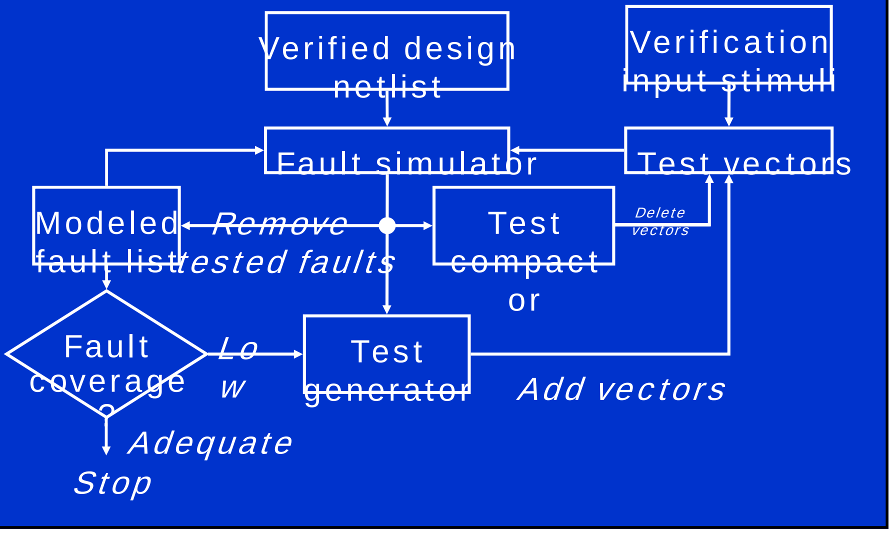
Fault Simulation

- 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

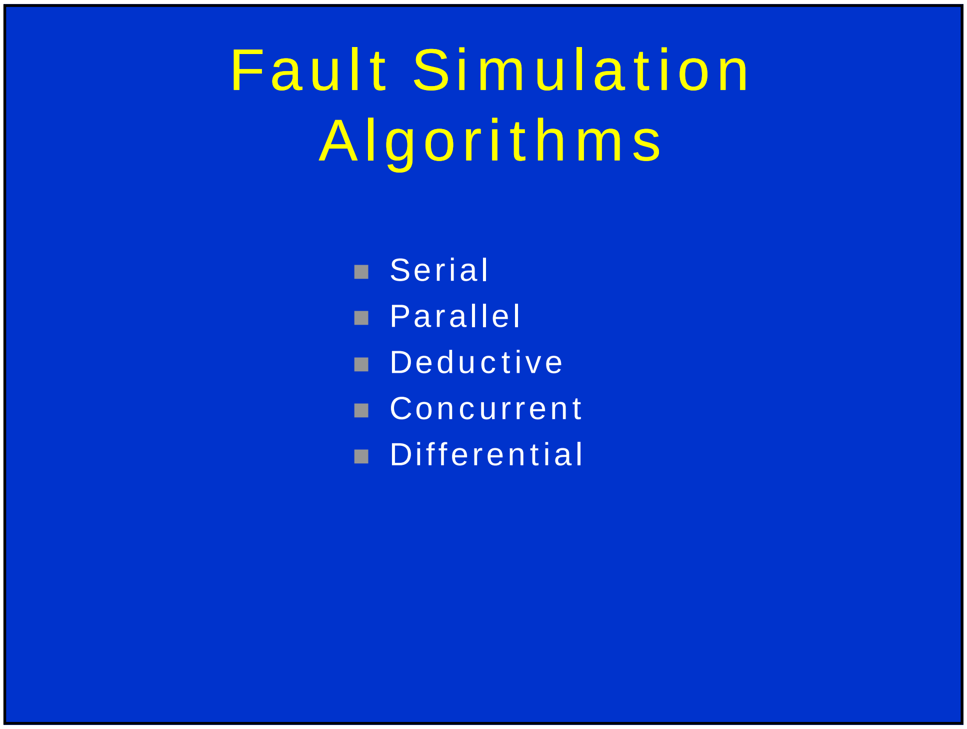


Fault Simulation Scenario

- 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

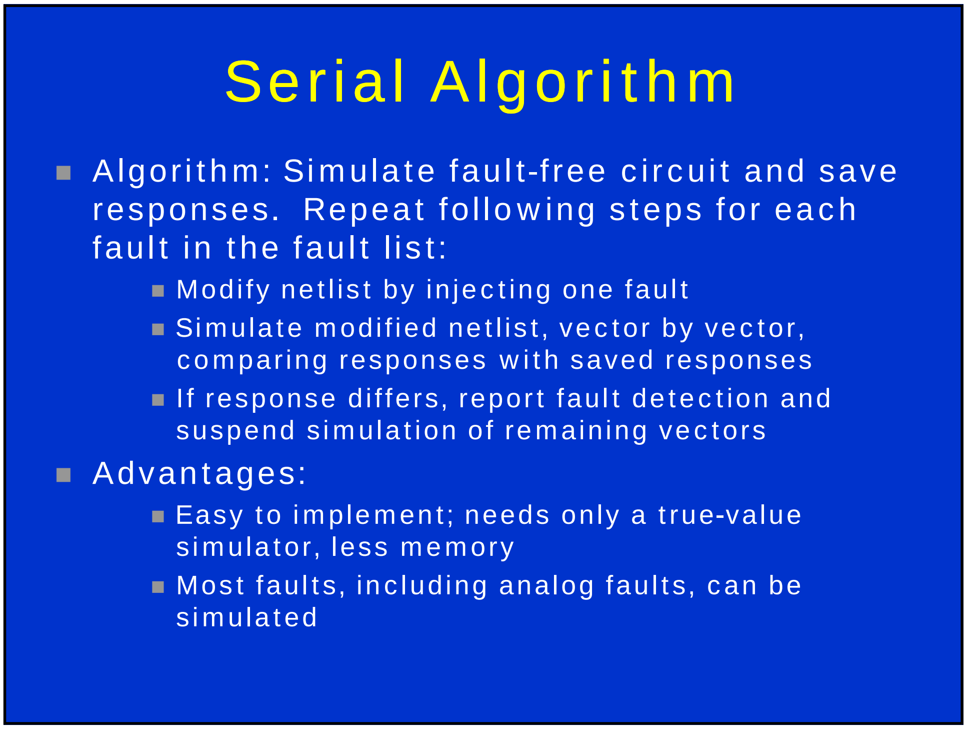
Fault Simulation Scenario (continued)

- 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

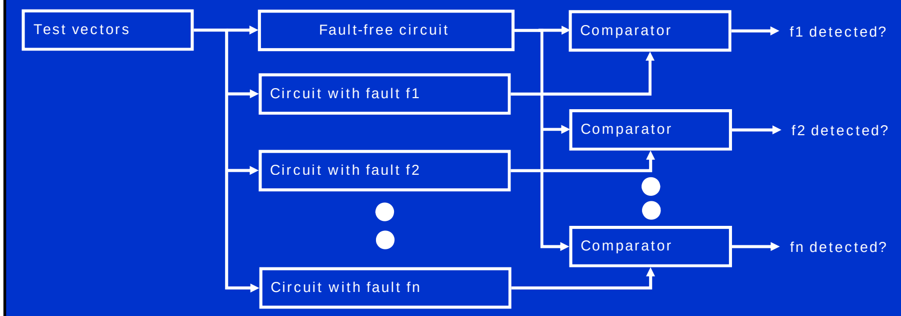


Serial Algorithm

- 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.)

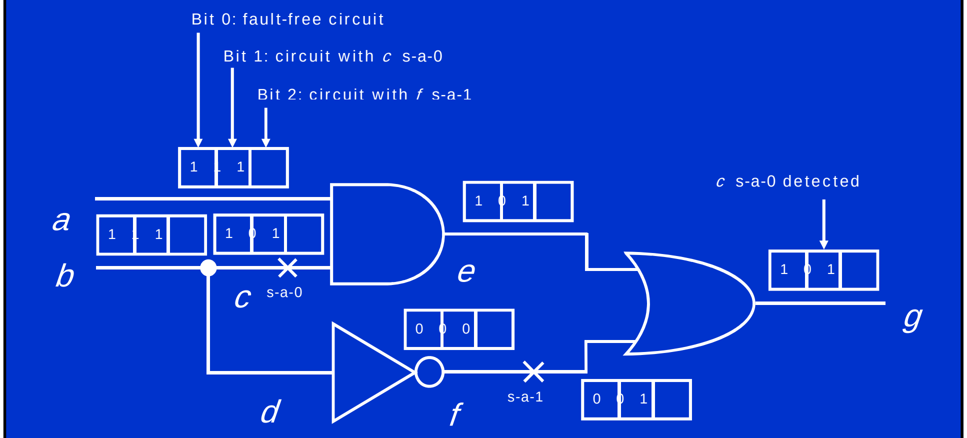
- Disadvantage: Much repeated computation; CPU time prohibitive for VLSI circuits
- Alternative: Simulate many faults together



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 ~ *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

c s-a-0 detected

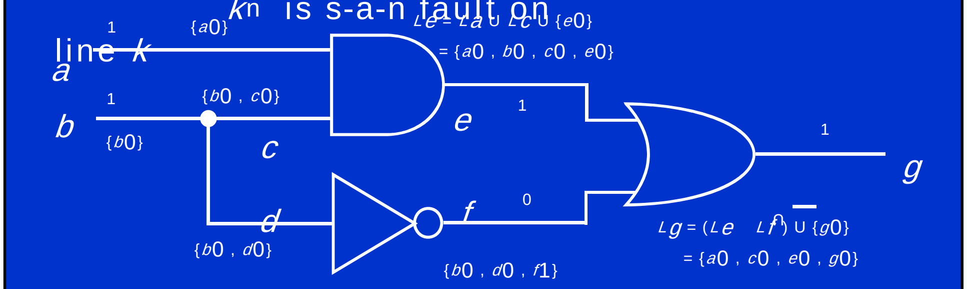
Deductive Fault Simulation

- 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

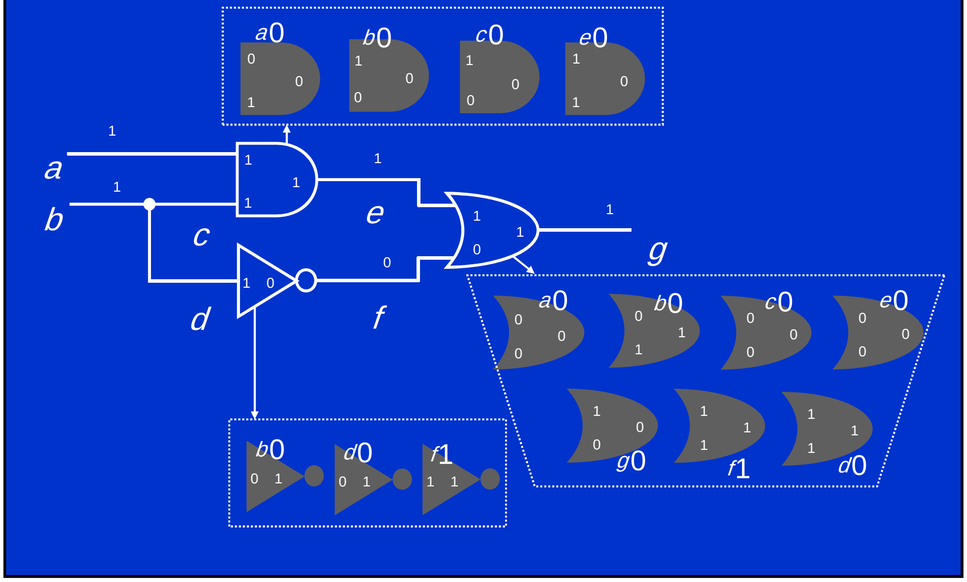

a
b
c
d
e
f
g
1
1
1
0
1
{a0}
{b0, c0}
{b0}
{b0, d0}
{b0, d0, f1}
$L_e = L_a \cup L_c \cup \{e_0\}$
$= \{a_0, b_0, c_0, e_0\}$
$L_g = (L_e \cap \overline{L_f}) \cup \{g_0\}$
$= \{a_0, c_0, e_0, g_0\}$

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.

Conc. Fault Sim. Example



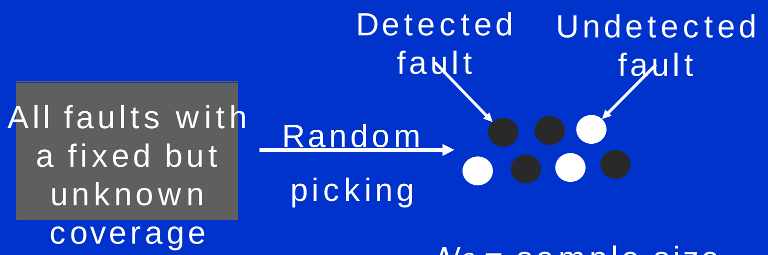
Fault Sampling

- 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



Detected fault

Undetected fault

All faults with a fixed but unknown coverage

Random picking

N_p = total number of faults (population size)

C = fault coverage (unknown)

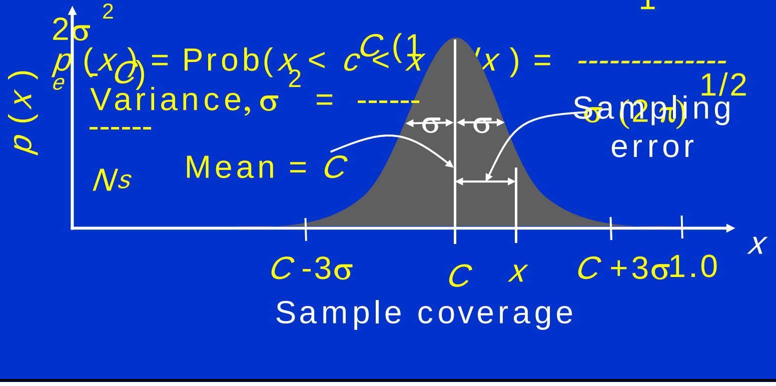
N_s = sample size

$N_s << N_p$

c = sample coverage (a random variable)

Probability Density of Sample Coverage, c

$$p(x) = \text{Prob}(x < c < x + dx) = \frac{1}{\sigma(2\pi)^{1/2}}\, e^{-\frac{(x - C)^2}{2\sigma^2}}$$



Variance, $\sigma^2 = \frac{C(1 - C)}{Ns}$

Mean = C

Sample coverage

Sampling Error Bounds

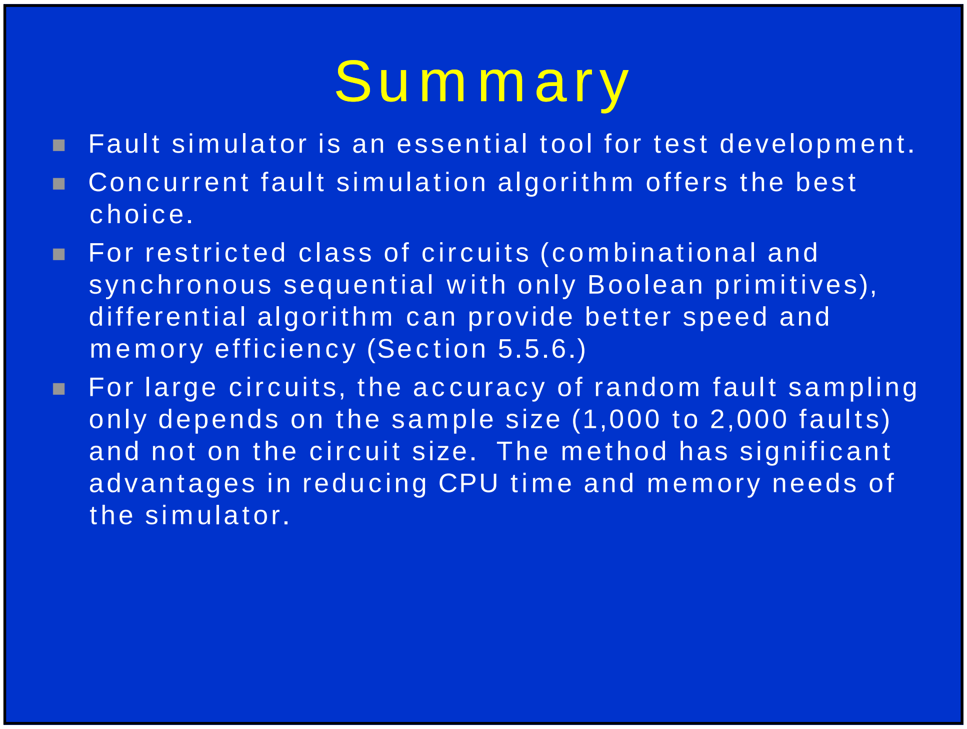
$$|x - C| = 3\left[\frac{C(1 - C)}{Ns}\right]^{1/2}$$

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

$$C_{3\sigma} = x \pm \frac{4.5}{Ns}\left[1 + 0.44\, Ns\, x(1 - x)\right]^{1/2}$$

Where Ns 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.