Verisity

Hardware and Software Verification for ARM SoC Design

EDP 2004

Jason Andrews
jason@verisity.com

Verification Defined.
Verification Delivered.

Definition of SoC

System-on-Chip (SoC)

A single chip that includes one or more microprocessors, application specific custom logic functions, and embedded system software

System-on-Board (SoB)

A design using one or more microprocessors along with multiple IC packages on a board

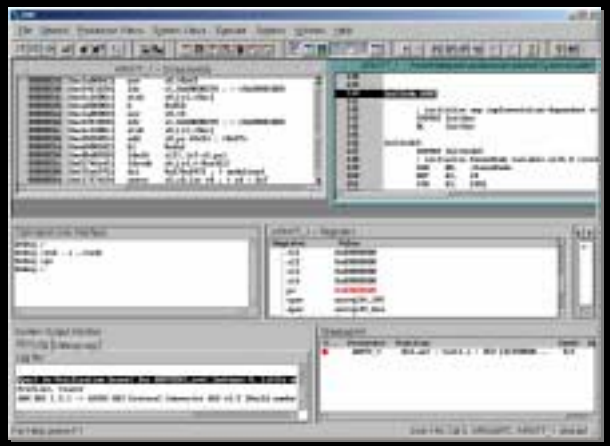
SoC Characteristics

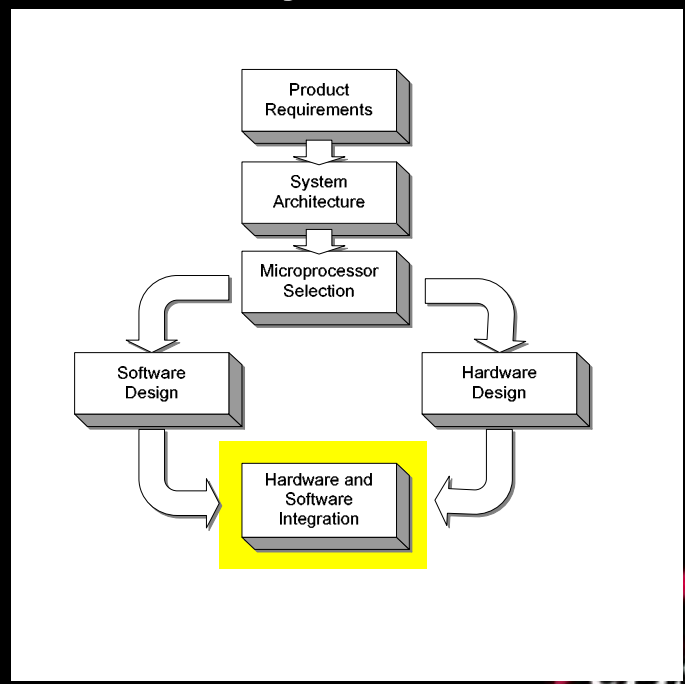
- One or more microprocessors on a chip
- Use of microprocessor IP such as ARM, MIPS, or Tensilica
- Often include DSP cores
- Integration, performance, and power are crucial
- Cost to develop is high
- High content of custom hardware
  - ✓ 1M gates is quickly becoming a small design
  - ✓ 2-4M gates is typical
  - ✓ 10M gates is large

Examples are Everywhere

Integration of Hardware and Software

- First chance for software to meet hardware
- Crucial part of project
- Sooner is better than later
- Too early makes for unnecessary work





Verification Platform

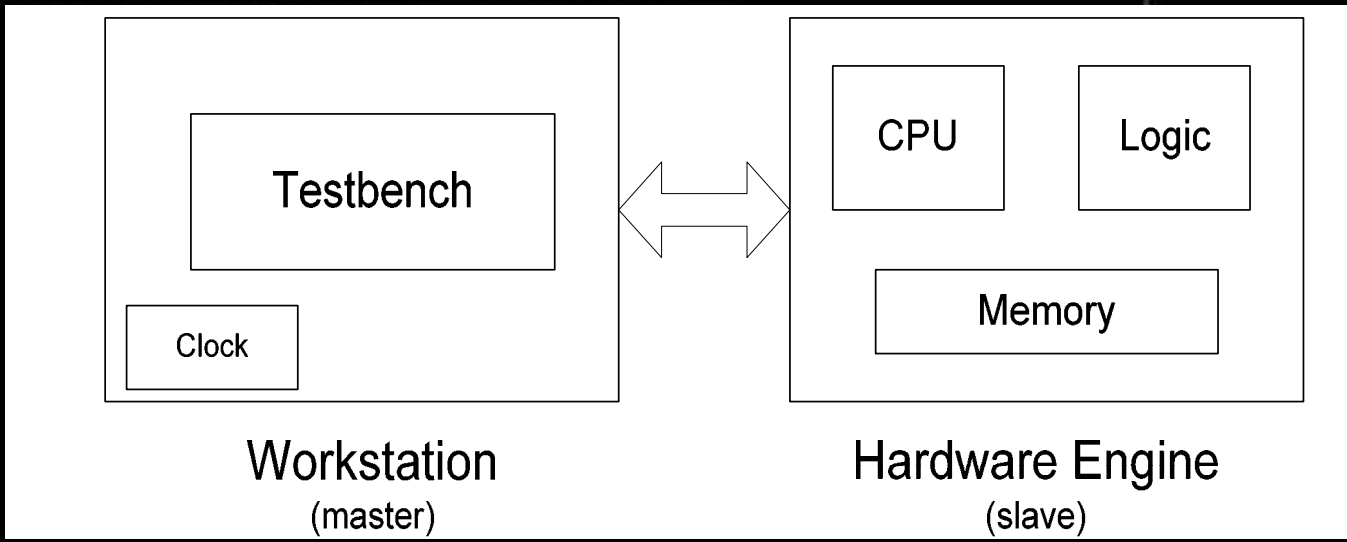
- Called hardware execution engine or virtual prototype
- 4 types
  - ✓ Logic simulation
  - ✓ Simulation acceleration
  - ✓ In-circuit emulation
  - ✓ Hardware prototype

Characterized by performance and ability to debug

Logic Simulation

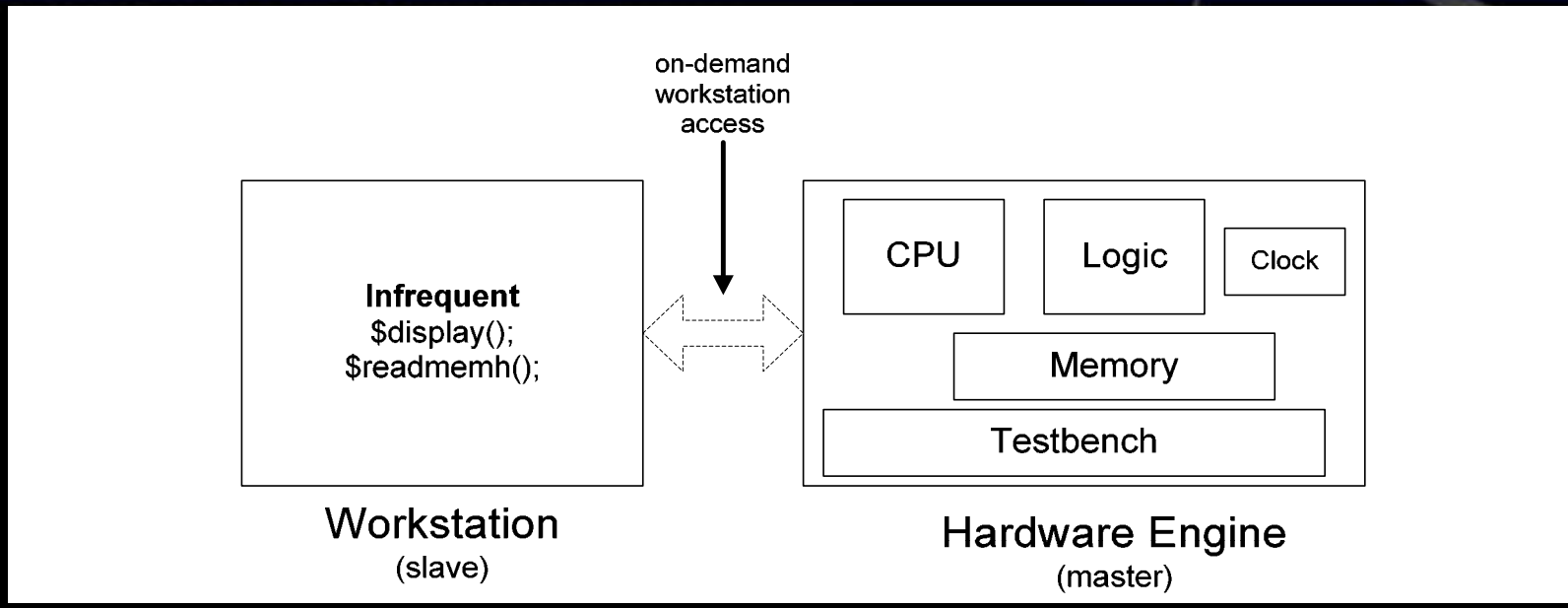
- Most commonly used verification platform
- Verilog and VHDL used to describe design
- Verification environment can be described in many different languages:
  - ✓ HDL
  - ✓ Special purpose verification language such as *e*
  - ✓ General purpose language such as C/C++ or SystemC
- Limited performance
  - ✓ 1000 cycles/sec for 1M gate design
  - ✓ 100 cycles/sec for 5M gate design
  - ✓ 1 cycle/sec for 50M gate design
- Referred to as “software simulation”
- Event-driven simulation algorithm

Simulation Acceleration



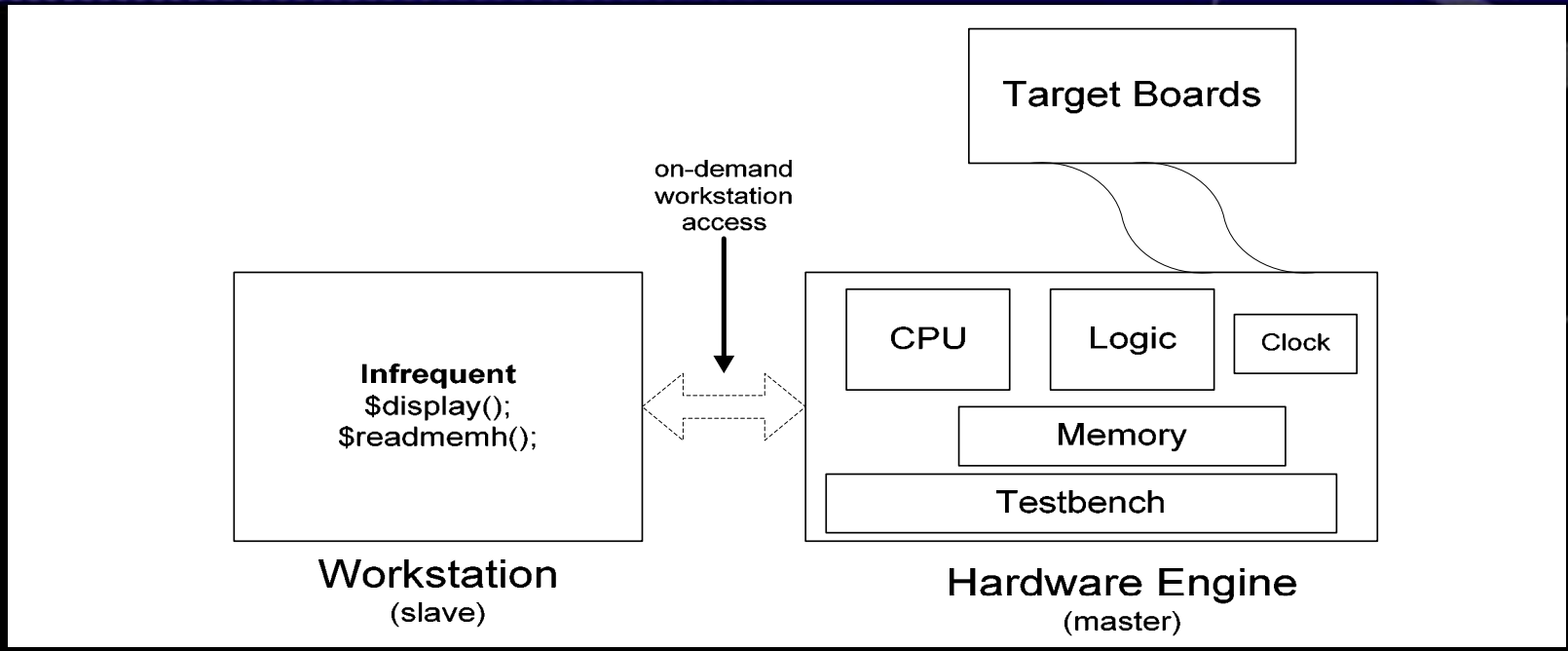
- Workstation is master and generates the clock
- Parallel processing of design improves performance
  - ✓ 20k cycles/sec to 100k cycles/sec
- Performance depends on:
  - ✓ Speed of the hardware engine
  - ✓ Amount of time spent on the workstation

Targetless Emulation



- ✓ Entire design and testbench mapped into hardware
- ✓ Hardware becomes the master
- ✓ Clocks generated in emulator
- ✓ No external hardware used for stimulus

In-Circuit Emulation



- ✓ Uses external hardware for stimulus
  - Live stimulus for applications such as PCI, Ethernet, UART, and JTAG
  - 250k cycles/sec to 750k cycles/sec

Hardware Prototype

- Alternative hardware representation of the design
- Available sooner than final design
- Willing to made trade-offs in size, performance, power, etc. in exchange for early availability
- Can be a design project of its own
  - ✓ Interconnect management
  - ✓ Technology issues such as clocking differences

Platform Attributes

Fast

Run in-circuit, real environment

Serve as an evaluation or demonstration system

Easy

Easy to compile designs

Easy to debug

Runs and feels like a logic simulator

Flexible

Connects to all types of models

Works with logic simulation

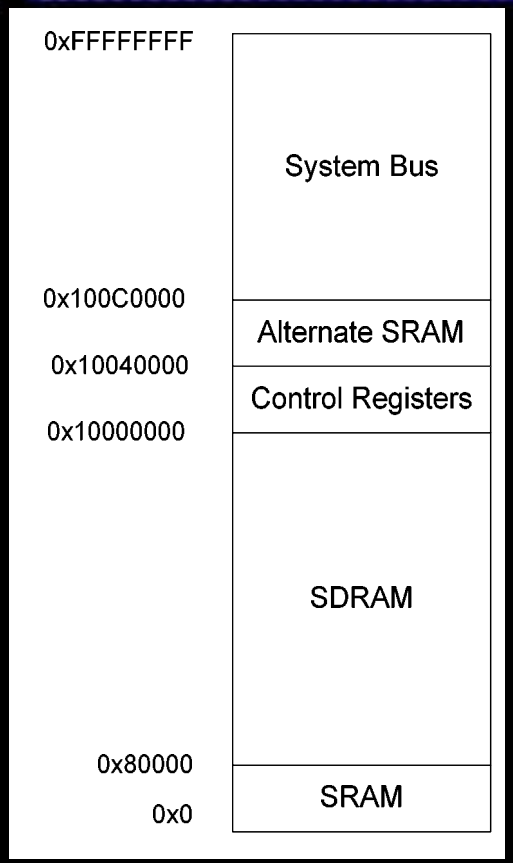
Cost

To model entire system

Replicate for multiple users

To support

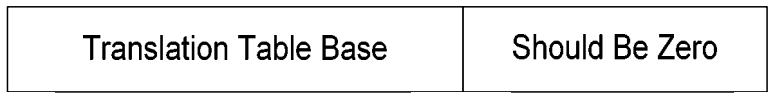
Software Engineer's World



R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13 (SP)
R14 (LR)
R15 (PC)
CPSR

Reset	0x0
Undefined Instruction	0x4
Software Interrupt	0x8
Instruction Fetch Abort	0xc
Data Abort	0x10
Interrupt (nIRQ)	0x18
Fast Interrupt (nFIQ)	0x1c

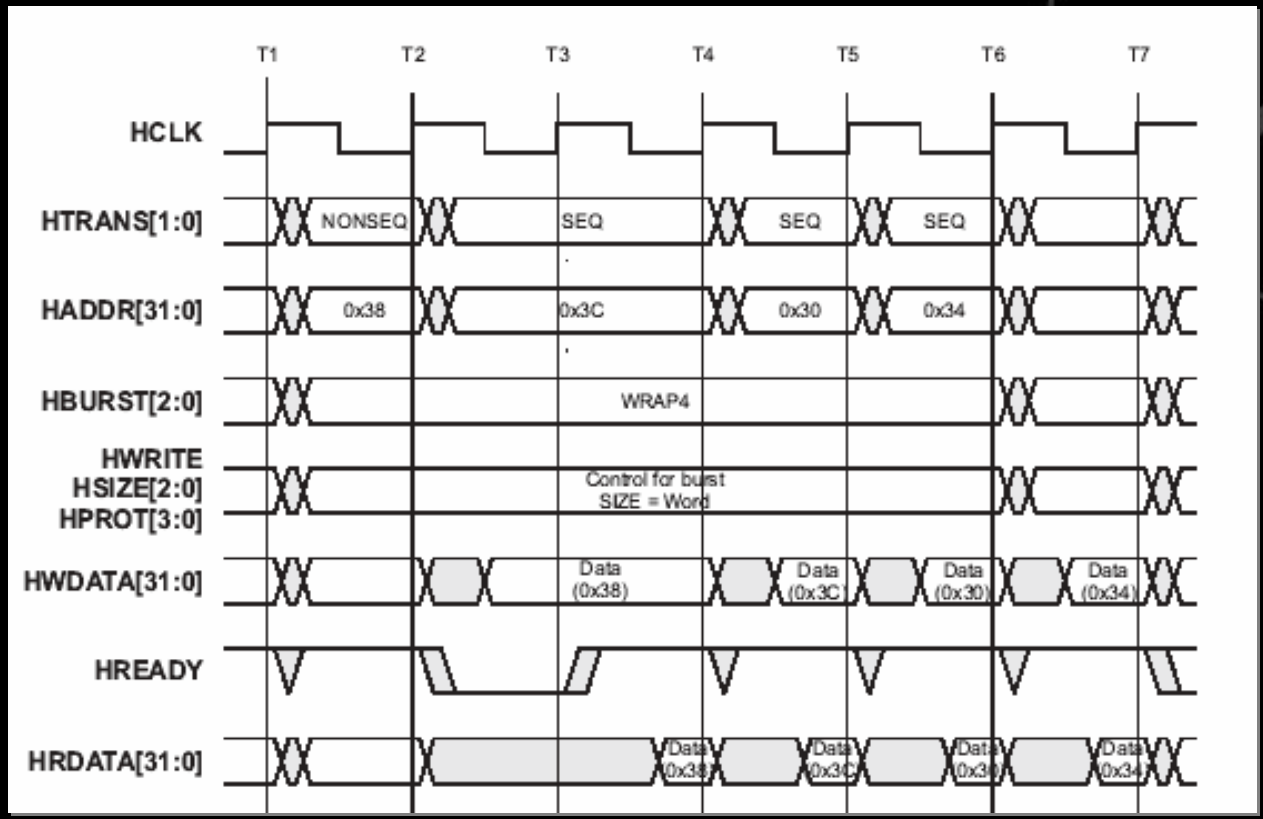
CP15 Translation Table Base Registers r2



MRC p15, 0, R1, c2, c0, 0 ; read TTBR
MCR p15, 0, R1, c2, c0, 0 ; write TTBR

Interrupts are vectors
Bus Interface means memory data

Hardware Engineer's World



Bus Interface and Interrupts are Pins

Software Debugging Methods

- Completely interactive
- Historically done in a lab with a board running 25 MHz or faster
- Use printf() statement to trace execution and variables
- Trace software execution with source-level debugger
- Use breakpoints to stop execution and inspect memory (variables and data structures), call stack, and register contents
- Iteratively reboot/restart and adjust breakpoints until bugs are found

Hardware Debugging Methods

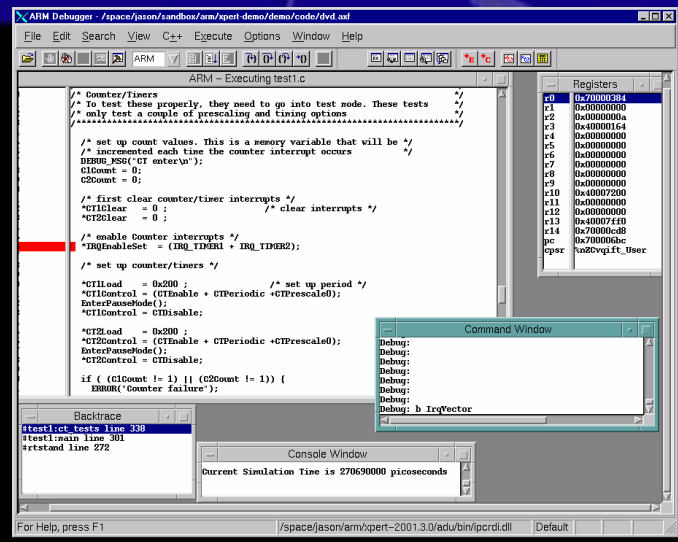
- Run logic simulation at 10-100 Hz
- Use print statements (or lack of print statements) to detect errors
- Use of waveform dumps to examine signal values until bugs are found

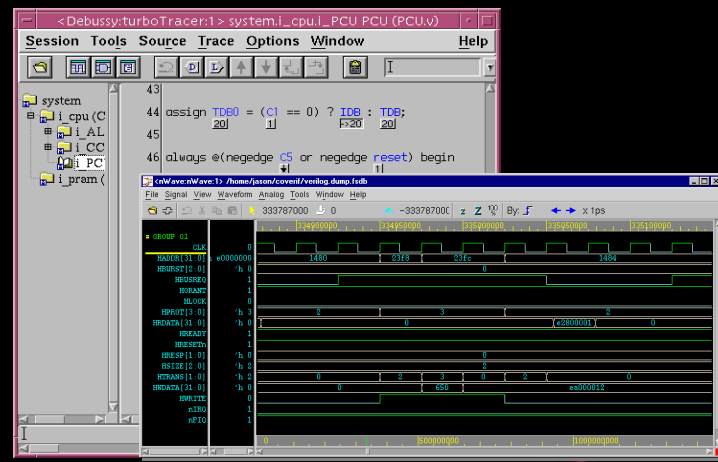
Hardware/Software Co-Verification

- Executing embedded system software on a simulated representation of embedded system hardware
- Provides verification and debugging capability for both hardware and software

Benefits

- Early testing of hardware-specific software
  - No waiting for prototypes in the lab
- Additional test stimulus for hardware
  - Better than contrived testbench
- Find real-world issues earlier in design process
  - Increased design confidence





The State of Verification

- Verification Efficiency
  - ✓ Verification Environment Creation
    - Environment, models, test generation
    - Methodology and Process Automation
  - ✓ Execution
    - Performance is king
  - ✓ Interpretation of Results and Debugging
    - Find and fix problems
    - Coverage

Language-Neutral Software Simulation

- Native code compiled, single kernel mixed-HDL software simulator
  - ✓ VHDL IEEE1076-1993 except for VITAL acceleration
  - ✓ Verilog IEEE1364-1995, 1364-2001, PLI 1.0 and PLI 2.0
- Arbitrarily mixing of VHDL and Verilog blocks
- Common user interface for interactive and post-processing debug
- Common API support with PLI for Verilog and VHDL

Acceleration and Emulation

- ASIC gate capacity up to 50MG
  - ✓ Expandable to 100MG
- Extended memory capacity up to 2.5GB
- On-board memory capacity up to 48MB
- Internal memory capacity up to 506Mbits
- I/O Capacity up to 1940
- Performance of up to 1MHz
- Multi-purpose hardware platform for simulation, acceleration and in-circuit verification

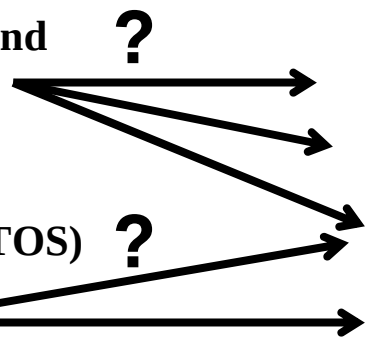
Types of Embedded System Software

- System initialization software and hardware abstraction layer (HAL)
- Hardware diagnostic test suite
- Real-time operating system (RTOS)
- RTOS device drivers
- Application software

SoC Methodology Confusion

Types of Embedded Software

1. System initialization software and hardware abstraction layer
2. Hardware diagnostic test suite
3. Real-time operating system (RTOS)
4. RTOS device drivers
5. Application software



Hardware Representation

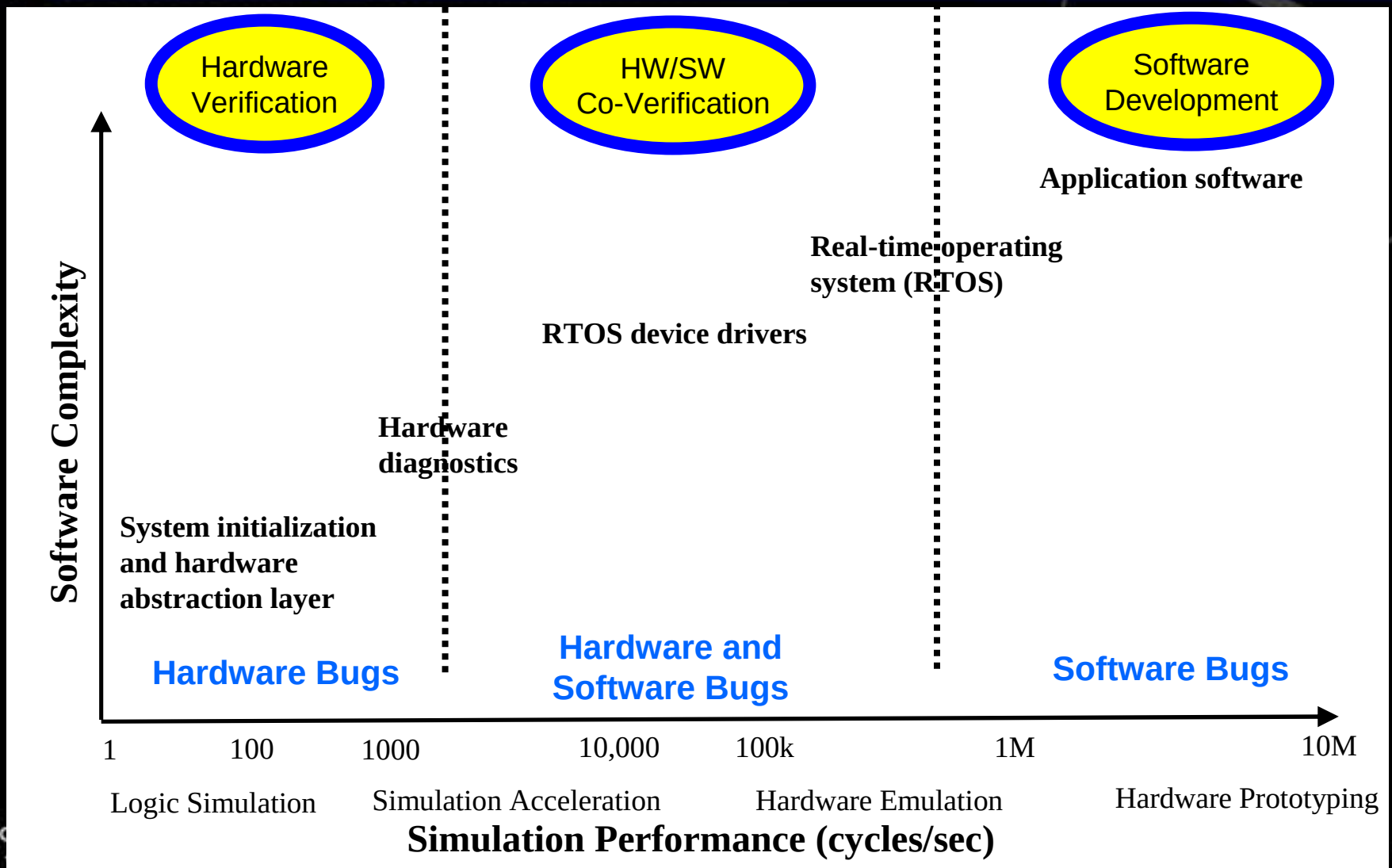
A. Logic Simulation
B. Simulation Acceleration
C. Hardware Emulation
D. Hardware Prototyping

Performance vs. Debugging

- Length of tests
- Probability of finding bugs in hardware
- Probability of finding bugs in software

Which software should be run on each representation of the hardware design?

Three Phases in SoC Verification

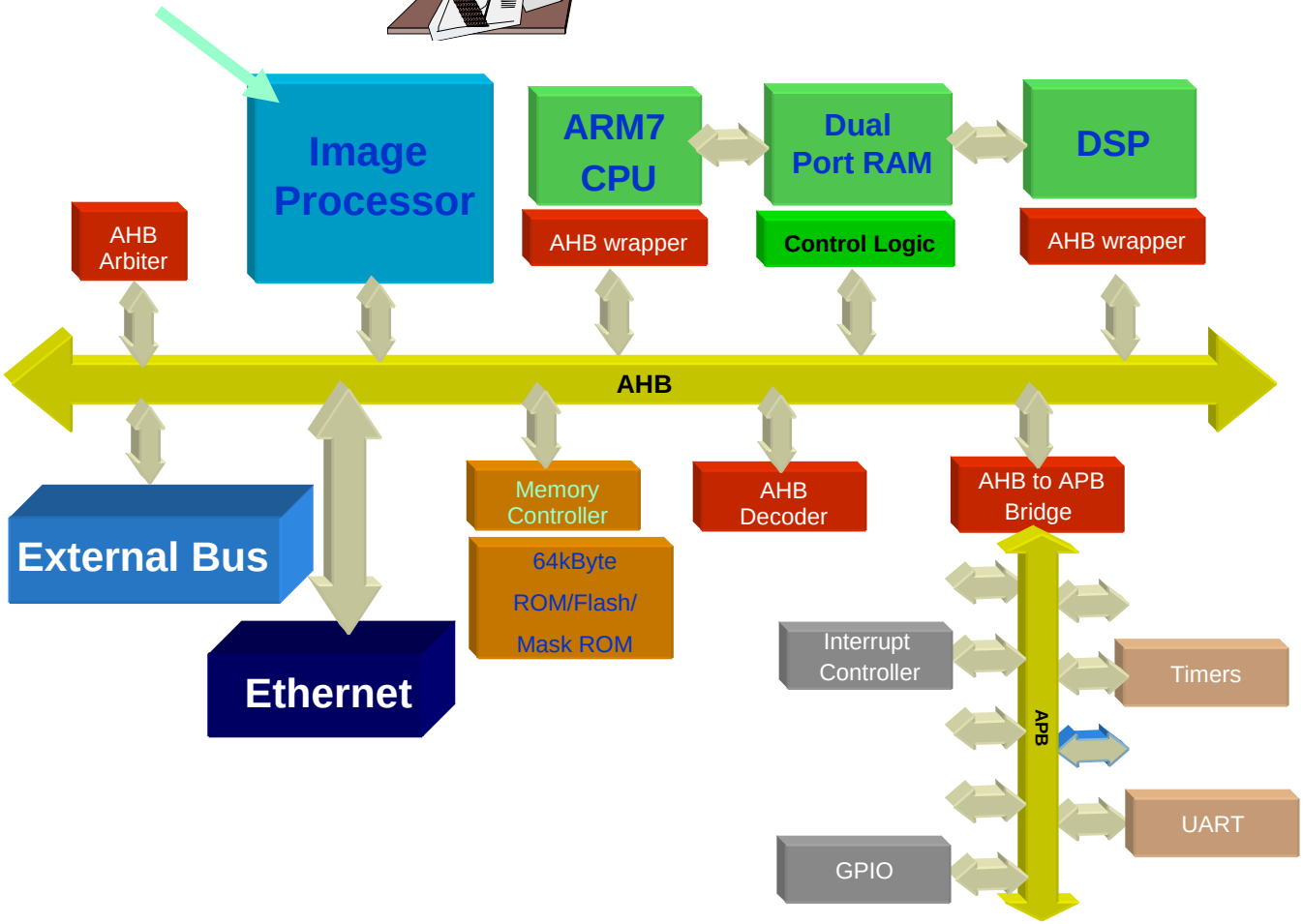


The Verification Matrix

	Transaction based model of CPU bus	Software model of ARM CPU	Hardware Model of ARM CPU
Software Logic Simulation	Block level testing 1	Initialization Software 2	In-Circuit Interface Testing 3
Simulation Acceleration	Directed Test Generation 4	Diagnostic Software 5	RTOS Porting 6
In-Circuit Emulation	Random Test Generation 7	Device Drivers 8	Application Software 9

Example ARM Design to be Verified

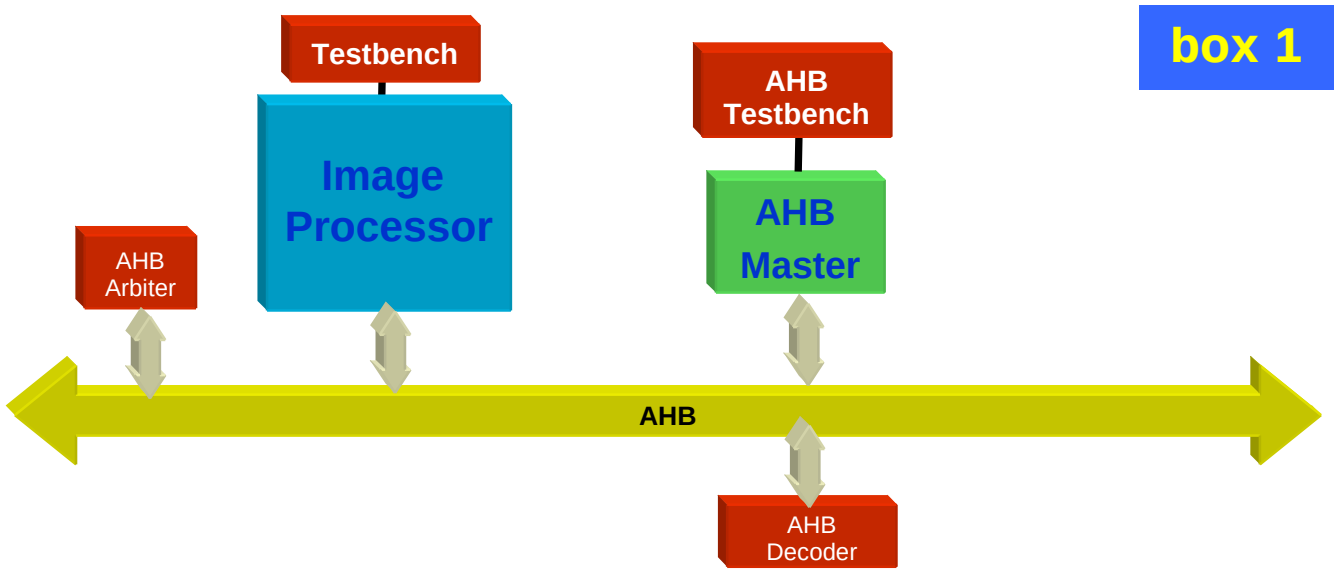
Joe is designing 1M gate image processor



Characteristics of Hardware Verification

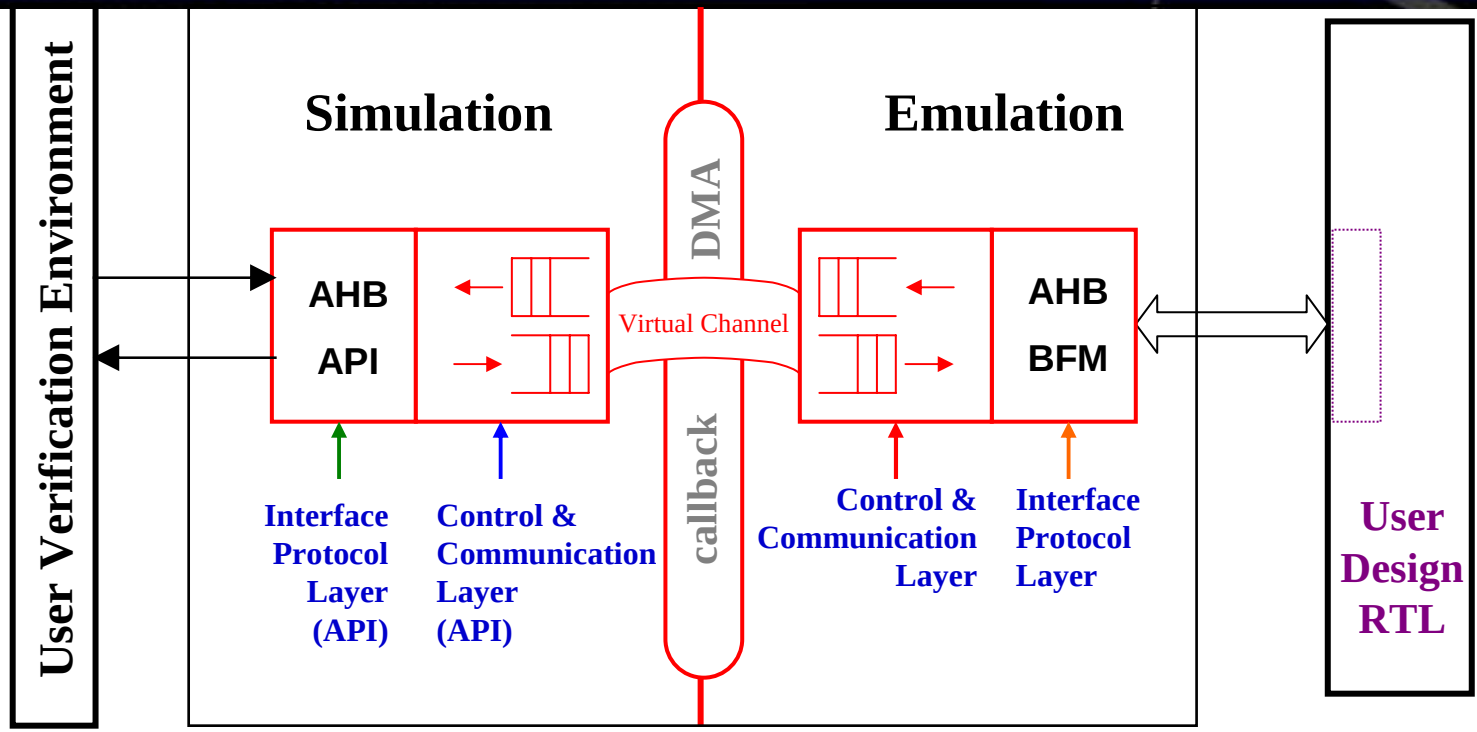
- Hardware design definitely has bugs
- Hardware engineers not interested in software unless running it is absolutely necessary
- Automation of testbenches, testcases, and environment is important
- Requires the best hardware debugging tools
- Performance is important but secondary
  - ✓ Engineers have learned patience

Hardware Verification Scenario



- Joe uses testbench to perform unit testing on his design
- ARM CPU replaced by AHB master
- Utilizes acceleration or simulation depending on design size and length of test

High-Performance AHB Verification Environment

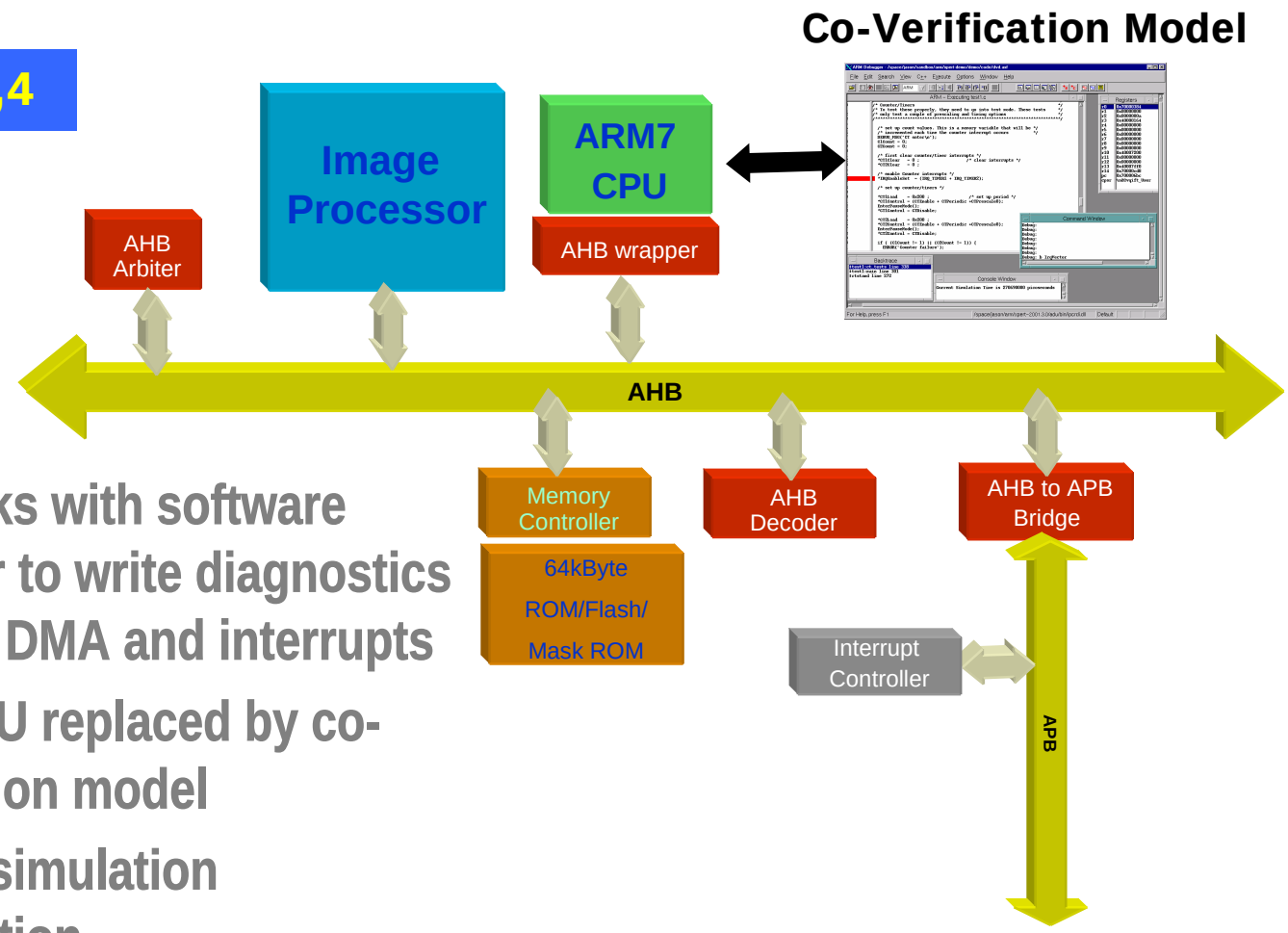


- Synthesizable AHB master and slave
- Transaction Level API callable from C or *e*
- Performance optimized for acceleration architecture
- Scales to boxes 4 and 7

Characteristics of HW/SW Co-Verification

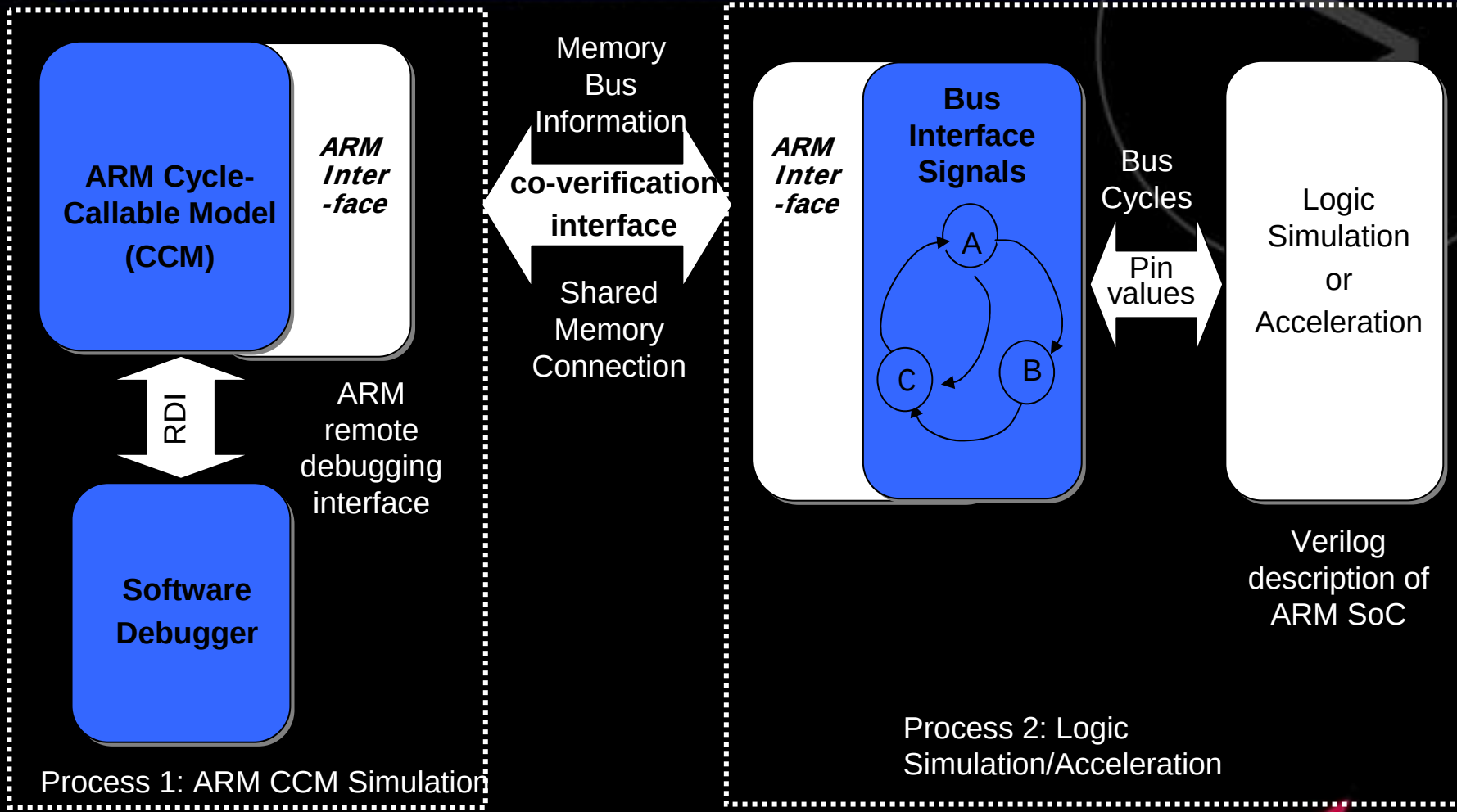
- Integration of hardware and software
- Both hardware and software likely to have bugs
- Hardware and software engineers must work together to resolve problems
- Hardware debugging requires good tools
- Software debugging requires good tools
- Need enough performance to run tests and provide responsive software debugging

Co-verification Scenario

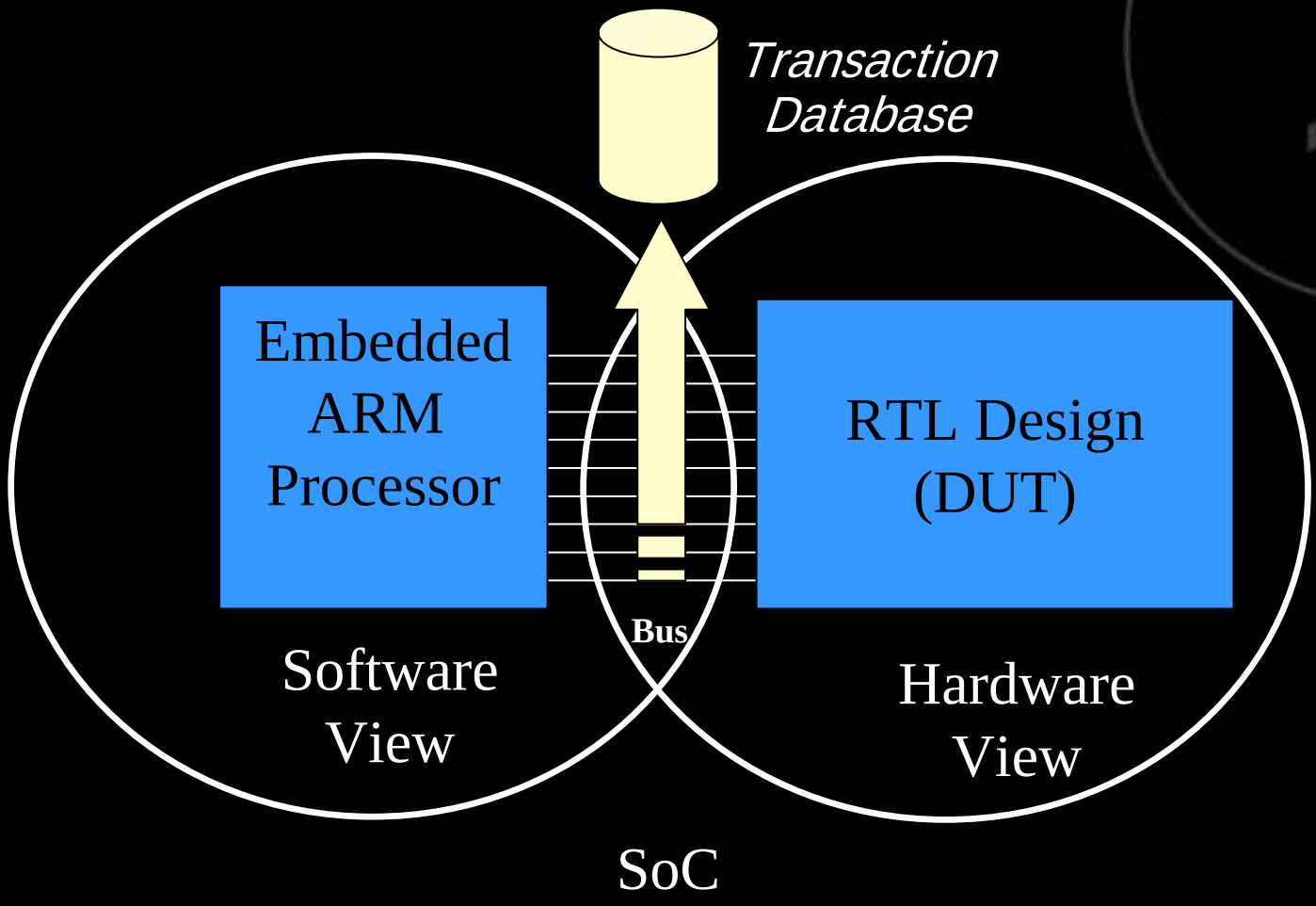


- Joe works with software engineer to write diagnostics to verify DMA and interrupts
- ARM CPU replaced by co-verification model
- Utilizes simulation acceleration

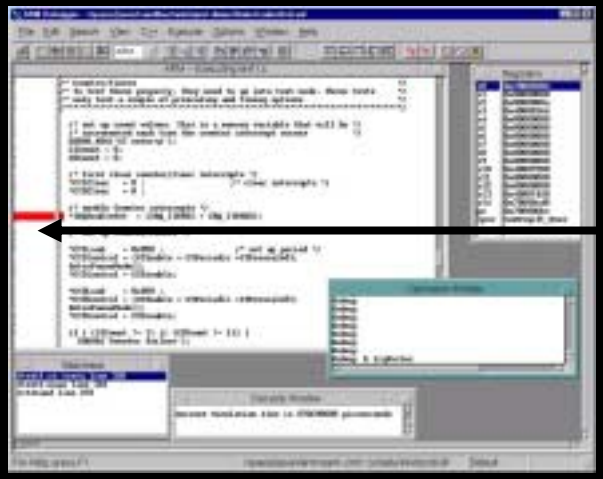
Co-Verification Architecture



Capturing Simulation History



Post-processing software and HW/SW Correlation

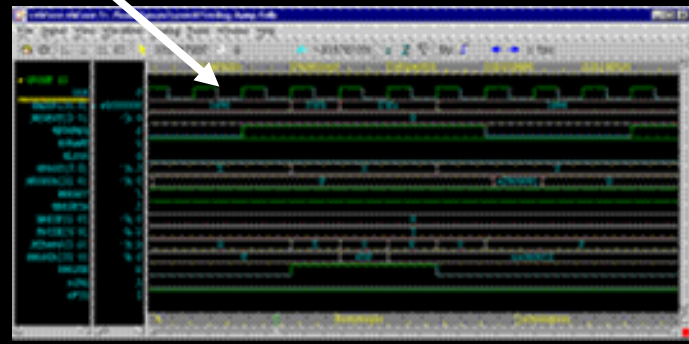


Software Source Debugger

Transaction Database

Source Line

Emulation Time



Hardware Waveform Display

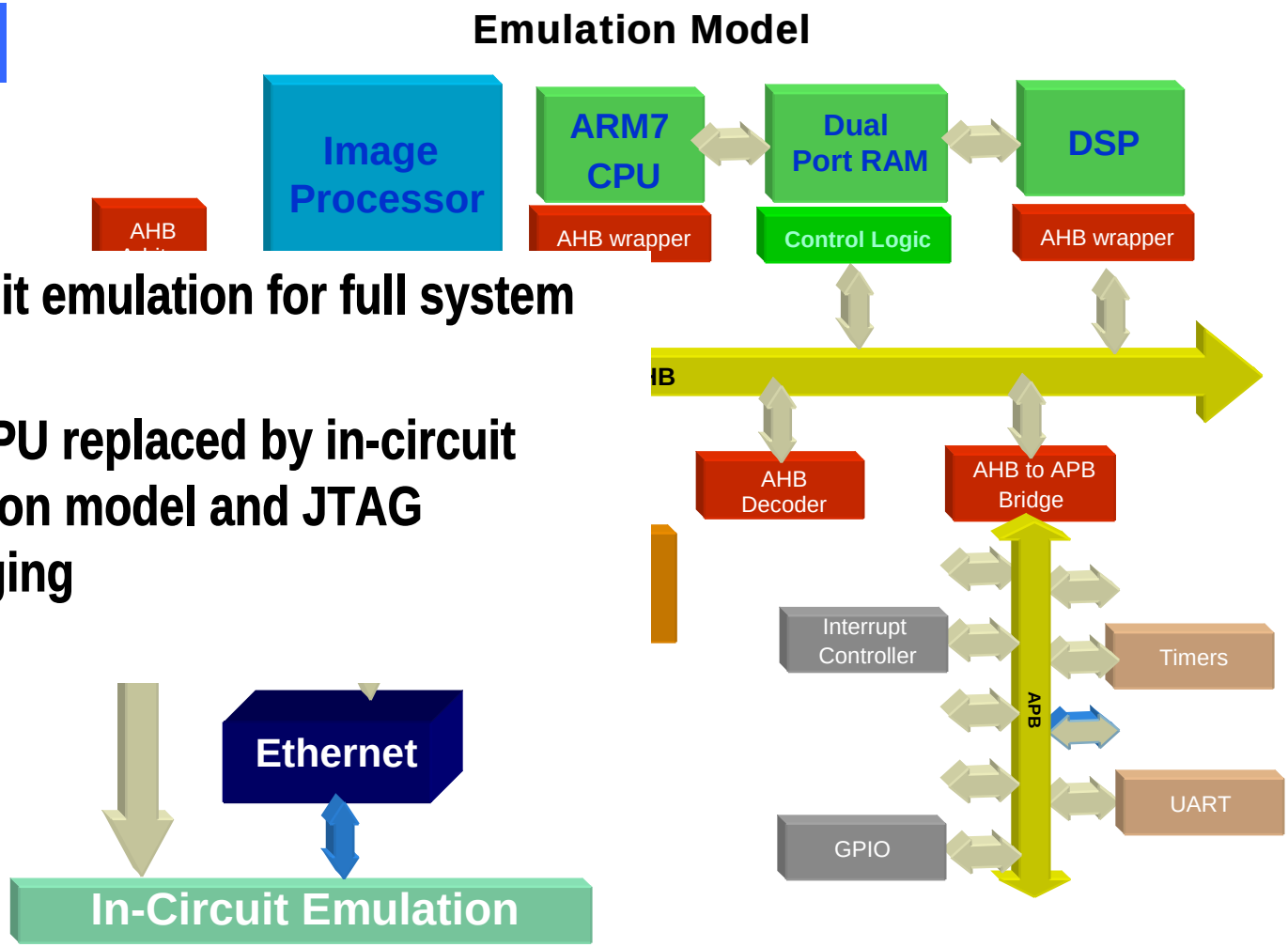
Characteristics of Software Development

- Software engineers assume hardware has no bugs
  - ✓ If it has bugs they go home
- Performance is most important
  - ✓ They measure slowdown from real speed
- Good software debugging is required
  - ✓ As close as possible to the final target system

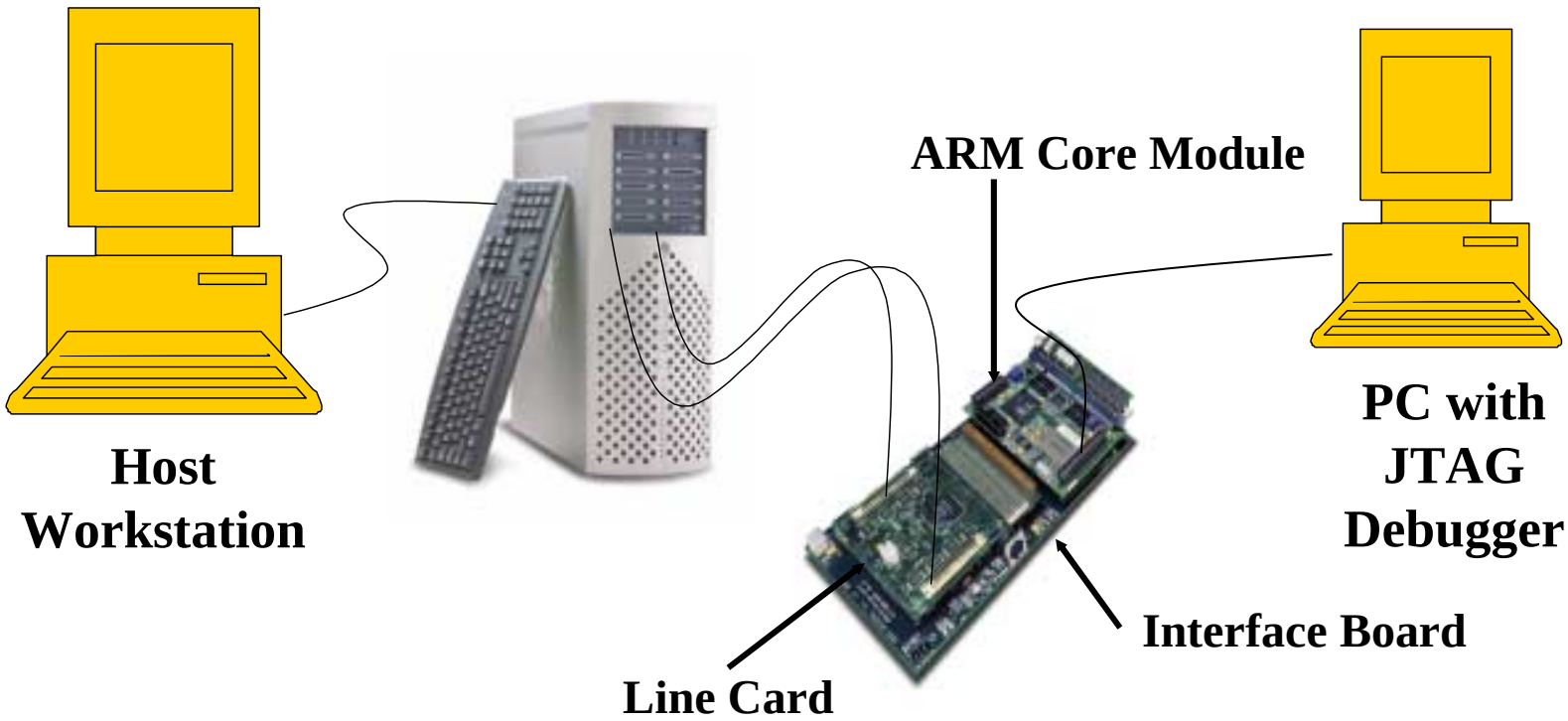
Software Development and System Verification Scenario

box 9

- In-circuit emulation for full system testing
- ARM CPU replaced by in-circuit emulation model and JTAG Debugging



In-Circuit Emulation with ARM JTAG Debugger



Three Distinct Applications

- Hardware verification
- HW/SW co-verification
- Software development and System Verification

Characterized by performance and debugging

Trends and Conclusion

- Search for common platform
  - ✓ Meet the needs of multiple applications
  - ✓ Without it too much time is wasted
- Must address the 3 applications separately
  - ✓ Solutions are not the same
  - ✓ Grid lock occurs if a common solution cannot be agreed upon
- Stronger link between testbench and embedded software
  - ✓ Constrained random techniques applied to software
  - ✓ Better control of hardware and software to stress design
  - ✓ Better debugging views of testbench, design, and software