Simulation and the Multicore Revolution

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

virtutech

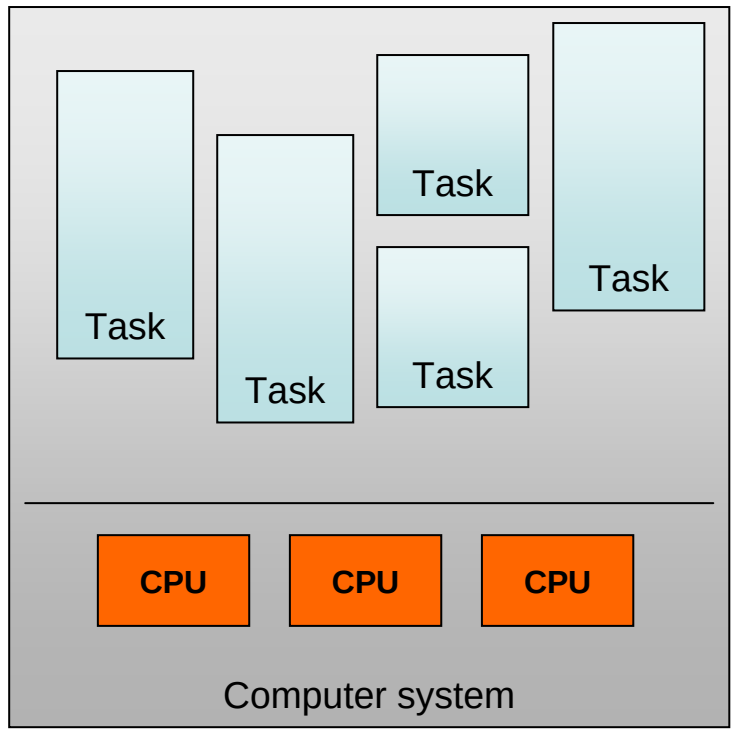
The Multicore Revolution is Here!

- The massive move to parallel computers instead of single processors has been trumpeted before.
- This time it is for real. Why?

- More instruction-level parallelism hard to find
  - Very complex designs needed for small gain
- Clock frequency scaling is slowing drastically
  - Too much power and heat when pushing envelope
- Cannot communicate across chip fast enough
  - Better to design small local units with short paths
- Effective use of billions of transistors
  - Easier to reuse a basic unit many times
- Potential for very easy scaling
  - Just keep adding processors/cores for higher performance

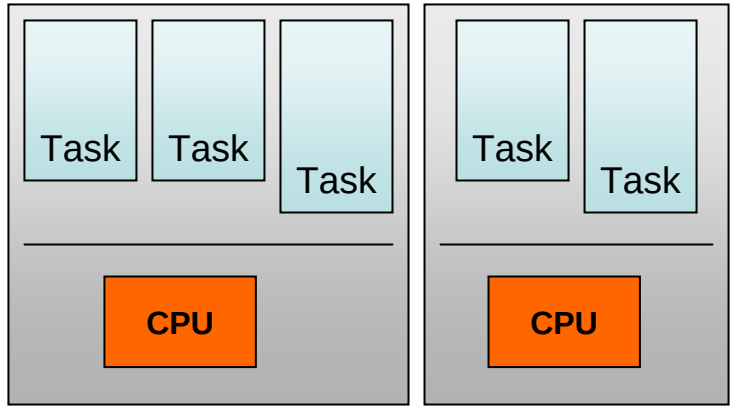
Vocabulary in the “Multi-” Era

- **Multitasking**: Multiple tasks running on a single computer
- **Multiprocessor**: Multiple processors used to build a single computer system

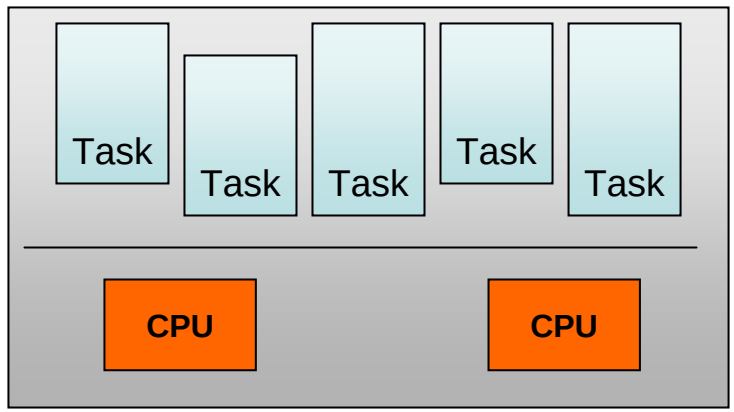


Vocabulary in the “Multi-” Era

- **AMP, Assymetric MP**: Each processor has local memory, tasks statically allocated to one processor

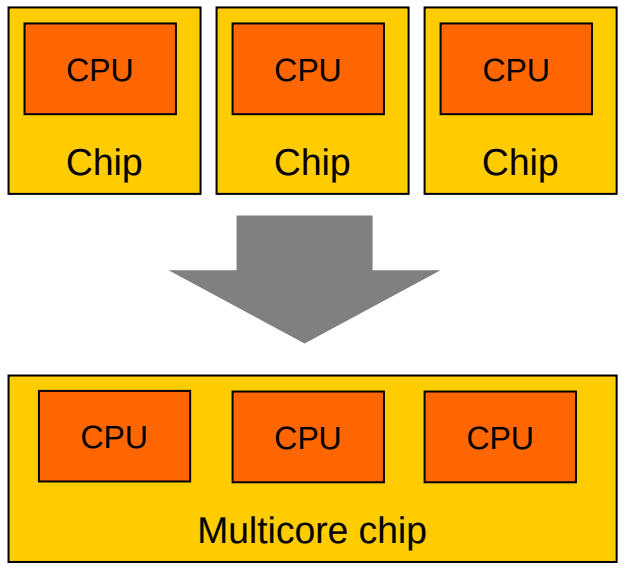


- **SMP**, **Shared-Memory MP**: Processors share memory, tasks dynamically scheduled to any processor



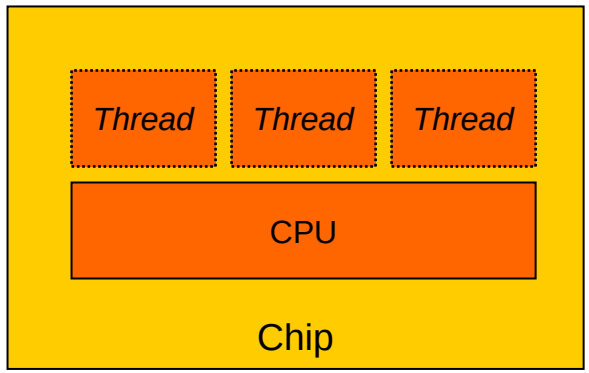
Vocabulary in the “Multi-” Era

- **Multicore**: More than one processor on a single chip, AMP or SMP or hybrid
- **CMP, Chip Multiprocessor**: Shared-memory multiprocessor on a single chip



Vocabulary in the “Multi-” Era

- **MT, Multithreading**: One processor appears as multiple thread.
  The threads share resources, not as powerful as multiple full processors.
  Very efficient for certain types of workloads.



Multicore Processors History

- Multiprocessors have been around since the 1950's
  - 1959: Burroughs D825, 1960: Univac LARC, 1965: Univac 1108A, IBM 360/65, 1967: CDC6500, 1982: Cray X-MP
- Multicore is more recent – but already prevalent
  - 1995: TI C80: video processor: RISC + 4xDSP on a chip
  - 2001: IBM Power4: 2 cores, first non-embedded multicore
  - 2002: TI OMAP 5470: ARM + DSP on a chip
  - 2005: Sun UltraSparc T1/"Niagara": 8 cores x 4 threads, AMD Athlon64 2 cores, IBM XBox 360 3 cores x 2 threads
  - 2006: Intel Core Duo, Freescale MPC8641D, IBM Cell, ...



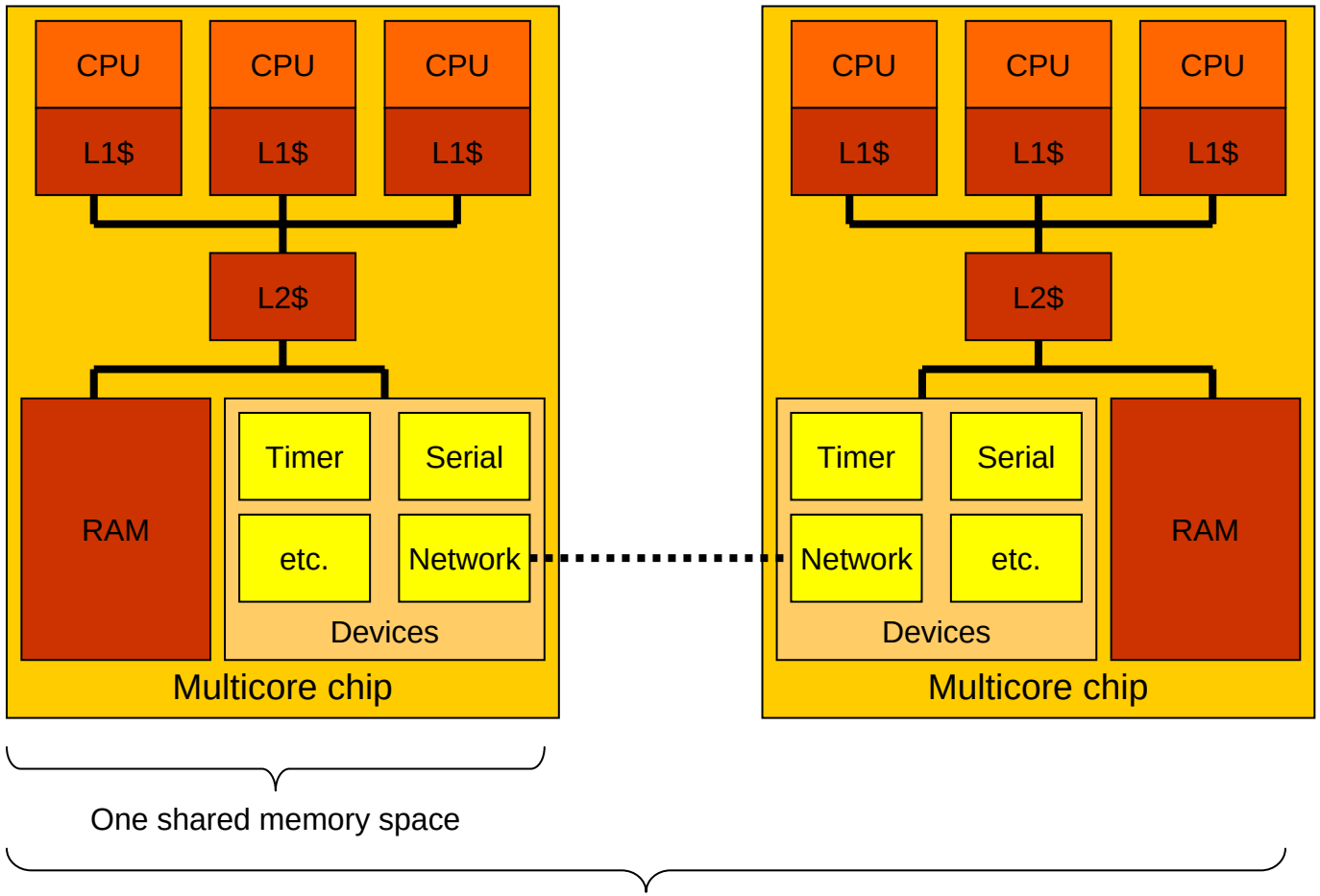
virtutech

Multiprocessing is the Future

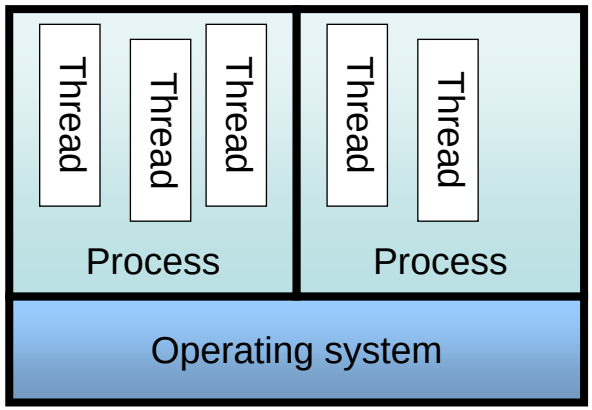
- Multiprocessor and multicore systems are the future
  - Dominant in server market since 1980s
  - Prevalent in SoC design since 2000
  - Standard on the desktop in 2006
- **Now the only option for maximum performance**

Vendor	Chip	Max #Cores	Arch	AMP	SMP
ARM	ARM11 MPCore	4	ARMv6	X	X
Cavium	Octeon CN38	16	MIPS64		X
Freescale	MPC8641D	2	PPC	X	X
IBM	970MP	2	PPC64		X
IBM	Cell	9	PPC64,DSP	X	
Raza	XLR 7-series	8	MIPS64		X
PA Semi	PA6T custom	8	PPC		X
TI	OMAP2	3	PPC,C55,IVA	X	

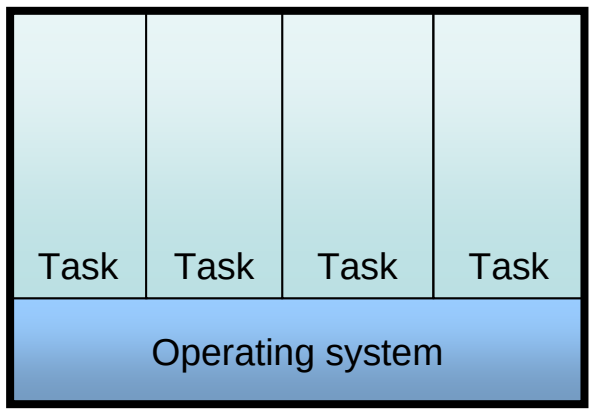
Future Embedded Systems Template



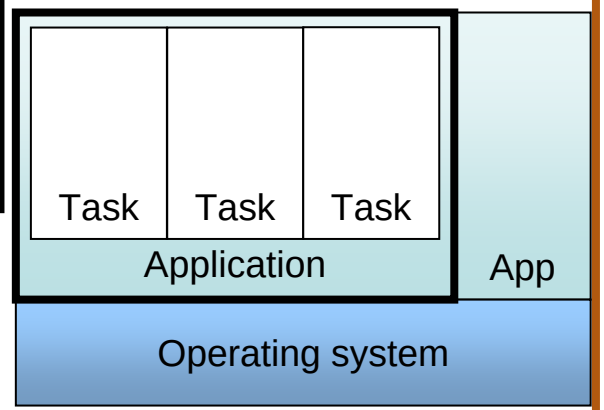
Software Structure



Desktop/Server model: each process in its own memory space, several threads in each process with access to the same memory



Simple RTOS model: OS and all tasks share the same memory space, all memory accessible to all



Generic model: a number of tasks share some memory in order to implement an application

It’s All About Software

virtutech

Electronics Is Software

Electronics is software. Shipping a system is largely about identifying and removing defects from the software and keeping them from creeping back in as the product evolves.

St. Jude Medical Cardiac Defibrillators

St. Jude Medical Inc., a Canada-based medical technology company, announced in June 2005 that some of its implantable defibrillator models, or ICDs, have a software problem that could cause the heart-shocking device to malfunction.

The Explosion of the Ariane 5

On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off from Kourou, French Guiana. The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million. A board of inquiry investigated the causes of the explosion and in two weeks issued a report. It turned out that the cause of the failure was a software error in the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer. The number was larger than 32,767, th integer storeable in a 16 bit signed integer, and thus the conversion failed.

Los Angeles Times latimes.com

2:43 PM PDT, October 13, 2005

Software Glitch Triggers Toyota Prius Recall

By James F. Peltz, Times Staff Writer

In what's believed to be the first recall of hybrid cars for engine-related problems, Toyota Motor Corp. said today that it is notifying about 75,000 owners of its hot-selling Prius about a potential software glitch that could cause the car to stall or shut down.

The voluntary recall dented the good reliability record of the Prius, whose sales have jumped in the past two years as drivers sought better fuel economy in the face of soaring gasoline prices.

ADVERTISEMENT

The problem involves the hybrid's computer software, rather than mechanical parts,

The Register

Software glitch blamed for CryoSat loss

Rockot still ok to fly again

By Lucy Sherriff

Published Thursday 27th October 2005 14:30 GMT

Get breaking Reg news straight to your desktop - click here to find out how

Officials investigating the loss of the CryoSat mission have revealed that a software glitch in the on board flight control system on the new, upper stage of the rocket was to blame.

There is no fault with the Rockot launcher itself, Russian officials said,

virtutech

Software Even Harder on Multicore

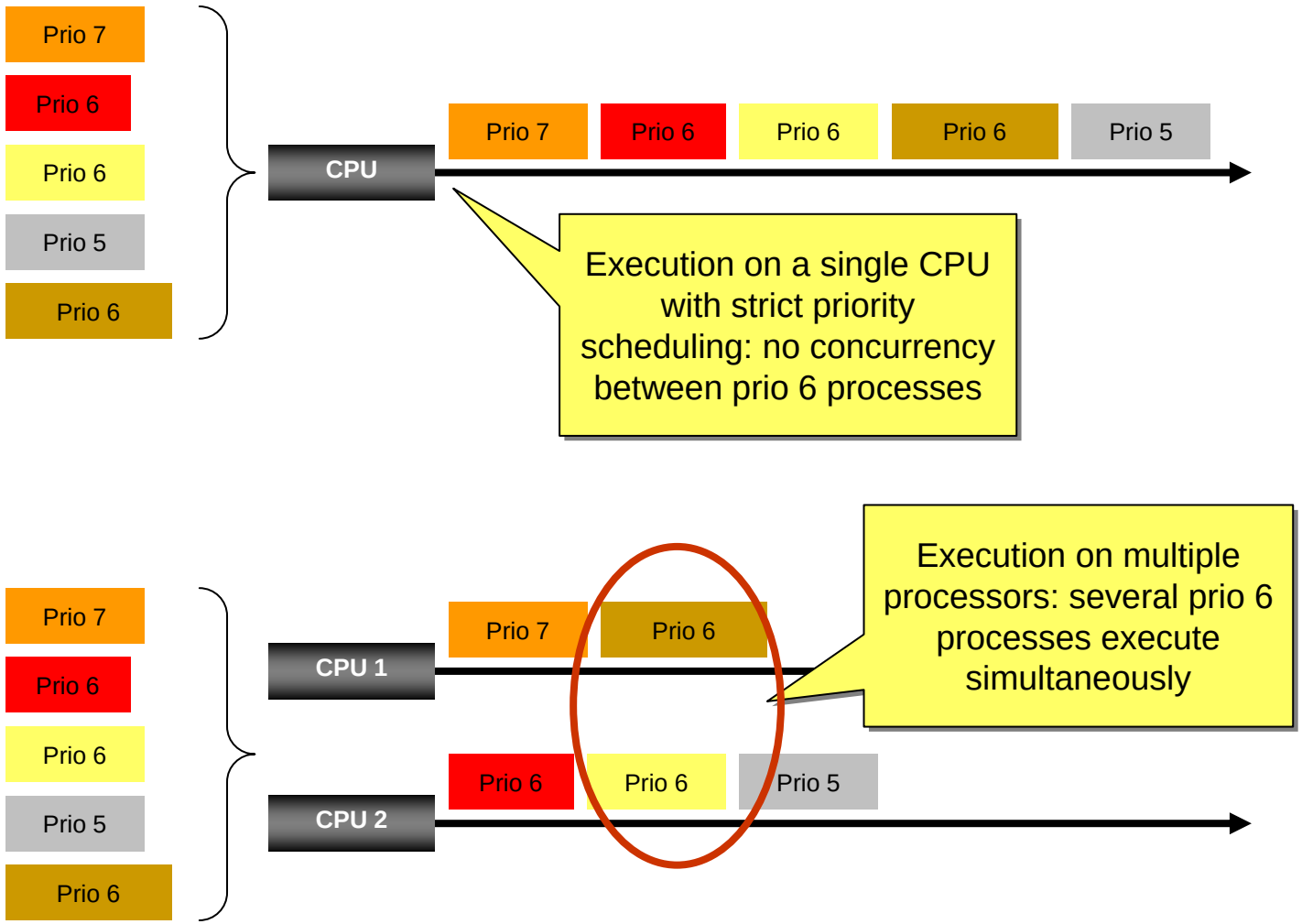
- Parallelism required to gain performance
  - Parallel hardware is “easy” to design
  - Parallel software is known to be hard to write
- Existing software assumes single-processor
  - Multitasking != multiprocessor-ready
  - Multiprocessors expose latent problems in code
  - Might break in new interesting ways on multipro
  - Multitasking no guarantee to run on multiprocessor
- Fundamentally hard to grasp true concurrency
  - Especially in complex software environments
  - Some new phenomena cannot occur on a single processor running multiple threads
- *...some example problems to follow*

virtutech

Disabling Interrupts is not Locking

- Single processor: DI = cannot be interrupted
  - Guaranteed exclusive access to whole machine
  - Cheap mechanism, used in many drivers and kernels
- Multiprocessor: DI = stop interrupts on one core
  - Other cores keep running
  - Shared data can be modified from the outside
- Big issue for kernel porting and low-level code

Same Priority is not Locking



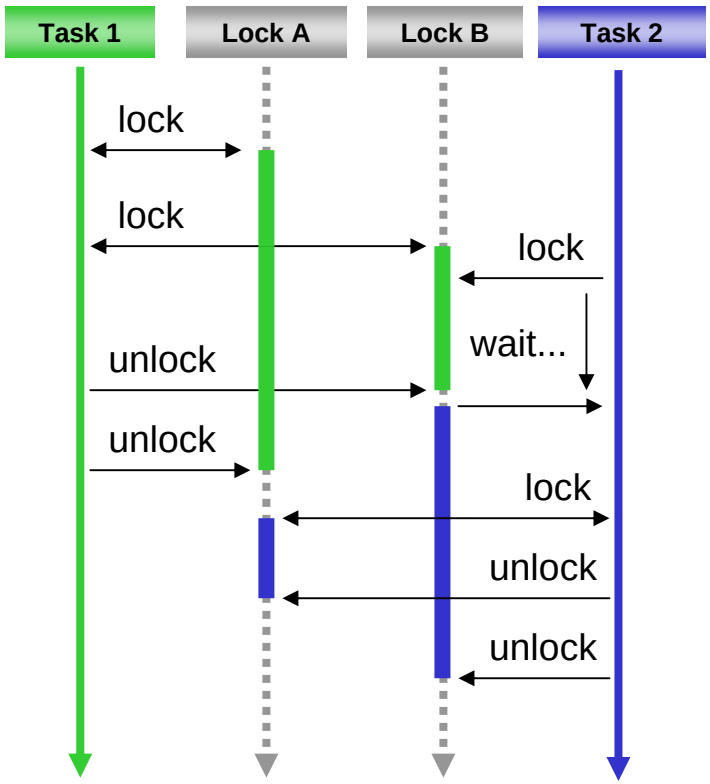
virtutech

Deadlocks

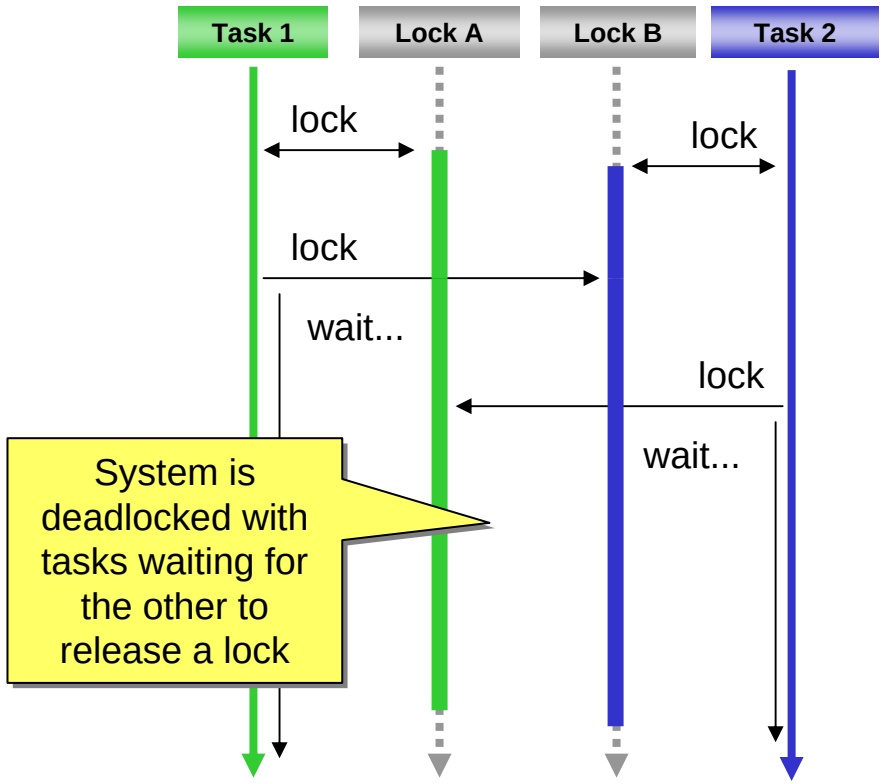
- Locks are intrinsic to parallel programming
  - Necessary to protect shared data, for example
- Taking multiple locks requires care
  - Deadlock occurs if tasks take locks in different order
  - Impose locking discipline/protocol to avoid
  - Hard to see locks in shared libraries and OS code
  - Locking order often hard to deduce
- Also occurs in multitasking, but not as easily

Deadlocks

- Lucky Execution



- Deadlock Execution



Nasty Facts

- Going to two processors from one
- All the trouble
  - Most multiprocessing issues manifest beyond one processor
- Small benefit now
  - At best double performance
- But it opens up for future scaling

virtutech

Multiprocessors & Debug

- Limited visibility into hardware
  - Single debug port, multiple processors
  - High speed, concurrent execution
- Timing-sensitive
  - Small changes in timing alters system behavior radically
  - Hardware variations impact software behavior
- Indeterminism
  - Rerunning a program gives different results
  - Hard to reproduce bugs
- Heisenbugs
  - Inserting probes to trace behavior alters behavior
  - Bugs hide when they are being debugged
- System keeps running even if one core stopped

How Can Simulation Help?

virtutech

Three Steps of Debugging

1. Provoking errors
   - Forcing the system to a state where things break
2. Reproducing errors
   - Recreating a provoked error reliably
3. Locating the source of errors
   - Investigating the program flow and data
   - Depends on success in reproduction

A simulator can help with all three steps

virtutech

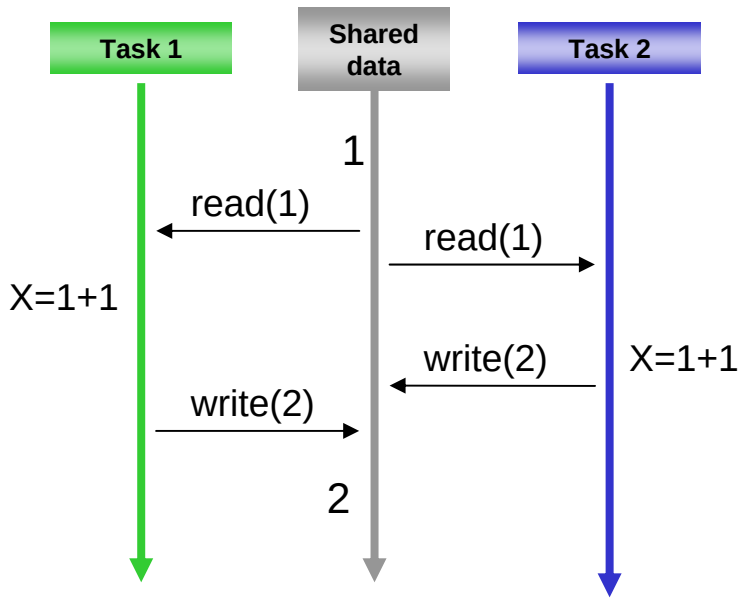
Provoking Errors

- Control over system configuration and execution
- Vary system configuration
  - Number of processors
  - Speed of processors
  - Like testing on a range of real hardware
- Systematically search for error cases
  - Make a processor slower or stop it entirely
  - Increate communication latencies
  - Slow down processors to increase perceived load
  - Guide a system into bad cases (“TimeBending” project)

Pursuit of a Race Condition

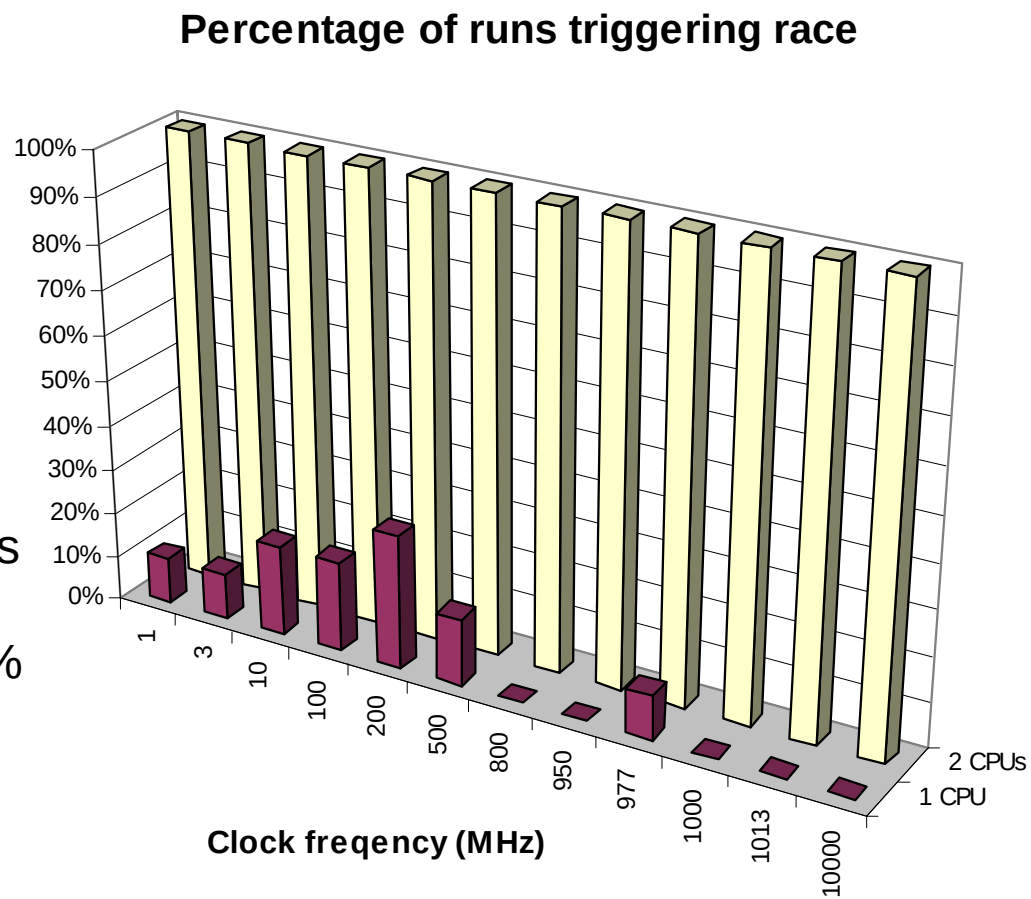
- Test program:
  - Two parallel threads
  - **Loop 100000 times:**
    - **Read x**
    - **Inc x**
    - **Write x**
    - **Wait...**
  - Thanks to Lars Albertsson

- Very severe race condition
- No locks around shared variable read/write
- Will trigger **very** easily
- Symptom: final value of x less than 200000



Pursuit of a Race Condition

- Simulated single-CPU and dual-CPU
- Different clock frequencies
- Test program run 20 times on each
- Count percentage of runs triggering race
- Results:
  - Race always triggers in dual-CPU mode
  - Triggers around 10% in single-CPU mode
  - Higher clock = less chance to trigger



virtutech

Divide-by-zero in OS Kernel

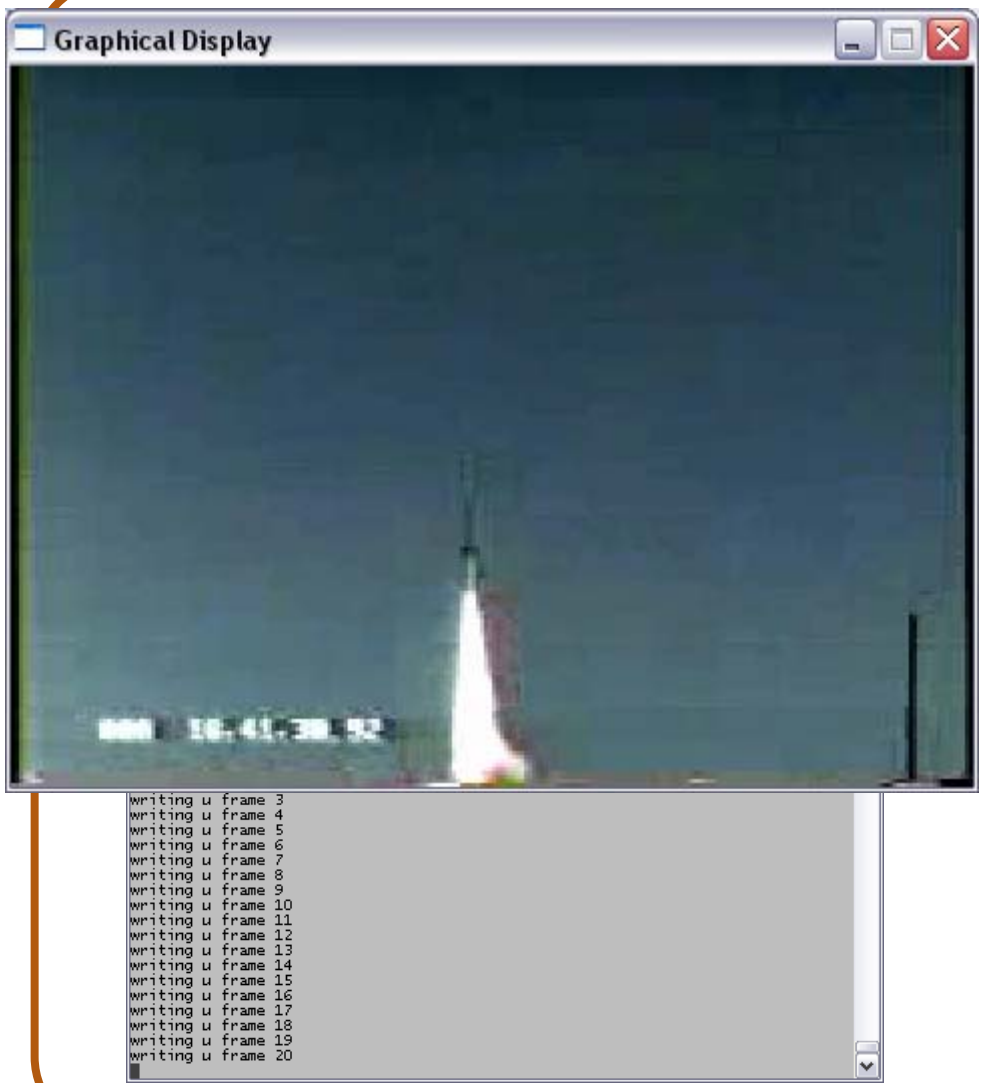
- Operating-system kernel crash in simulation
  - Divide-by-zero
  - Algorithm to determine and compensate for clock skew
  - Division by difference in time between two processors
- Simulation has zero clock skew = provoked error
  - Could have happened on a real system
  - Just not very likely to happen
  - Typical rare problem in the field
- Simulator essentially provided a corner case

virtutech

Reproducing Errors

- Simulation has to be **deterministic**
  - Timing of execution on one processor
  - Communication between processors
  - Handling of asynchronous inputs to simulation
- If an error has been provoked in simulation, it can be reproduced in simulation
  - Go back to initial state (or restore a checkpoint)
  - Execute system forward
  - Replay asynchronous inputs
  - Same error state results
- Better: support the go-back process in the simulator

Demo



- MPC8641D dual-core PowerPC processor
- Playing mpeg2 movie
- Single thread
- Dual thread
- Hindsight to analyze

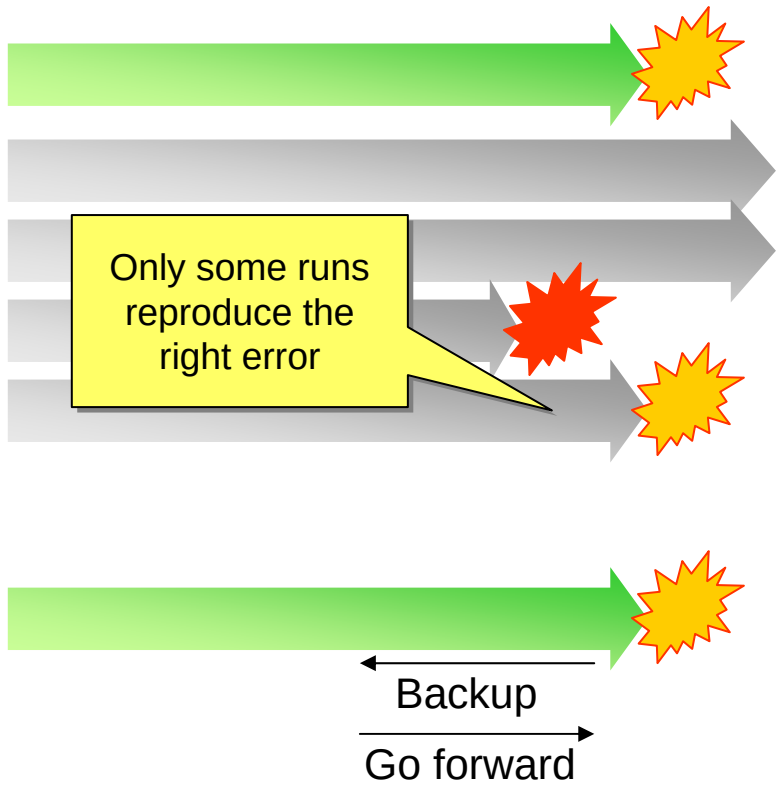
virtutech

Locating and Fixing Errors

- Determinism and control key features
- No probe effect from instrumentation
  - Tracing and observation from the outside, not by code mod
- No timing disturbance from debugging
  - Breakpoints behave like hardware breakpoints
- Global system stop
  - For practical reasons, this has a small skid of 1-100kcycle
- Heisenbugs cannot occur
  - No intrusion in timing behavior, no varying behavior

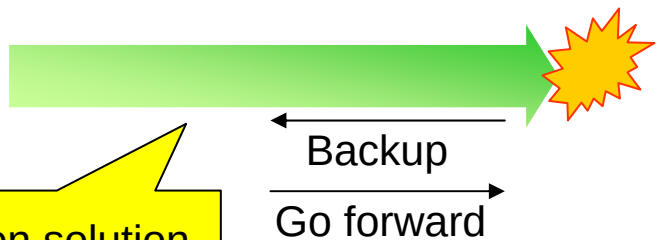
Reverse Debugging

- Stop & go back in time
  - Instead of rerunning program from start
  - No need to rerun and hope for bug to reoccur
  - Investigate exactly what happened this time
  - Breakpoints & watchpoints backwards in time
  - Very powerful for parallel programs

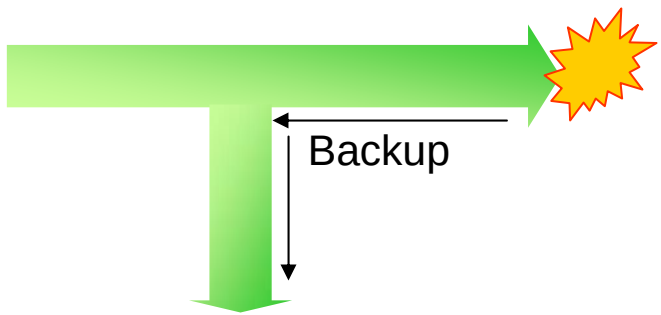


Reverse Debugging Techniques

- Trace-based
  - Record system execution
  - Special hardware support or simulator
  - Use as “tape recorder,” fixed execution observed
  - Hard to extend to multipro
- Simulation-based
  - Record in simulator
  - Replay in same simulator
  - Can change state and continue execution
  - More powerful solution



Common solution in industry, found in tools from GreenHills, IAR, & Lauterbach



And go somewhere else

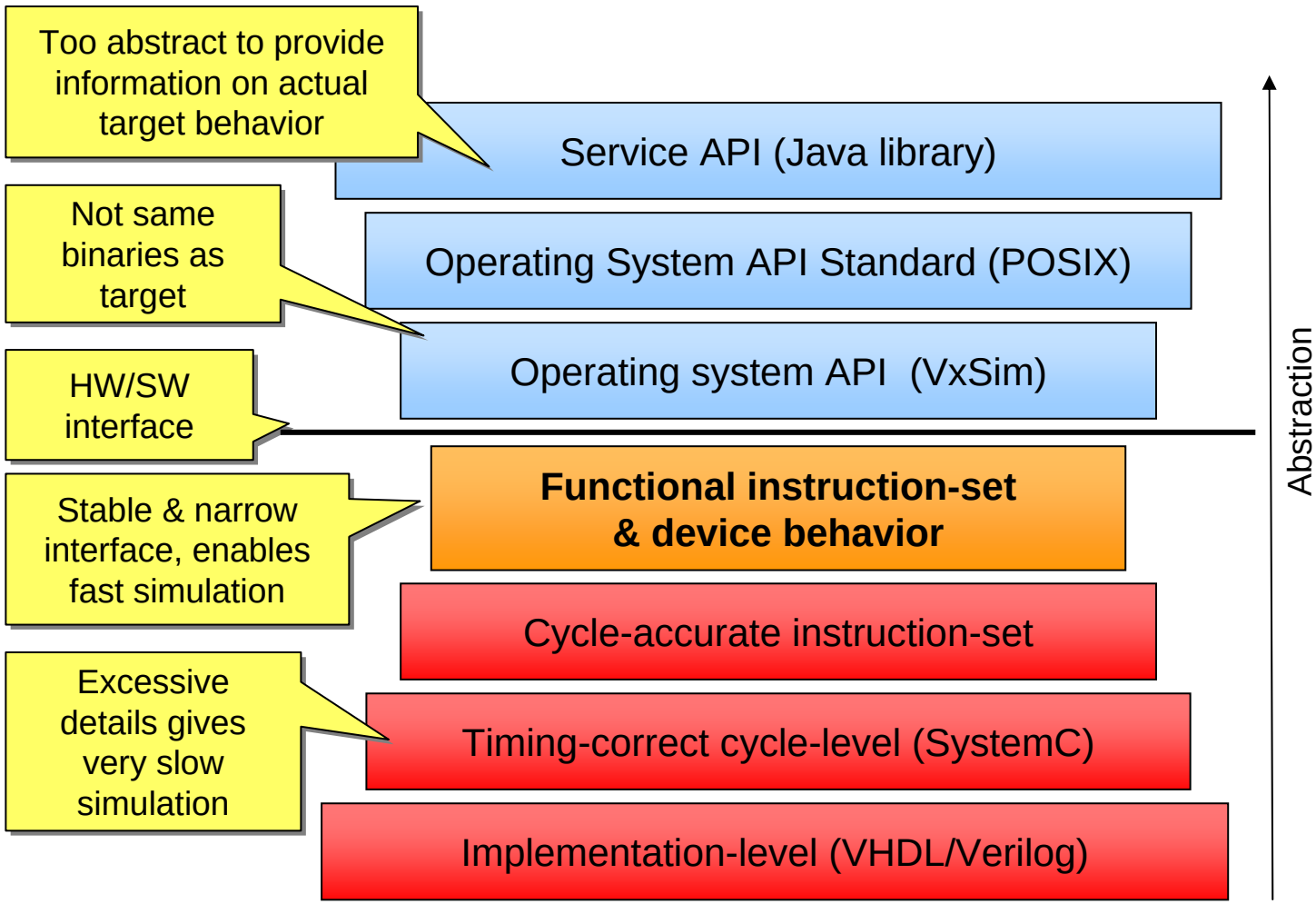
Is Simulation Really Practical?

virtutech

Is It Really Valid?

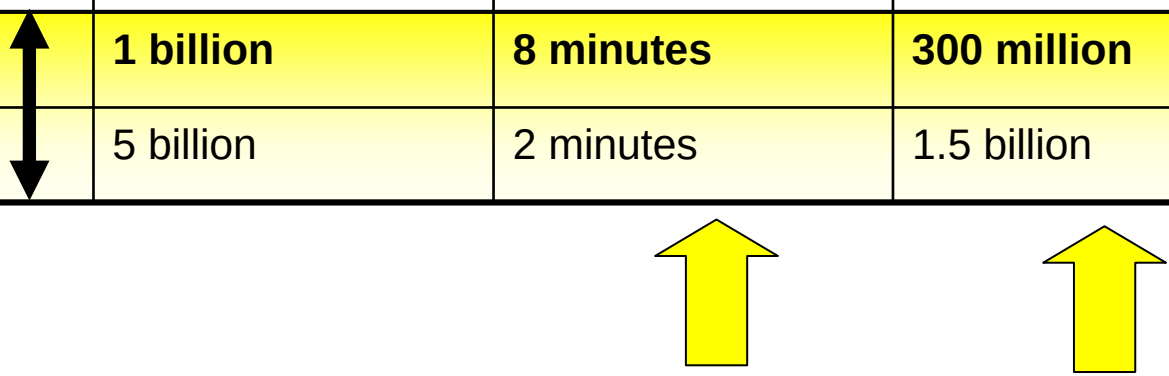
- Function
  - All functional aspects are the same as on a real system
  - We run the real software
- Timing
  - Will differ from real target system
  - But will be coarse-grain representative
- Determinism
  - Variation needs to be explicitly programmed
  - Control over variation rather than random variation
- An error found in simulation is a real error
  - Even if it has not been observed in current hardware
  - It could happen in some unlucky circumstances

Creating a Fast Simulator



How Fast is It?

Technology	Peak MIPS	Full workload	Responsiveness
Circuit	0.01	1.6 million years	0.003
Gate-level	1	16,000 years	0.3
RTL	100	160 years	30
SystemC	10,000	19 months	3,000
Classic instruction set simulator	100 million	1½ hours	30 million
Simics	**1 billion**	**8 minutes**	**300 million**
Native code	5 billion	2 minutes	1.5 billion



Significant workload of 500 billion instructions

Instructions in 0.3 seconds

virtutech

Why is Simics Fast Enough?

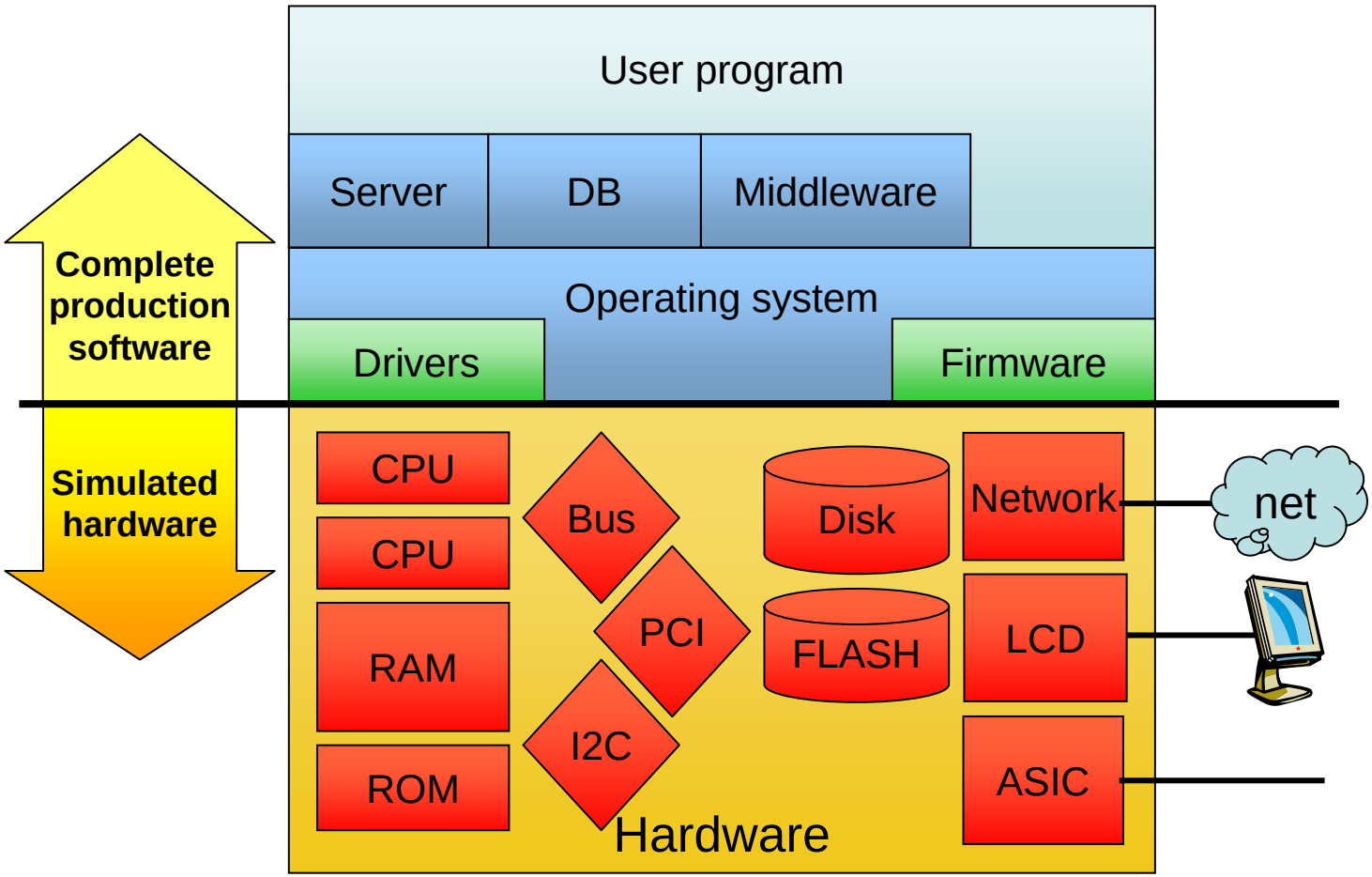
- Raw power of PC hardware
  - Typical x86 server/PC faster than embedded CPU
  - Current hardware much faster than legacy systems
- Simulation technology
  - Binary translation from target to host
- Fast simulation of quiet time
  - Simulation skips idle time (OS idle loops, for example)
- Faster turn-around times
  - Less setup hassle compared to real hardware
  - Use checkpoints to quickly get to interesting points
- Selectable level of instrumentation
  - Only instrument interesting details of system

Creating a Simulation Model

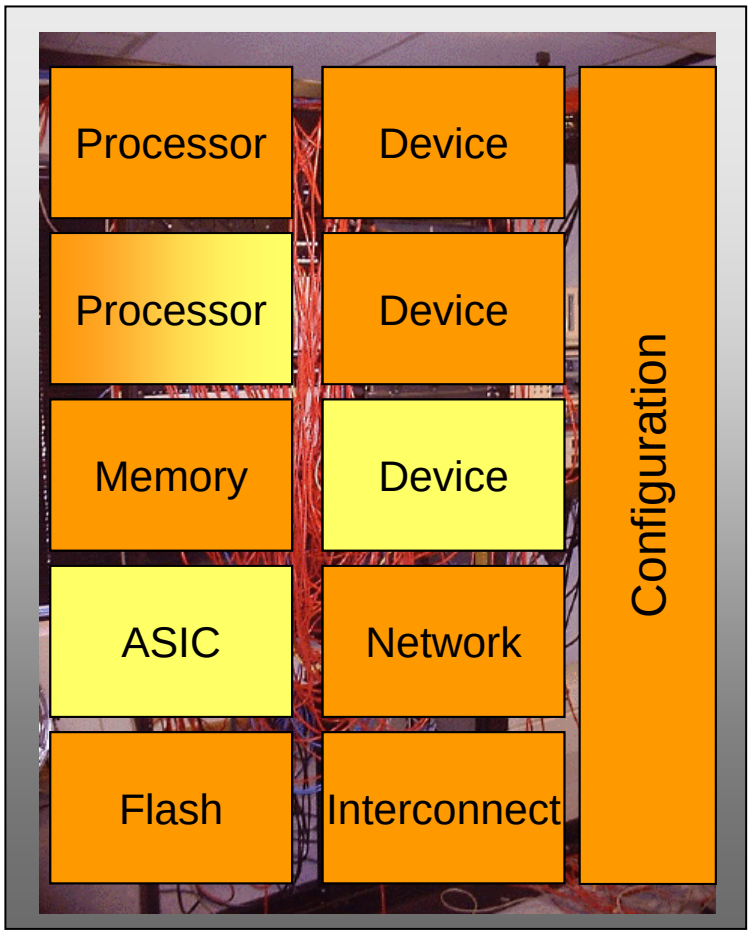
Identical build tools chain

The software can't tell the difference

Runs binaries from real target



Creating a Simulation Model



- Use Simics framework
- Reuse VT components
- Adapt VT components

- Model custom parts
  - DML
  - C, C++
  - Python
- Device modeling by
  - Virtutech
  - Customer
  - Partner
  - Consultant

Simulation in the Real World

"Simulation is the key to advanced microprocessor development, and Simics is by far the most advanced realization of this technology available"

Kevin Collins, Director Global Firmware Development

"There are things you can do in a simulation environment that you can't in the real world"

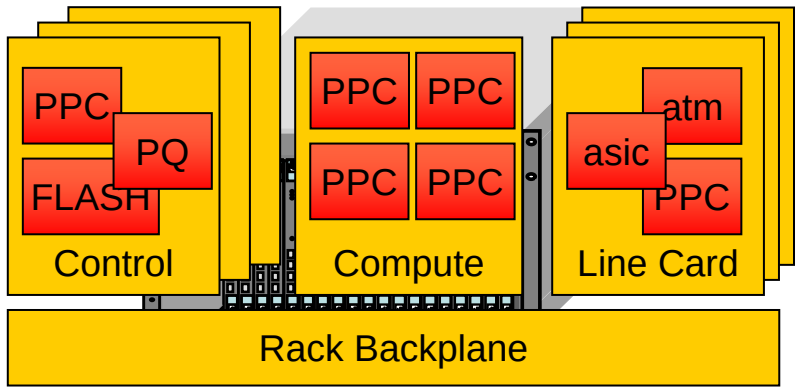
Mike Turnlund, Director Software Tools

"With Simics we were able to locate defects in the flight software within hours versus days using traditional hardware in the loop testing"

Joe Pizzicaroli, Vice President Satellite and Launch Operations

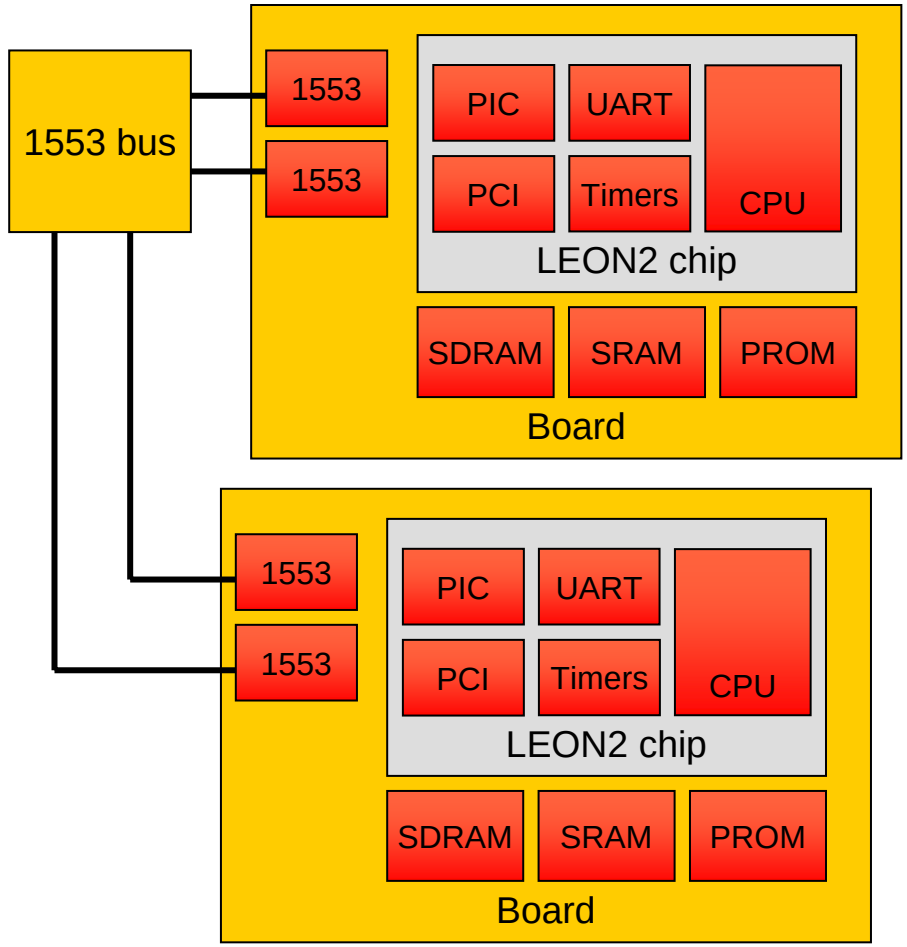
Telecomm

- 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
  - Multiple processor types
    - PowerPC 750, 750fx, 750gx
    - PowerPC 403, 405, 440
    - PowerQUICC II 8260, 8270, 8280
    - TI C64 DSP

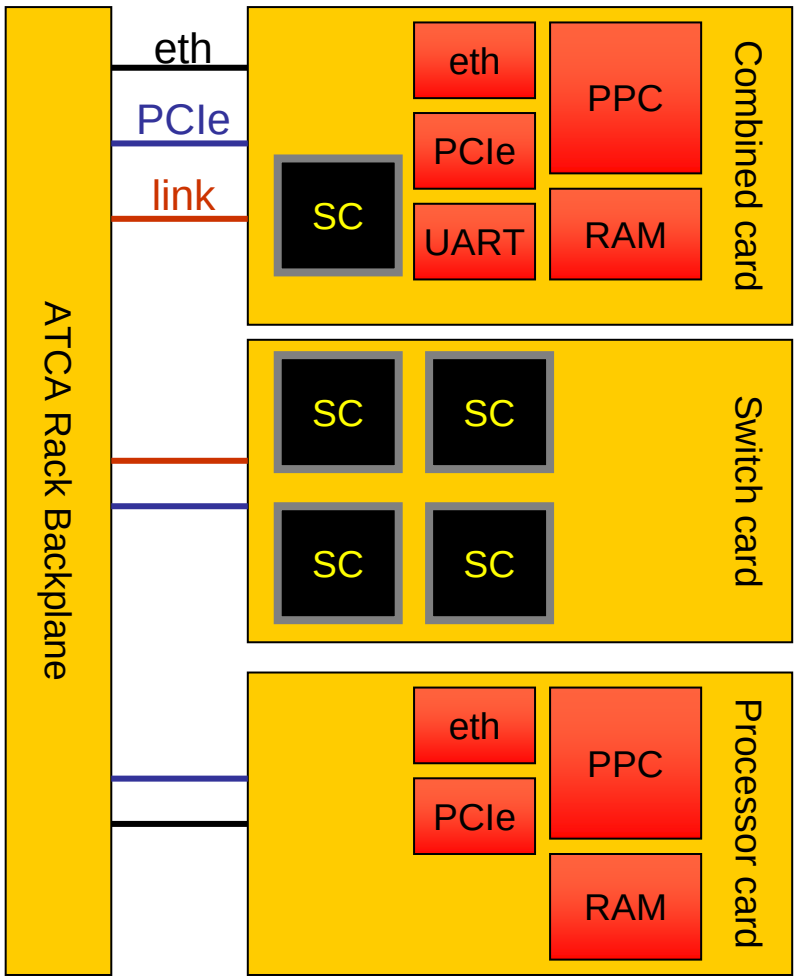


LEON Space Computer Board

- LEON2 Processor
  - Core complex
  - Memory controller
  - Serial
  - PCI
  - Interrupts
  - Clock
- Mil-Std 1553 bus
- Multiple 1553 controllers
- Multiple cards
- RTEMS OS



Virtual Reference Kit for Switch Chip



- Standard components from VT
  - ATCA rack
  - PPC 8548
  - PCIe & Ethernet traffic
- Customer work
  - Custom switch chip models
  - Custom backplane link
- Simics integration
  - Wrapping customer models into Simics models
- Features
  - Multiple cards, multiple chips
  - Same SW as real platform
  - Months before hardware
  - Shipped to subcontractors and OEMs

Questions?