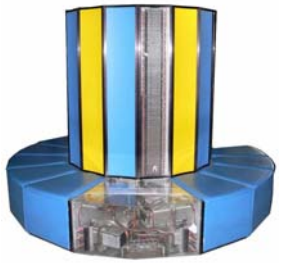
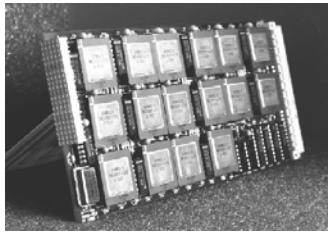
Debugging Multicore Software Issues using Virtual Hardware

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

Multiprocessing is the Future

- No need to further elaborate ☺
- Note that I make no real distinction between multicore and discrete multiprocessor systems From a software perspective, they are the same
  - The difference is in the details and timing, not essentials
- I will mainly address multicore from the embedded software perspective
  - Multicore is subjecting embedded software to a brutal learning-curve that nobody really expected



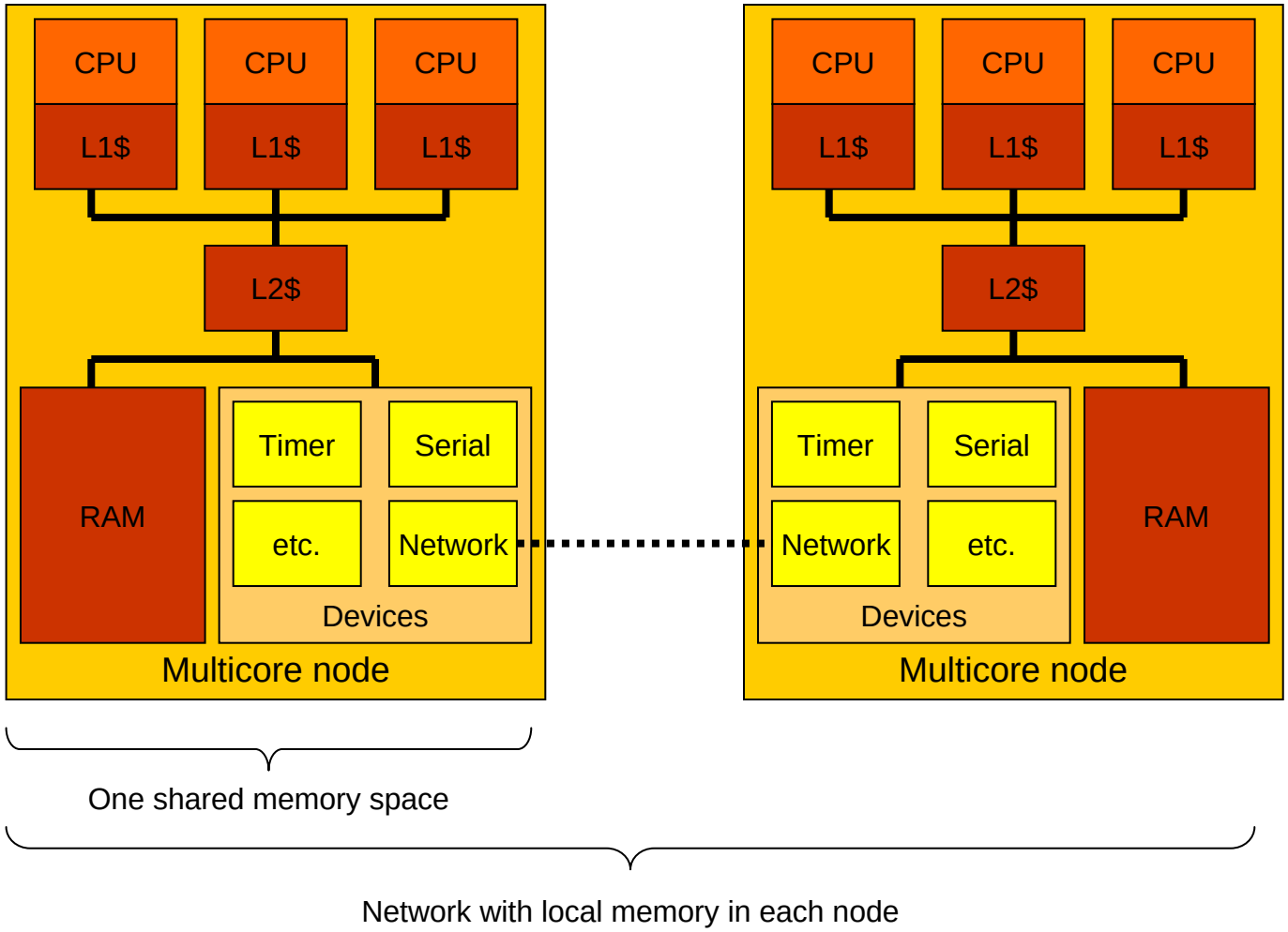



Embedded Multiprocessing is Here

- Multiprocessor and multicore systems are the future
- Massive launches of large-scale multicores in the embedded space in recent years
- Some current examples:

Vendor	Chip	#Cores	Arch	AMP	SMP
ARM	ARM11 MPCore	4	ARMv6	X	X
Cavium	Octeon CN38	16	MIPS64		X
Freescale	MPC8572E	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	ARM,C55,IVA	X	

Future Embedded Systems Template



It’s All About Software

After all, the sole reason computer hardware exists is in order to run software

virtutech

Electronics Is Software

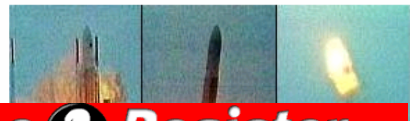
Electronics is software. Shipping a system is largely about identifying and removing defects from the software and keeping them from creeping back in as the product evolves.

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



Los Angeles Times latimes.com

2:43 PM PDT, October 13, 2005

Software Glitch Trig

By James F. Peltz, Times Staff Write

In what's believed to be the firs
that it is notifying about 75,00
car to stall or shut down.

The voluntary recall dented th
as drivers sought better fuel e

ADVERTISEMENT

Science

Lockheed's F-22 Raptor Gets Zapped by International Date Line

Brandon Hill (Blog) - February 26, 2007 10:28 AM

Print E-mail del.icio.us Digg 154 comment(s) - last by seaman.. on Mar 6 at 6:47 PM

Six Lockheed F-22 Raptors have Y2K-esque glitch of their own over the Pacific

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, which means it has been cleared for future flights, the BBC reports.

virtutech

Software gets Even Harder on MP

- Parallelism required to gain performance
  - Parallel hardware is “easy” to design
  - Parallel software is **hard** to write
- Most existing software assumes single-processor
  - Shared-memory multiprocessors used to be very rare
  - Multitasking != multiprocessor-ready
  - Multiprocessors expose latent problems in code
  - Might break in new “interesting” ways on multiprocessor
  - 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
  - Many writers note that this is hard for the human brain
    - Hardest in threads + shared-memory model where every action can influence every other action

virtutech

(Embedded) Software Reality Today

- Programmers used to single-threaded programs
- Legacy code in C, C++, Java, Ada, assembler, Plex
  - Essentially sequential languages
  - Very little in concurrent languages like Erlang
- Fine-grained parallelism added to sequential code
  - OpenMP, pthreads, OS threads, MPI, special C variants, Java threads, Ada concurrency, ...
- Debuggers designed for single processors
  - Or multiple instances of single processors
- If we programmed using better languages, libraries, and tools, many problems would go away. You wish.

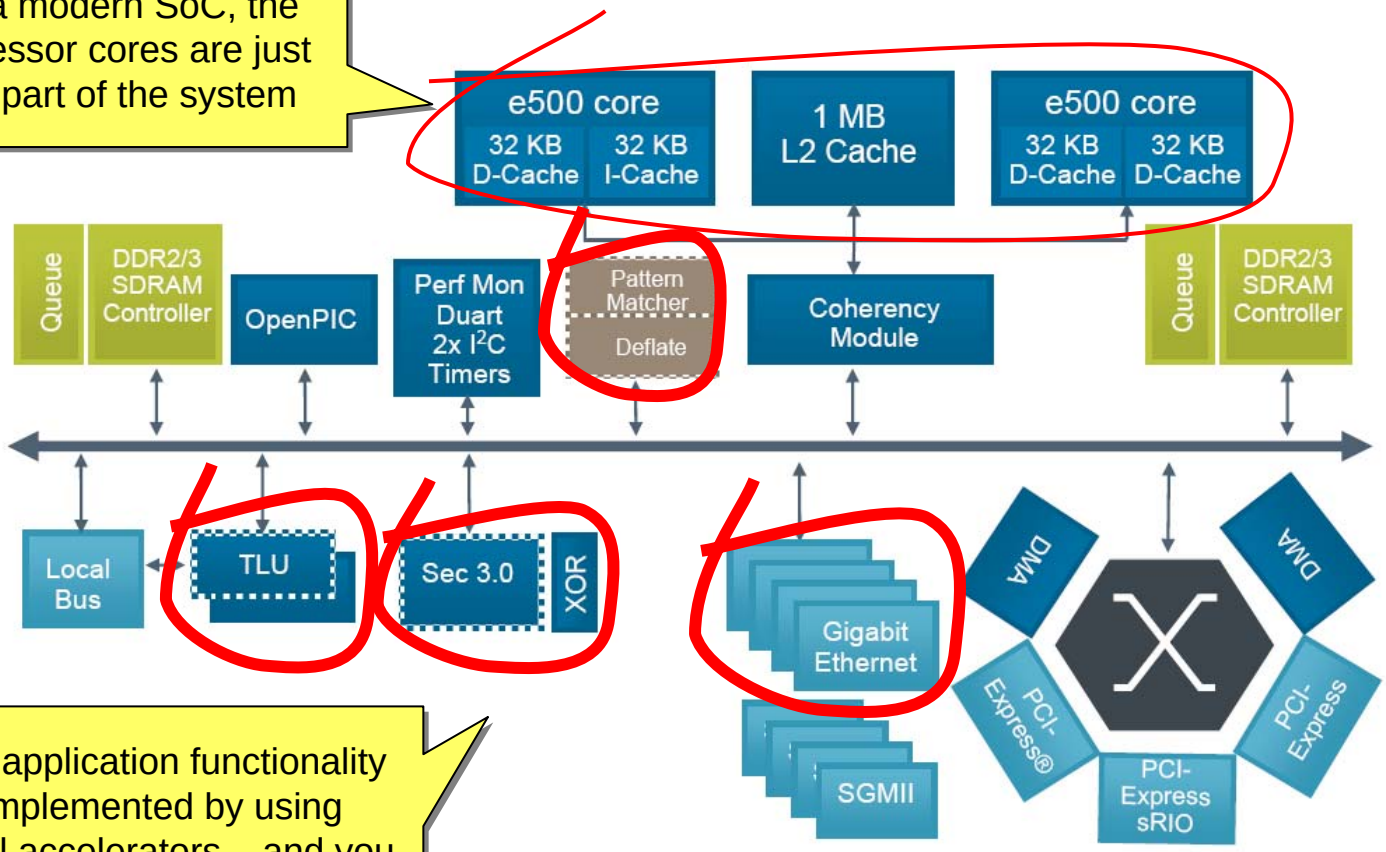
virtutech

(Embedded) Multiprocessors & Debug

- Limited visibility into hardware
  - Single low-speed debug port, multiple processors
  - High speed, concurrent execution
  - Caches and multicore designs hide memory traffic
- Timing-sensitive chaotic behavior
  - Small changes in timing alters system behavior radically
  - Hardware variations impact software behavior
- Lack of determinism
  - 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

And not Just CPU-focused Code

On a modern SoC, the processor cores are just one part of the system

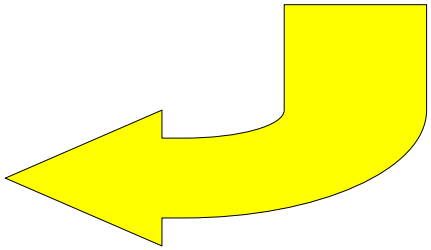


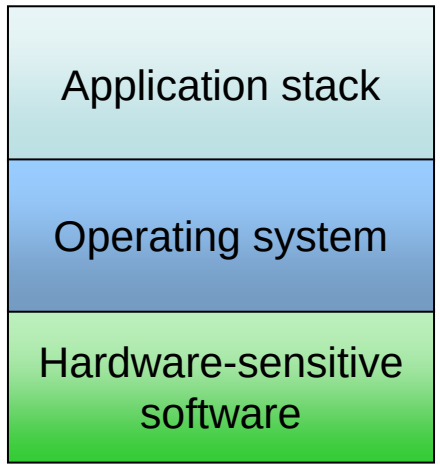
Much application functionality is implemented by using special accelerators... and you need to debug their interaction with the processors & software

Virtual Hardware for Software Development

Traditional Software Development

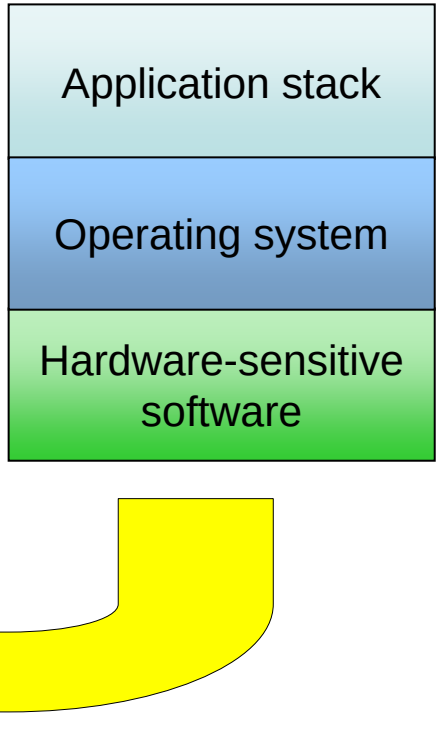
- Software developers create a production binary
- Production binary runs on the physical hardware





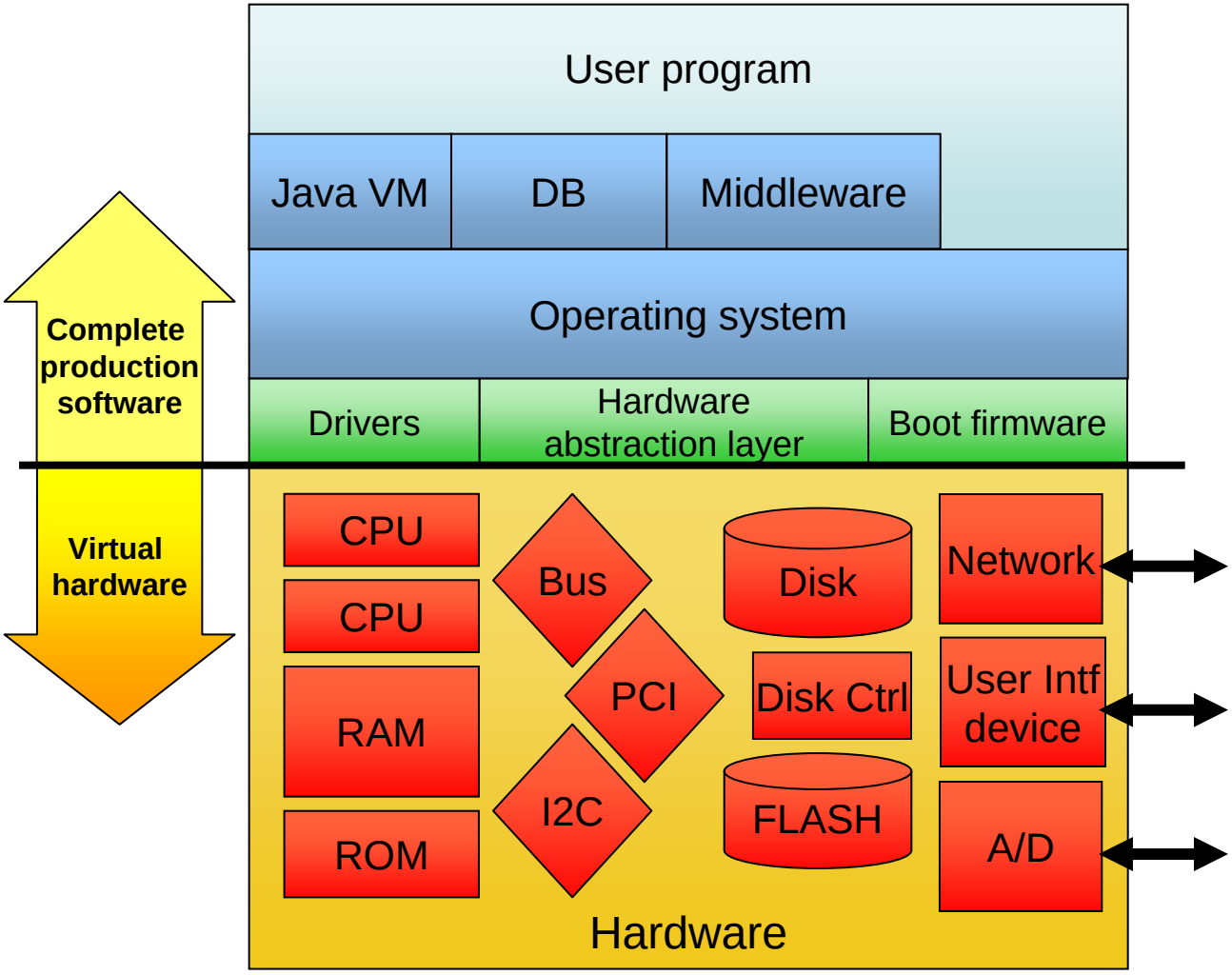
Virtualized Software Development

- Same binary runs inside virtualized software development environment
  - On a virtual model of the hardware



Basic Technology Premise

- The software can't tell the difference
- Identical build tools chain
- Runs binaries from real target
- Replacement for physical hardware



virtutech

Why do we use Virtual Hardware?

Business Reasons

- Reduce software risk
- Decouple hardware and software development
- Shorten time-to-market
- Increase quality
- Reduce blockages due to very hard bugs
- Availability & Flexibility
  - Engineering workstation can be “any” system
  - Easy to change the system
  - Easy to distribute and supply to engineers
  - Infinite supply of test hardware

Engineering Reasons

- Deterministic
- Virtual time
  - Precisely synchronized
  - Stopped at any point
- Checkpoint & restore
- Reverse execution
- Configurable
- Control
  - Any variable or property can be changed
  - Controlled experiments, no real-world randomness
- Inspection power
  - Any state or variable
- No debug bandwidth limit

Virtual Hardware and Multiprocessor Software Debugging

virtutech

Three Steps of Debugging

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

Virtual hardware helps with all three steps

Provoking Errors

- Virtualization provides unparalleled **control** over system configuration and execution
- Replicate failing tests from production units
- Vary system hardware configuration
  - Like testing on a variety of real-world machines
- Vary system software configuration
  - Easy to test different software loads on different boards
- Systematically search for error cases
  - Controllably force system into likely error conditions
  - Make a processor slower or stop it entirely
  - Increase communication latencies
  - Slow down individual processors to increase perceived load
  - Inject hardware faults in the system

virtutech

Reproducing Errors

- Virtual hardware state can be **checkpointed**
- Virtual hardware execution is **deterministic**
  - Simulation engine imposes a well-defined sequential semantic to the parallel execution of the target machine
  - All machine-internal events have deterministic time
  - Input from real world recorded & replayed
  - ... any variability is explicitly programmed and controlled
- An error is provoked in simulation can be reproduced
  - Reset back to initial state (or restore a checkpoint)
  - Rerun the test case that ended in error
  - Same error state results
  - ...any number of times
  - ...on any machine running the simulator
  - ...from a checkpoint distributed to multiple developers

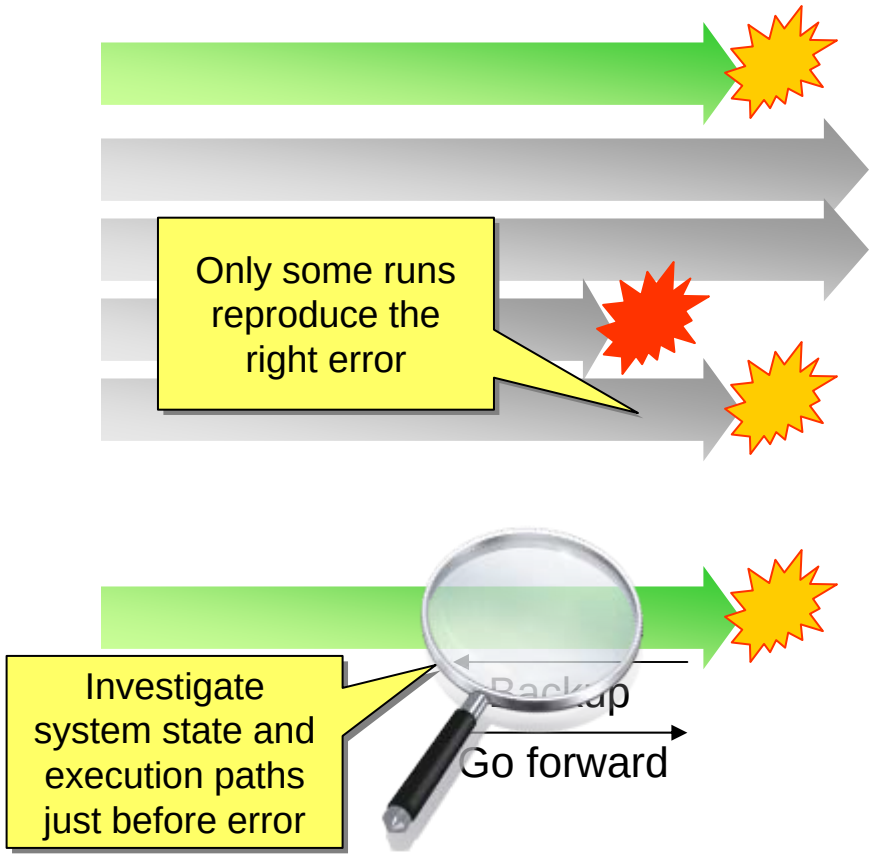
virtutech

Locating and Fixing Errors

- Global system stop
  - Virtual time does not progress anywhere in the system
  - Single-step multiprocessor coordination code, interrupt handlers, and other timing-sensitive code
  - Inspect state at your leisure
- Inspection access all parts of the system state
  - Memory, registers, device states, interrupts, MMU, ...
- Unlimited breakpoints and watchpoints
- No probe effect from instrumentation or inspection
  - Trace and observe any state without altering it
- No timing disturbance from breakpoints
- Heisenbugs cannot occur
  - In a deterministic virtual system, all bugs are Bohrbugs

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
  - Back out of crashed and utterly broken system state
  - Very hard to do for multiple processors using trace on physical hardware



Some Examples

Of multiprocessing bugs found
in virtual environments

virtutech

Divide-by-zero in OS Kernel

- Operating-system kernel crash in virtual model
  - Divide-by-zero right in the kernel
  - Algorithm to determine and compensate for clock skew
  - Division by difference in time between two processors
- Virtual model had zero clock skew = provoked error
  - Could have happened on a real system
  - Just not very likely
  - Typical rare problem in the field
  - Essentially testing a rare corner case in system state

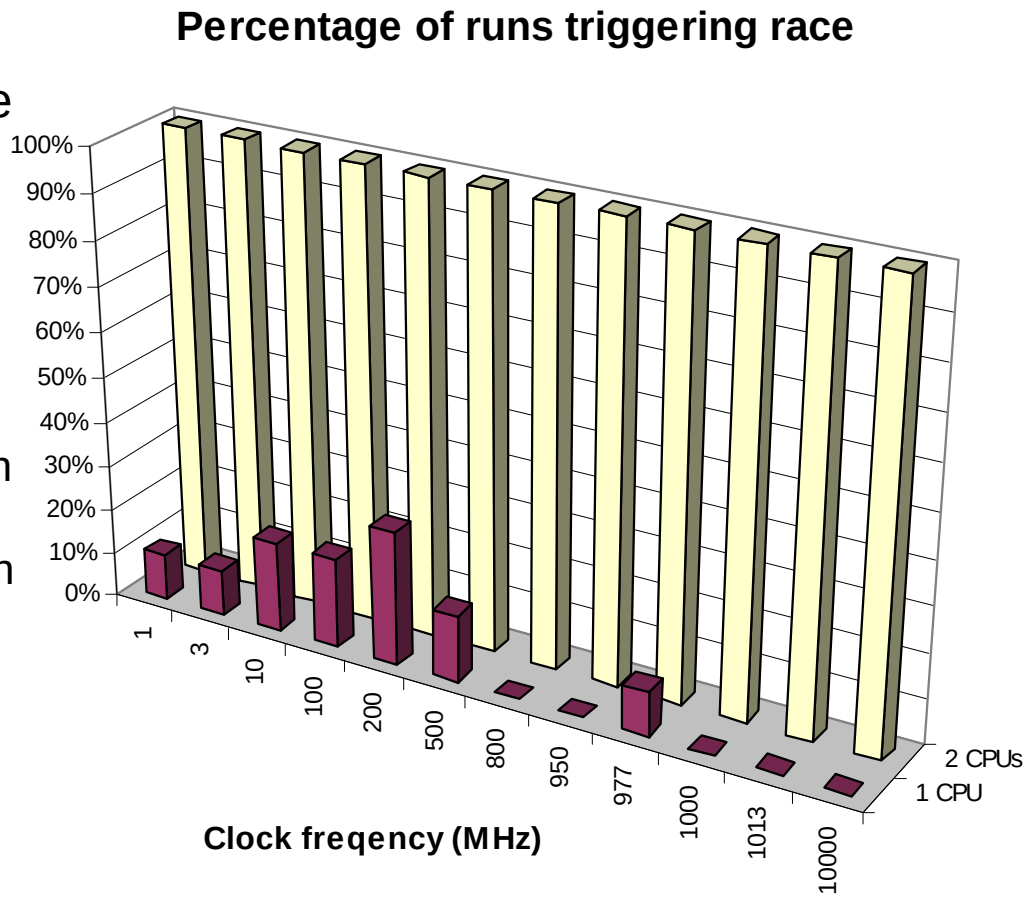
virtutech

Race Condition in Serial Driver

- The problem:
  - Dual-core MPC8641D machine
  - Changed clock frequency from 800 to 833 Mhz
  - OS froze on startup – quite unexpectedly
- Investigation:
  - Only happened at 832.9 to 833.3 MHz
  - Determinism: 100% reproduction of error trivial
  - Time control: single-step code feasible
  - Insight: look at complete system state, log interrupts, check the call stack at the point of the freeze, check lock state
- What we found:
  - An interrupt service routine attempted to take a lock, before re-enabling interrupts. In the case that froze, the lock was already taken when the service routine was entered, and with no interrupts enabled there was no way for it to be released.

Multipro vs Multitasking

- Simulated single-CPU and dual-CPU
- Test program with embarrassingly bad race run 20 times on a range of clock frequencies
- 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
  - **Multitasking hides errors quite well**



virtutech

The Disk Corruption

- Distributed fault-tolerant file system got corrupted
  - Not shared-memory machine, all boards single-processor connected by several networks
  - Intermittent error
  - Error seen as a composite state across multiple disks
  - **Months** spent chasing it on physical hardware
- Simics solution:
  - Reproduce corruption in Simics model of target
  - Pin-point time when it happens, by interval halving
  - Around the critical time, take periodic snapshots of disks
  - Check consistency of disk states in offline scripts
- Result:
  - Found the precise instruction causing the problem
  - Could capture the network traffic pattern causing issue
  - Communicated the complete setup to the file system creator, allowing the root cause to be fixed

Closing Remarks

virtutech

Is Validity an Issue?

- Fidelity of virtual models:
  - Virtual models are never exactly like the real thing
    - In details of timing, and abstracted aspects
  - But they are close enough to provide very useful service
- Any bugs found in simulation are valid bugs
  - Simulation does explore a range of behaviors that are possible in the real system.
  - Note that precise timing simulation is not really possible
    - "Cycle Accuracy" is really "Cycle Approximate"

See Ekblom and Engblom, "Simics: a commercially proven full-system simulation framework", *SESP 2006*, for a deeper discussion

Summary

- Virtual hardware provides an additional tool for embedded software development
- Leverages hardware-software equivalence
- Especially useful for tough bugs
  - Parallelism
  - Hardware-software interaction
  - ”Heisenbugs”
- Price is execution and speed and model creation

Questions?