



Simulating Embedded Systems For Software Development

Jakob Engblom, PhD
Business Development Manager
Virtutech
jakob@virtutech.com

- Simulation: a way to study the world
- Build a **model** of a system
- **Try** scenarios on this model
 - **Experimental**, not analytical approach
- Understand the real system from the model



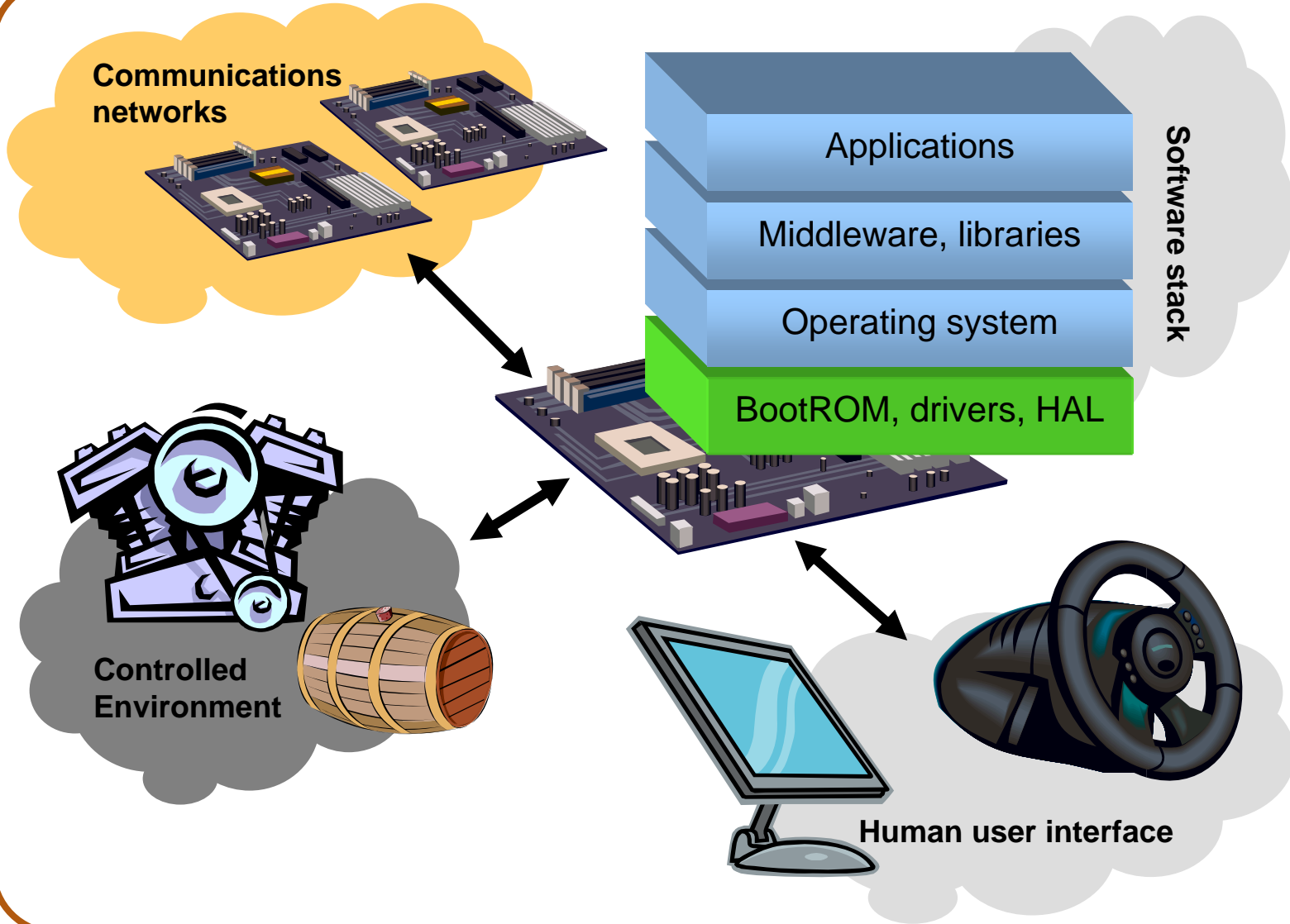
Simulation Advantages

- “Just software”
- Availability
 - Easy to copy & distribute
 - Requires no special lab
 - Good for global reach
- Flexibility
 - A computer can be “any” system, no fixed lab setup
- Inspectability
 - Any variable or property can be observed, even if hidden in the real world
- Controllability
 - Any variable or property can be changed
 - Controlled experiments, not real-world random
- Configurability
 - Easy to change configuration and create new configurations
 - Easy to vary parameters

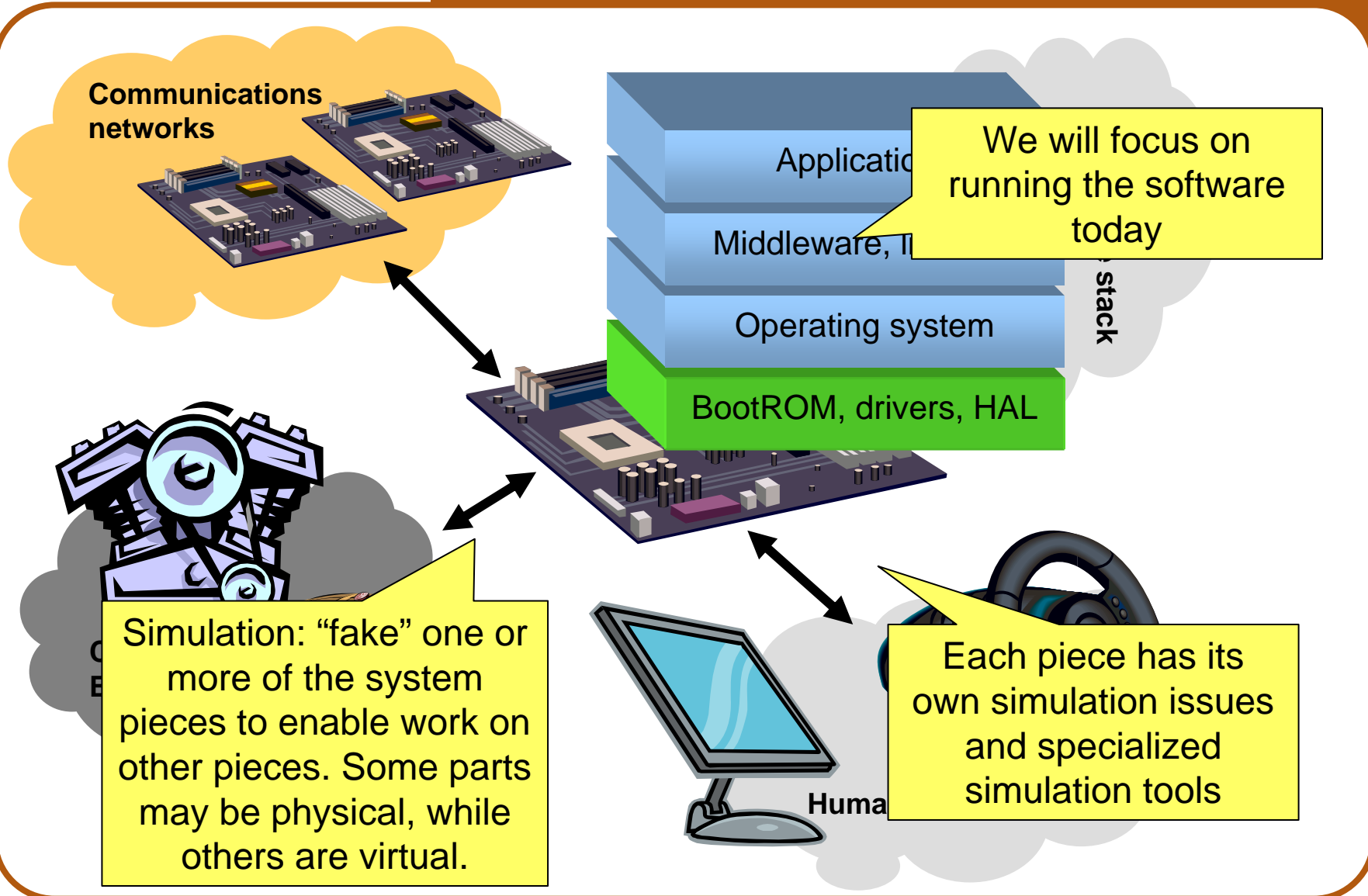


Simulating Computer Systems

Embedded Computer System



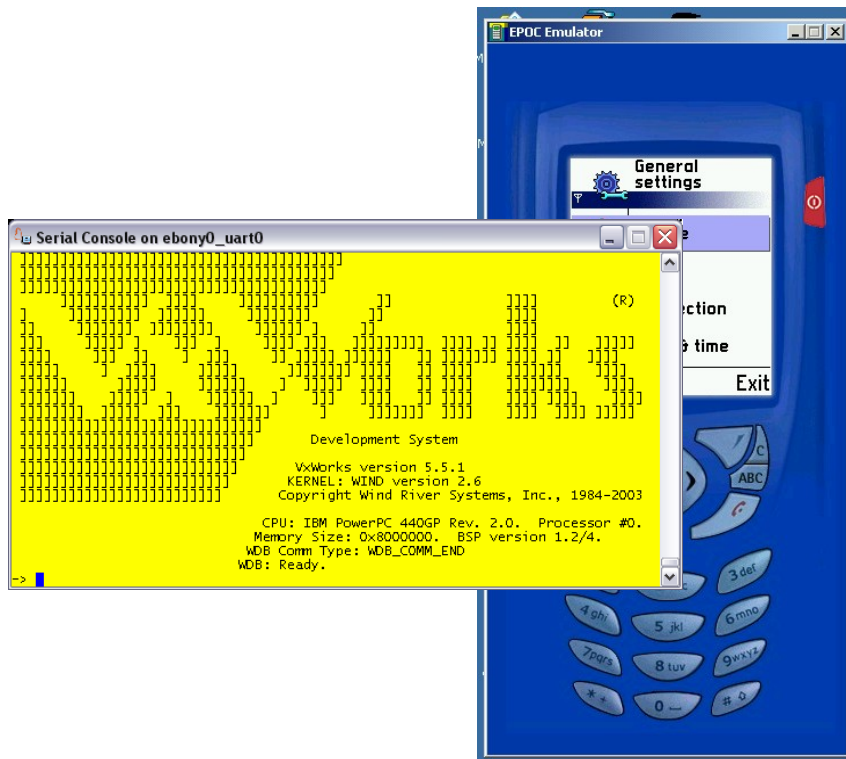
Simulating Embedded Computer System



- Some simulation work is motivated by the clear advantages of simulation over real hardware
 - Control, visibility, access
 - No damage to physical objects
 - Faster turn-around times for experiments
- Much simulation work is motivated by the necessity of avoiding physical hardware – even when hardware is considered the ideal solution
 - Avoid inconvenience
 - Save costs
 - Get around slipping hardware schedules
 - Insufficient number of real systems available

User Interface Simulation

- A category of tools of its own
- Part of many other simulation tools



- Many different levels:
 - Virtual screen & mouse like VmWare and its ilk
 - Clickable simulation of touch screens
 - Clickable panels of buttons
 - Graphics displays, text displays, LEDs, etc.
 - Full hardware mockups connected over CAN bus to PC
- Software:
 - Simulated by scripts
 - Special code for special API
 - Actual target code in some form of other simulator

- Simulation gets closer to real world
 - More details
 - Fewer assumptions
 - High compute cost
- Analytical models
 - Efficient predictors
 - Low computational workload
 - ... but more removed from world=less accurate



Environment Simulation

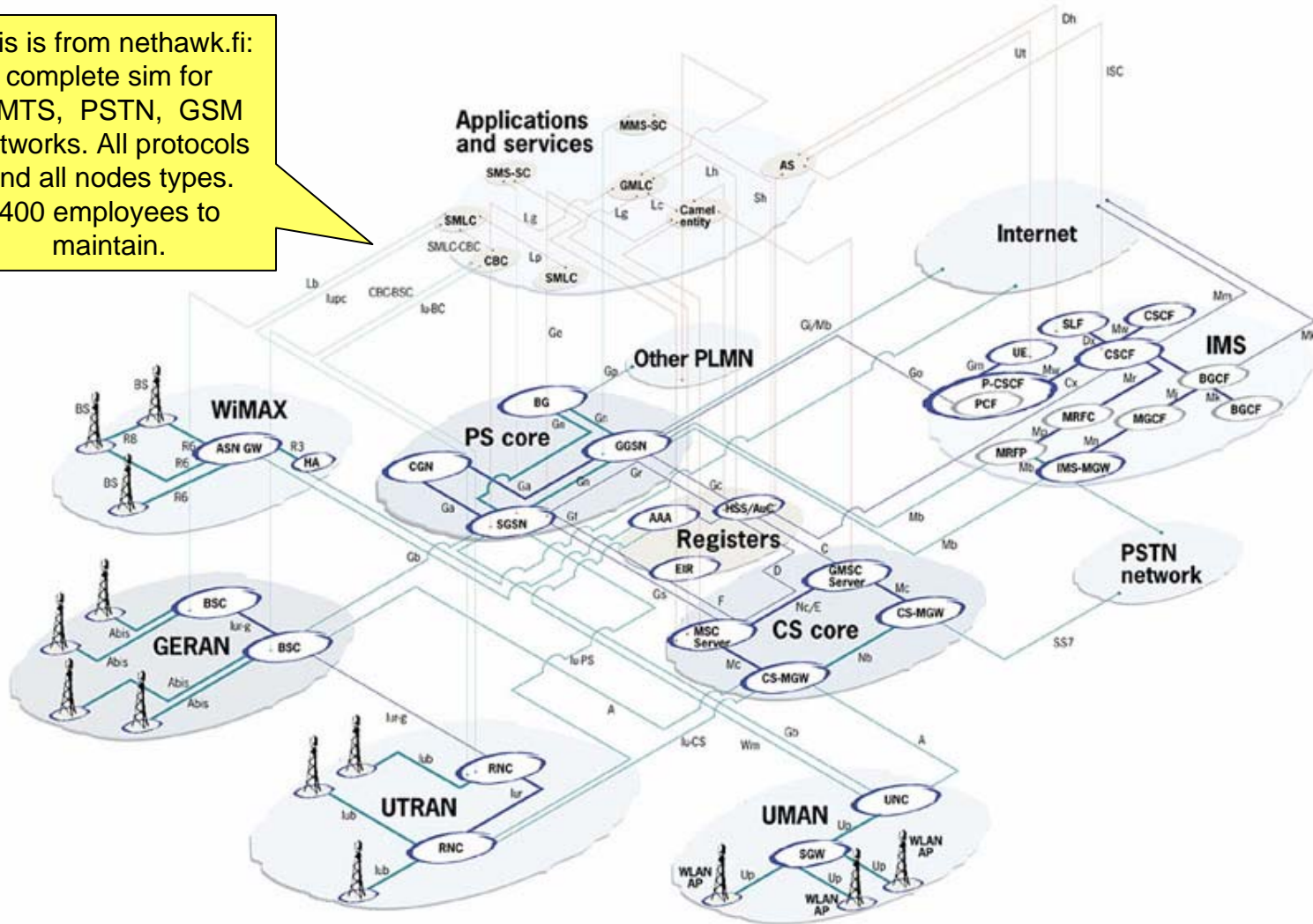
- Very large field
- Largest players:
 - MatLab/Simulink
 - LabView/Matrxix
- Everybody is using it, CAD has been doing mechanical simulations for 50 years
- Commonly used for control algorithm development
- Key part of the model-driven architecture/model-driven design paradigm in automotive

Network Simulation

- Connections between abstracted nodes, to study communication patterns
 - Contains models of node behavior, no actual code
 - ns2
- “Rest of network simulation” to provide environment for a single node
 - Understands protocols
 - Vector CANOE
 - Nethawk tools
- Dumb traffic generation
- Move packets between simulated nodes
 - No understanding of protocols
 - Runs real code on all nodes
 - Virtutech Simics
- Integrate physical and simulated nodes
 - “Hardware in the loop”
- Networks:
 - Ethernet, AFDX, CAN, LIN, FlexRay, MOST, PCIe, I2C, LonWorks, ARINC 429, MIL-STD-1553, serial, RapidIO, VME, SpaceWire, USB, FireWire, ...

Rest-of-Network Example

This is from nethawk.fi:
complete sim for
UMTS, PSTN, GSM
networks. All protocols
and all nodes types.
400 employees to
maintain.





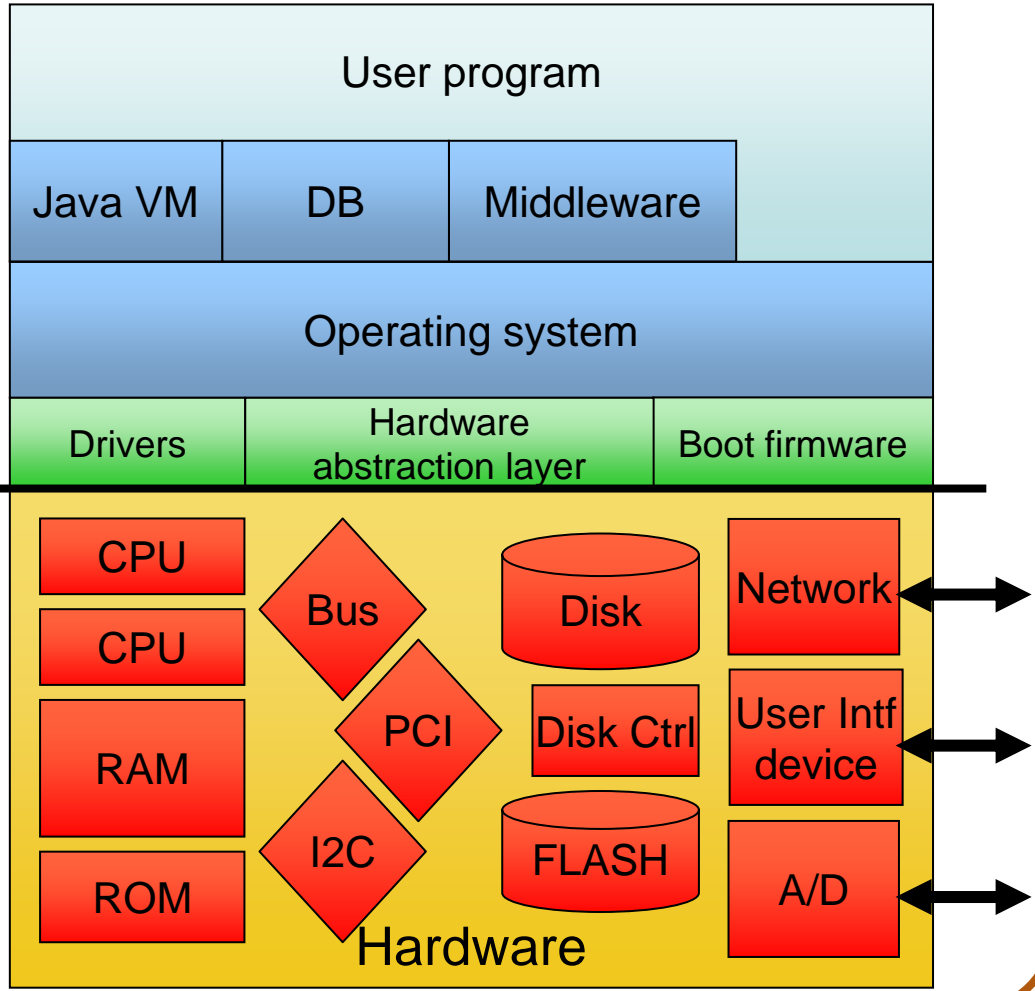
Simulating (for) the Software

“Virtual Software Development”

Simulating the Software Stack

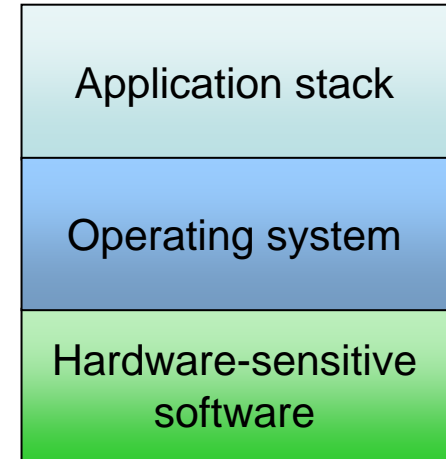
Goal: run your embedded software on a PC instead of on a physical target

HW/SW interface

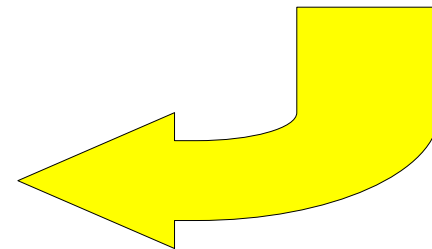


Traditional Software Development

- Software development methodology creates production binary
- Production binary runs on the real hardware

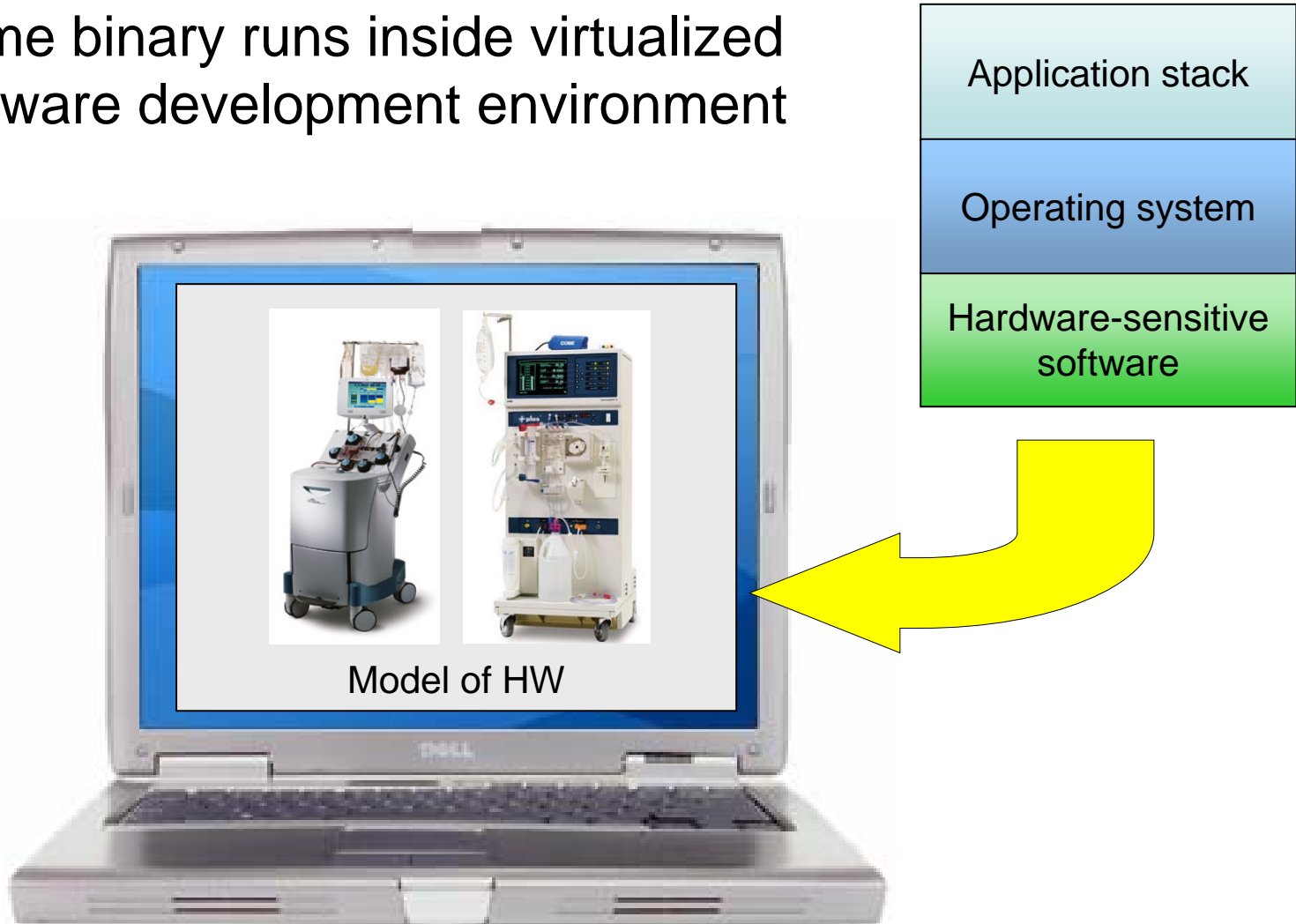


Actual hardware



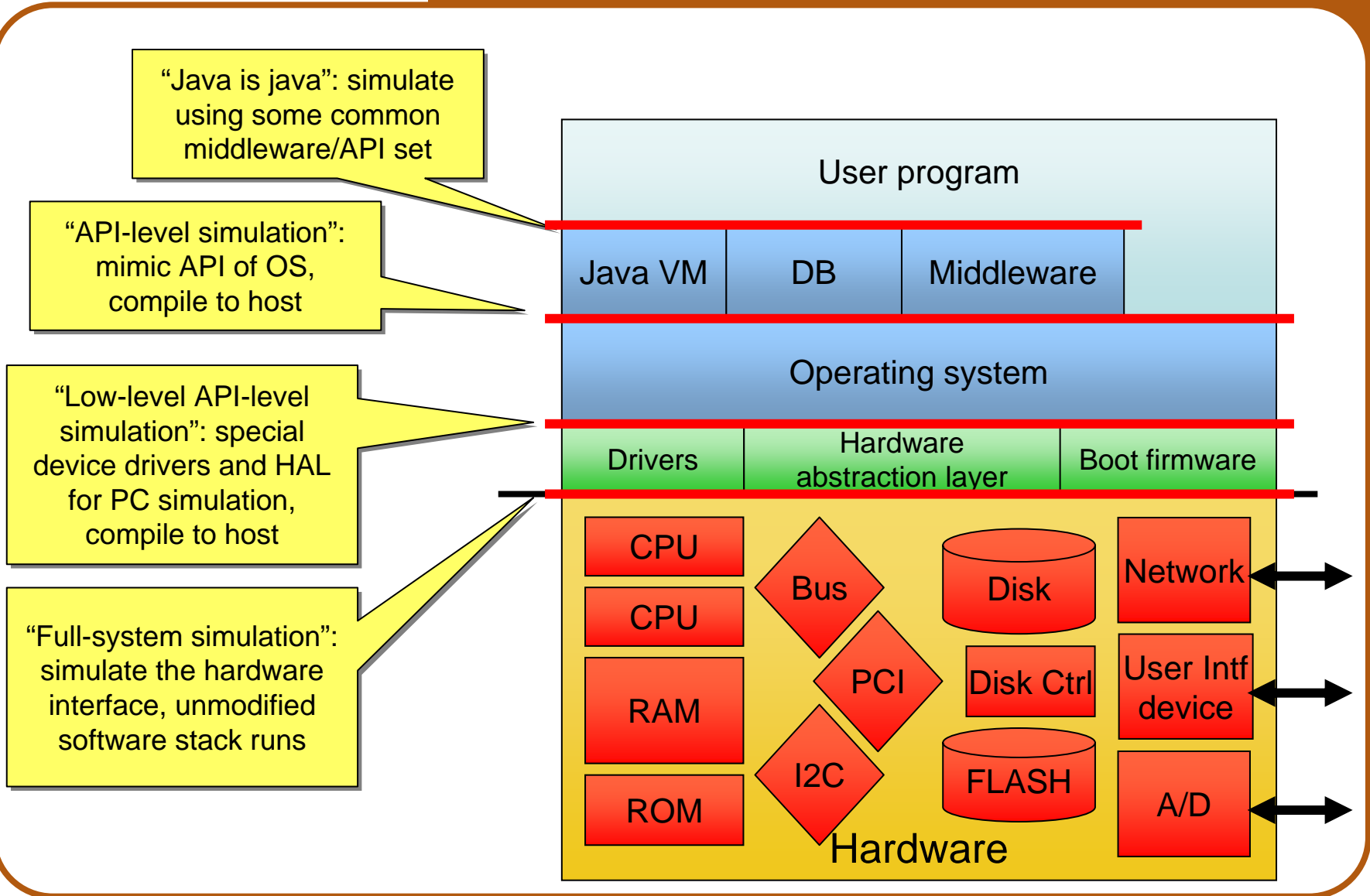
Virtualized Software Development

- Same binary runs inside virtualized software development environment

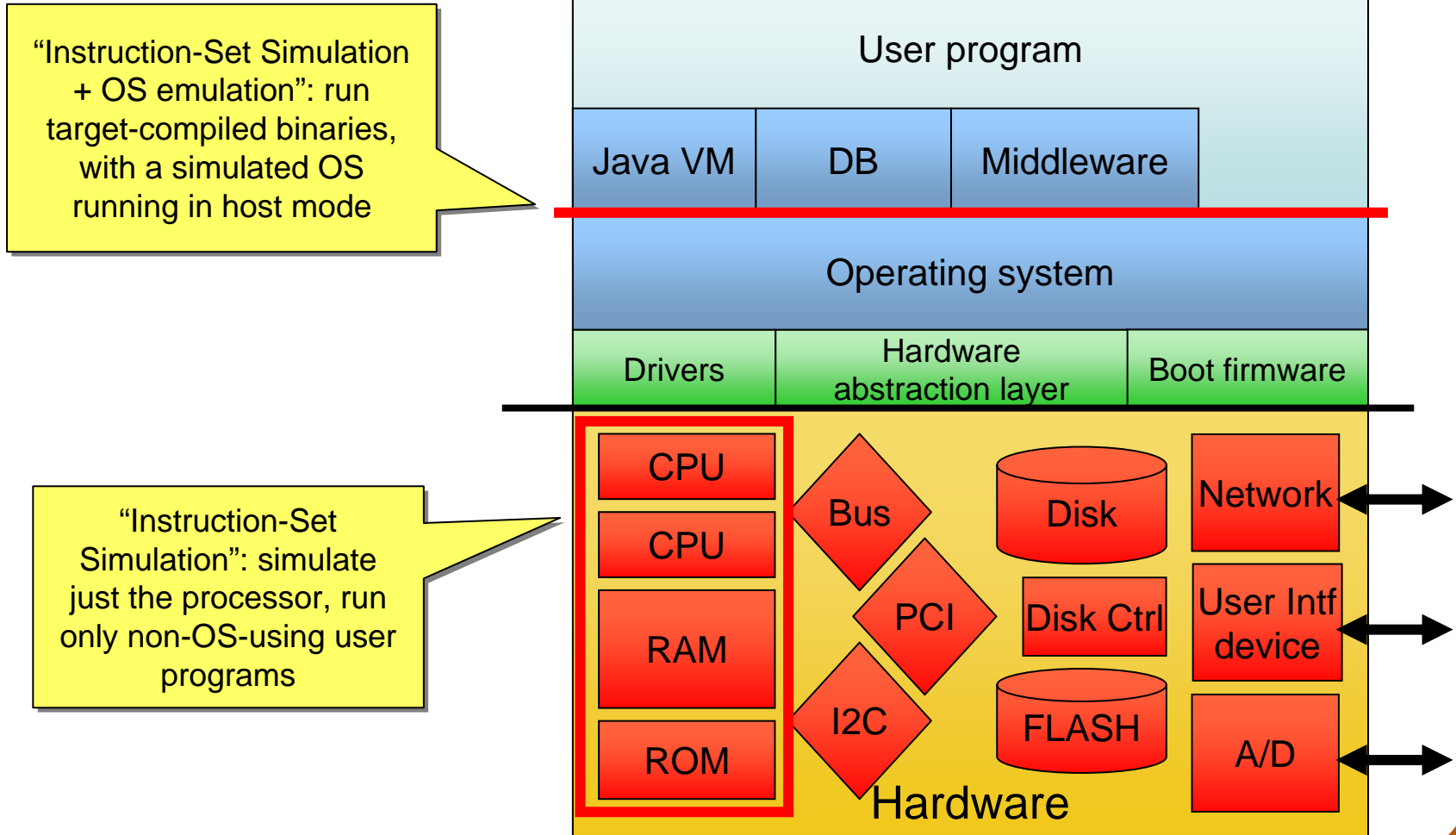


- **Host-compiled or target-compiled?**
 - Host: Compiled to x86 to run on PC, for simulation only
 - Target: Compiled to actual target code
- **Software stack completeness**
 - Which pieces of the software stack are the same as on target, which are faked-out or stubbed?
 - Which types of code can be tested in simulation?
- **OS Behavior**
 - Same scheduling as real target?
 - Same memory protection?
 - Same memory allocation and limitations?

Simulating the Software Stack



Interesting Hybrids



Simulation Detail Levels

- SystemC defines five levels of model abstraction which offer a good map of abstraction levels for system modeling

	Meaning	Style	Comments
AL	<i>Algorithmic Level</i>	<i>UML, SDL, Matlab models</i>	<i>Not concrete enough to run software</i>
PV	Programmer's View	Transaction-level with simple time	Can reach 100 MIPS+ performance, fast enough for SW dev
PVT	Programmer's View, with Timing	Transactions with precise time	Slower than PV, but more detail. Can reach 10 MIPS+
CC	<i>Cycle-Callable</i>	<i>Bit-level with precise timing</i>	<i>Very slow, <1 MIPS,</i>
RTL	<i>Implementation</i>	<i>VHDL/Verilog implementation</i>	<i>Seriously slow</i>

System Simulation use Cases

- **System-on-Chip Design**
 - Focus on hardware designer needs
 - Architecture exploration
 - Sizing, performance, optimization of hardware
- **Fidelity to target** is primary driver for models
 - Timing
 - Bandwidth
 - Latency
 - Bus structure
- All components are equals

- **Software Development**
 - Focus on software developer needs
 - Execute large workloads
 - Debug code
- **Speed of execution** is the primary driver for model
 - Abstract as far as possible
 - Approximate timing
- Work from the processor outwards
- Clear difference between processors and other devices

Enough detail is enough

- Computer games masters of sufficient abstraction
- "Life-like" action:
 - Momentum
 - Friction
 - Steering
 - Engine torque
- But: not nuts & bolts of cars
- **Simulate only what can be observed by intended user**

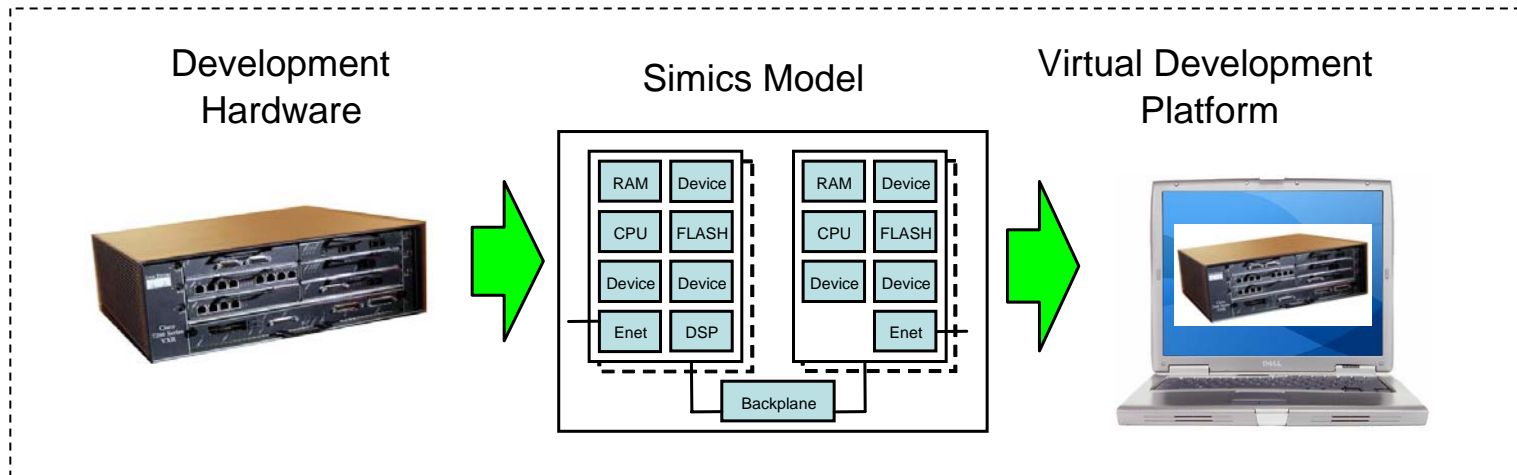


Grand-Prix Legends



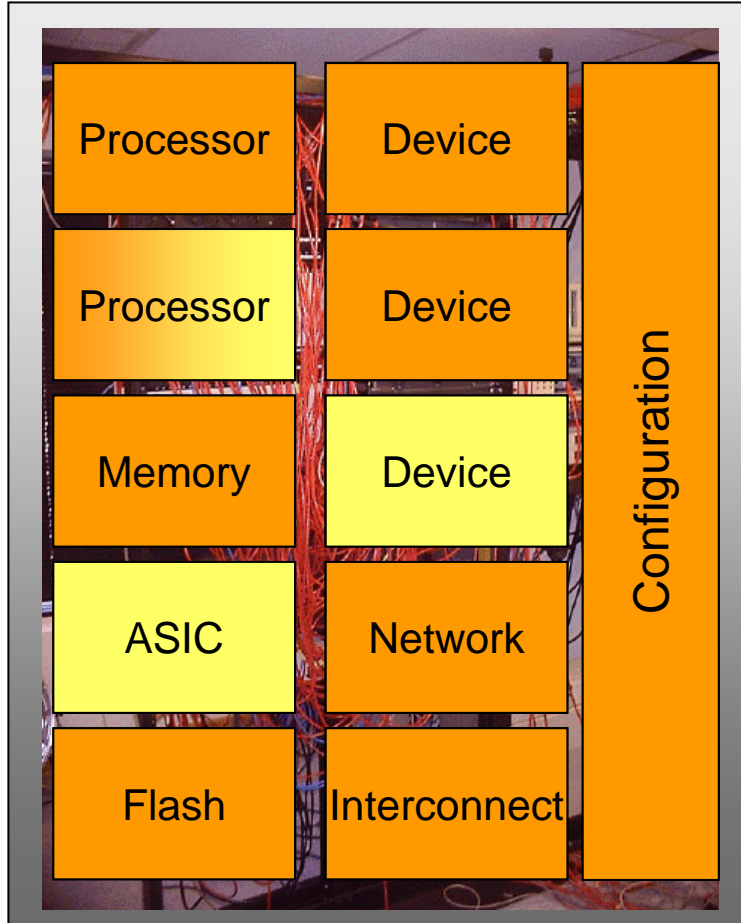
Modeling Hardware for Virtual Software Development

- Prerequisite to obtaining benefits of virtualized software development



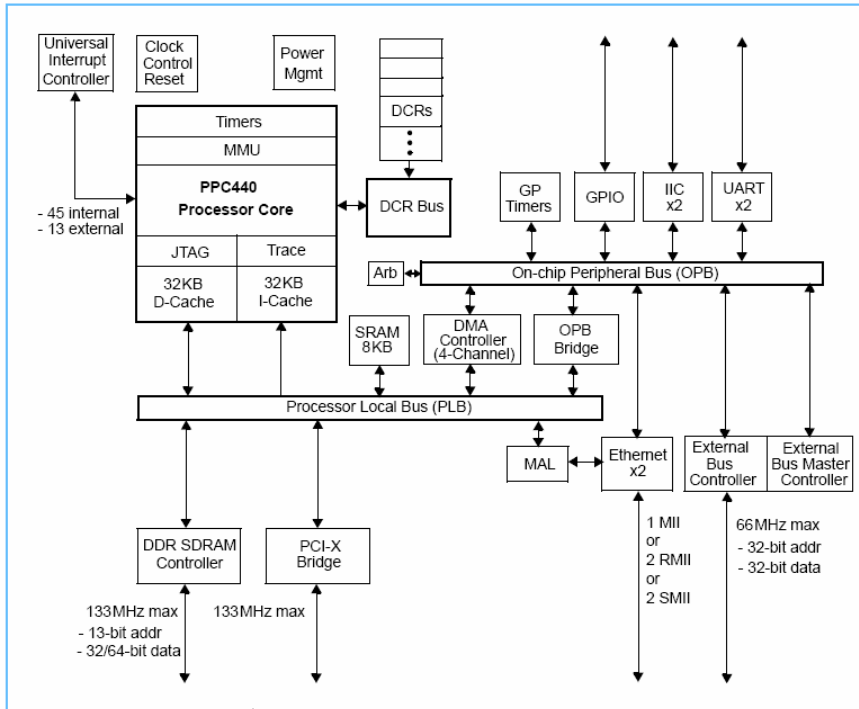
- How do we achieve this?

Modeling Your System

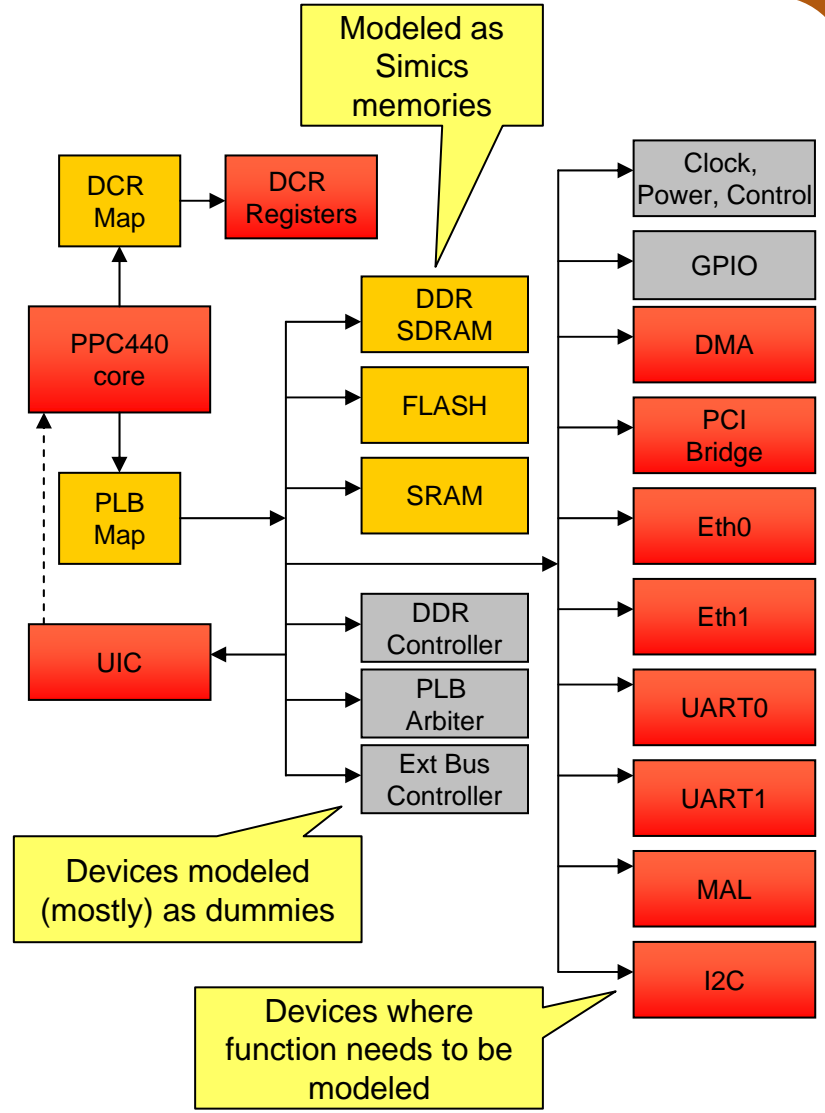


- Use Simics framework
- Reuse VT components
 - Large library available
- Adapt VT components
- Model custom parts
 - DML
 - C, C++
 - Python
- Device modeling by
 - Virtutech
 - Customer
 - Partner
 - Consultant

Mapping a System For Modeling



PPC 440 GP block diagram



Devices modeled (mostly) as dummies

Devices where function needs to be modeled

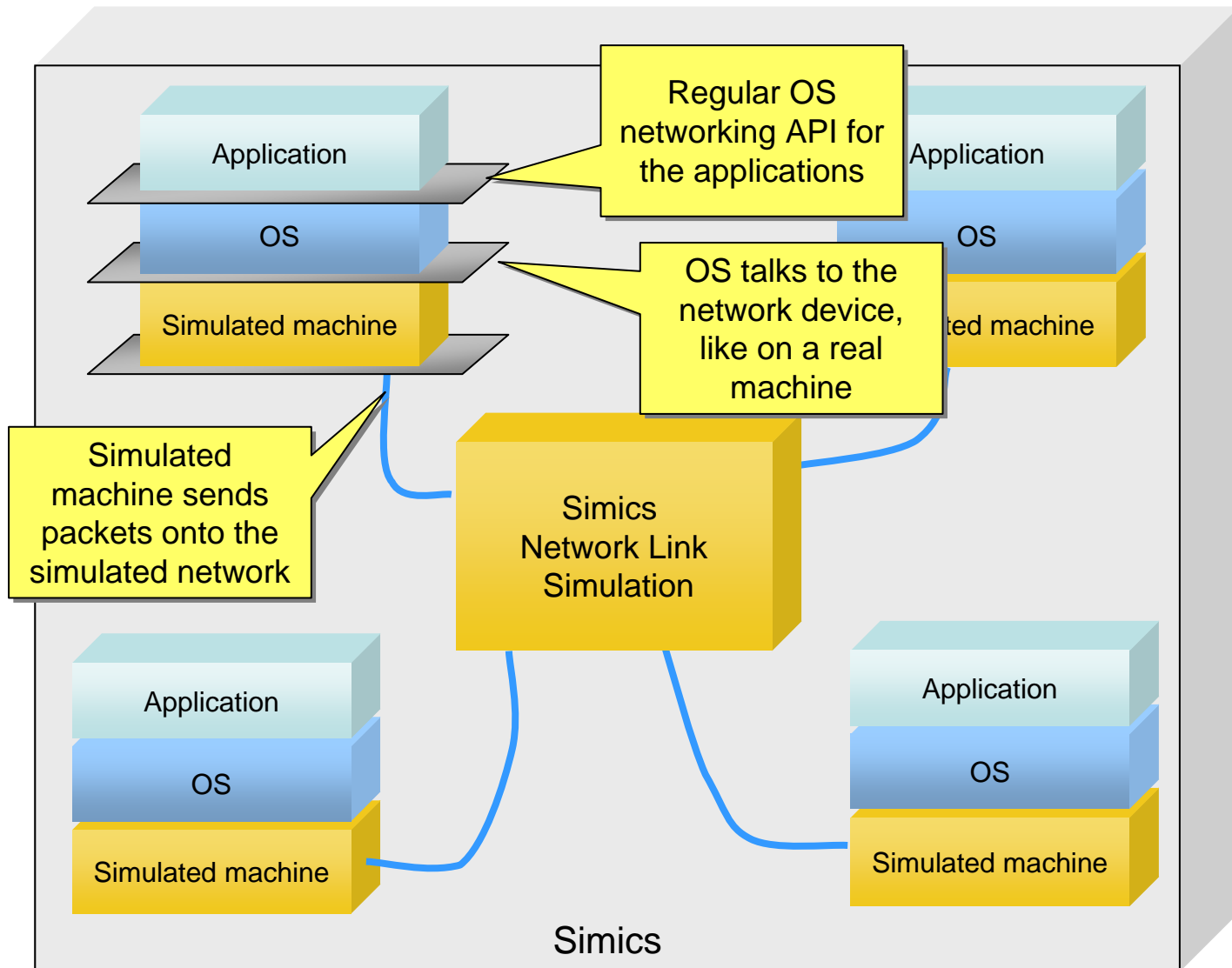
- Declarative
- Domain-specific for Simics
- Efficient coding
 - 5 times smaller than C
 - Quick start modeling
 - Iterative lazy development
- Fast compiled models
- Models redistributable as binaries
- Modeling time:
 - Days to weeks
 - Depends on model complexity

```
bank b {
  register DMA_control size 4 @ 0x20 {
    field EN [31] "Enable DMA";
    field SWT [30] "Software Trigger";
    field TS [15:0] "Transfer size";
    method after_write(memop) {
      inline $do_dma_transfer();
    }
  }
  register DMA_source size 4 @ 0x24;
  register DMA_dest size 4 @ 0x28;

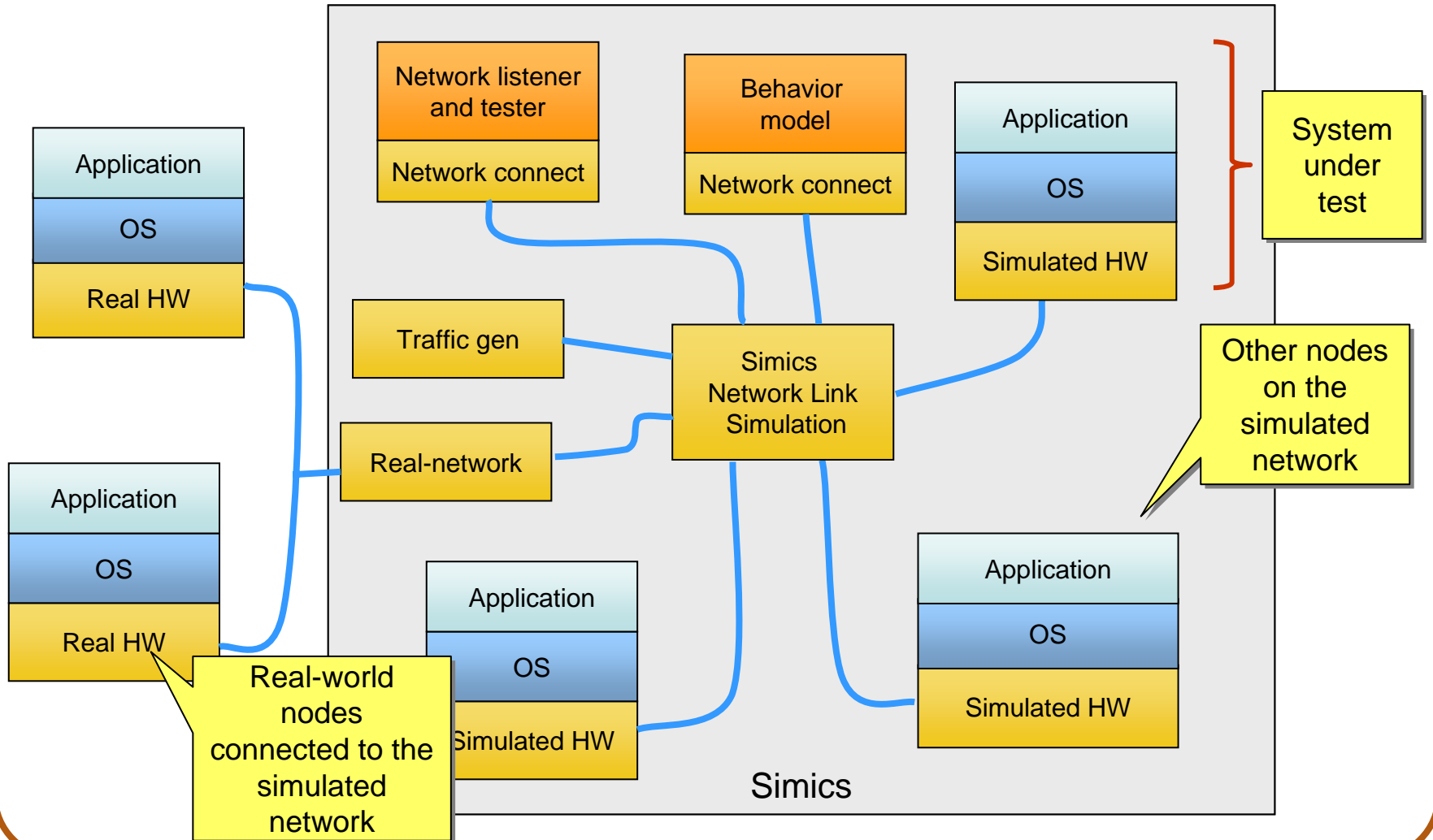
  method do_dma_transfer() {
    if ($DMA_control.EN==1) {
      local uint16 count = $DMA_control.TS;
      local uint8 local_buf[4];
      local exception_type_t result;

      while(count>0) {
        // copy memory details elided...
        $DMA_source += 4;
        $DMA_dest += 4;
        count -= 1;
      }
      // clear SWT bit, update TS
      $DMA_control.SWT = 0;
      $DMA_control.TS = count;
    } else {
      if($DMA_control.SWT==1) {
        // enable bit not set, so we cannot transfer
        log "info", 2 : "EN bit not set, SWT=1 has
no effect";
      }
    }
  }
  ...
}
```

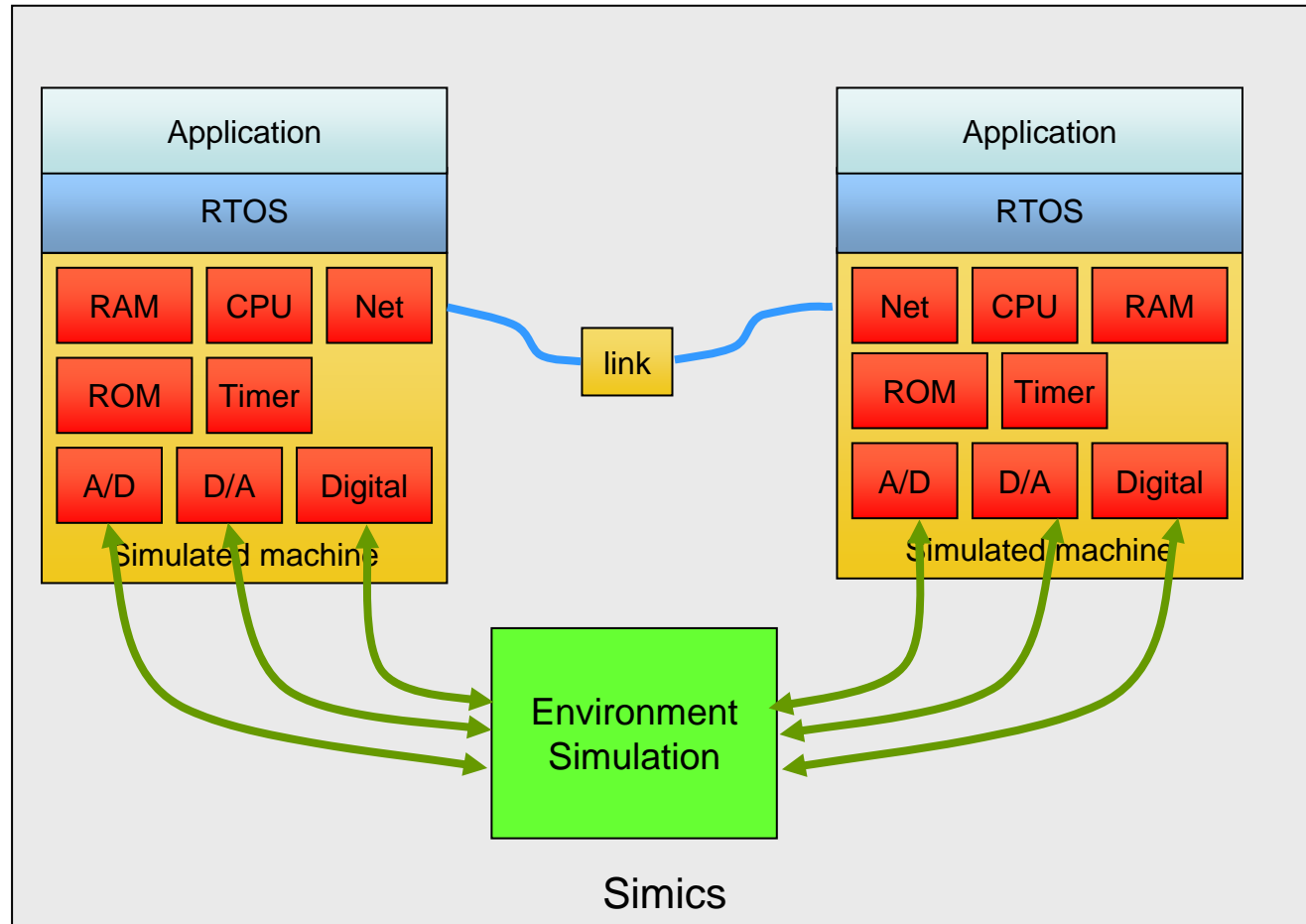
Network Simulation with Simics



Networked System Testing



Simulating Computers in an Environment



Demo: Network Software

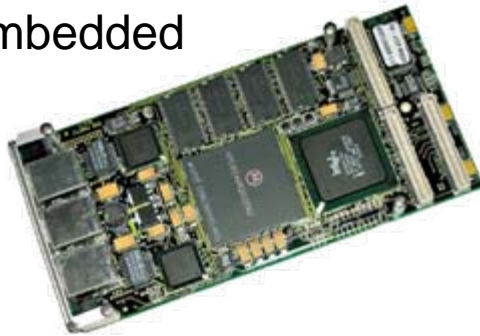




Simics Uses and Examples

What Types of Systems Can Be Handled?

Embedded



Wireless

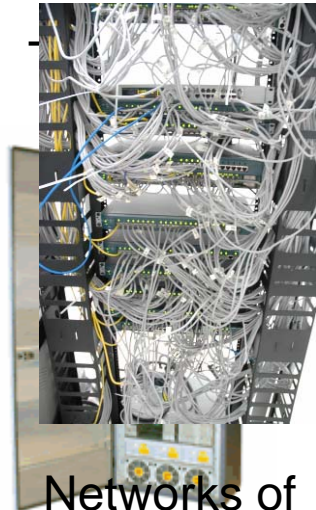


Networking

Servers and Storage



Aerospace & Defense

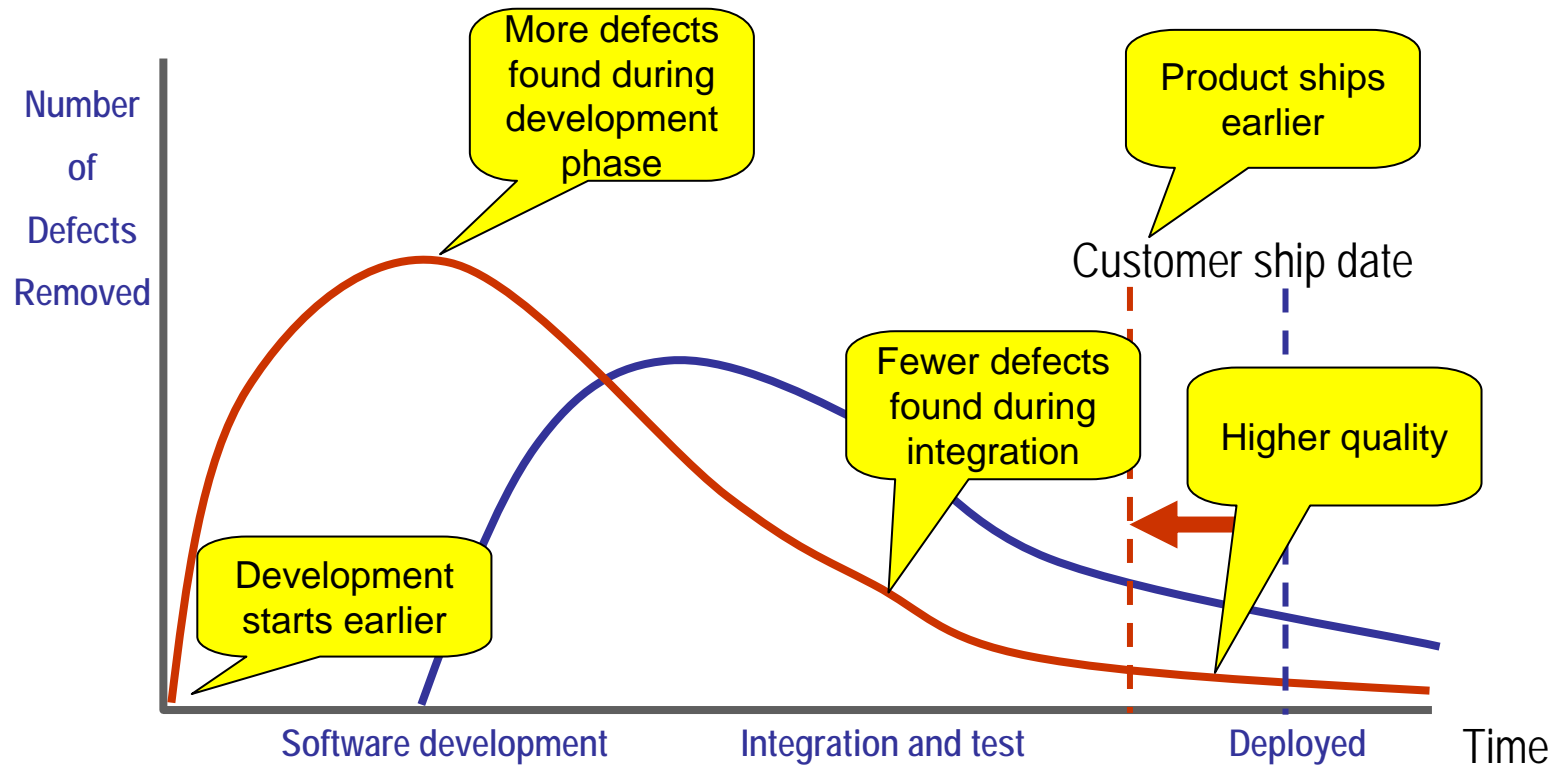


Networks of systems

Driving Quality Sooner

With hardware only —

With virtualized software development —

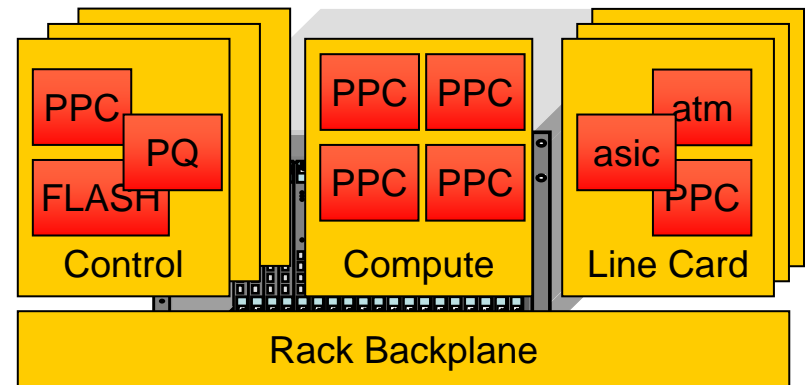


- Problem: Ericsson needed to test software on a large range of base-station configurations
- Challenges:
 - Hardware is expensive and takes 2-14 days to reconfigure before testing
 - Systems can have up to 66 boards and 700 processors
 - Test teams are geographically distributed
- Solution: Create Simics models for each board and handle all re-configuration through scripts
- Benefits:
 - Enormous reduction in cost of capital equipment used for testing
 - Can reconfigure a system almost instantly
 - Can handle even a fully populated system

Ericsson CPP Emulator

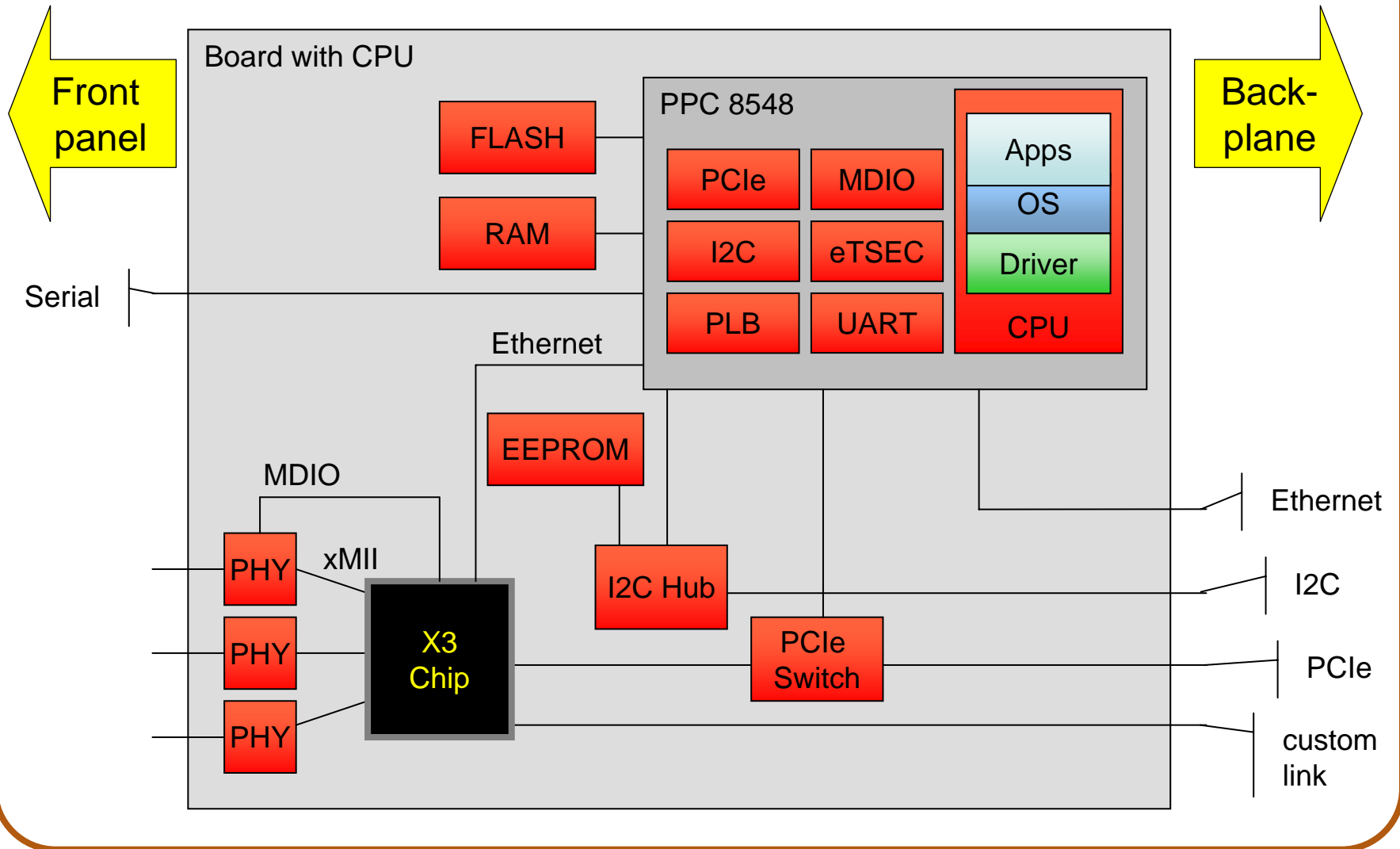
- Telecom Switch
 - ATM Backplane
 - Ethernet frontside
 - Serial
 - 20+ different card types
 - Control cards
 - Timer units
 - Line cards
 - Backplane switch cards
 - Multipro compute cards
 - DSP processing cards
 - 20+ cards in a rack
 - Combined arbitrarily

 - Extreme system size
 - 10-100s processors
 - 10+ of GB target RAM
- Multiple processor types
 - PowerPC 750, 750fx, 750gx
 - PowerPC 403, 405, 440
 - PowerQUICC II 8260, 8270, 8280
 - TI C64 DSP



- Problem: SwitchCore needed to develop and test drivers and protocol stacks for their next generation Xpeedium3 chips
- Challenges:
 - Silicon not yet available
 - SwitchCore customers need to evaluate performance and to develop their own software layers
 - Previously had used an internal simulator but slow and expensive to maintain
- Solution: Model Xpeedium3 using Simics
- Benefits:
 - Internal software development (including offshore)
 - Customers can develop their own software using the same model
 - Reduced delay between prototype availability and production orders

Switchcore Board Components



“We are very happy to partner with Virtutech. Simics will not only accelerate our own development, it will also enable us to share our full software and our functional model with our early access customers.

“With the advanced development and test capabilities provided by Simics, SwitchCore can dedicate additional time and resources to building the functionality of our switch chips”

*Peter Ygberg, Vice-president, product management,
SwitchCore*

- Problem: Wind River needed to develop software for the Freescale MPC8641D

- Challenges:
 - No prototype silicon was available
 - Silicon schedule was slipping but customers still required Wind River support on schedule
 - 8641D is a dual-core chip - this is not a straightforward port

- Solution:
 - Wind River's engineering organization ported VxWorks using Virtutech Simics with the 8641D processor model

- Benefits:
 - Development could start ahead of silicon
 - Improved productivity, improved software quality, earlier availability of 8641D software to Wind River's customers

- “By using Virtutech Simics, Wind River has been able to accelerate the development of its products for the MPC8641D. With this simulation model, our engineers not only have pre-silicon access to chips, they can tap into Simics’ unique debugging tool, Hindsight, to move forward and backward in the code with the ability to stop at any point.”
 - Tomas Evensen, Chief Technology Officer
Wind River

System Example: Aerospace SBC

- Aerospace board
- PowerPC 750gx
 - Also PPC 7447
- Marvell system controller
 - Memory controller
 - Ethernet network
 - Serial ports
 - PCI
- On-board memory
 - FLASH
 - RAM
- PCI-X expansion
- Ethernet/AFDX network
- Mil-Std-1553 network
- VxWorks, Linux, in-house OS

