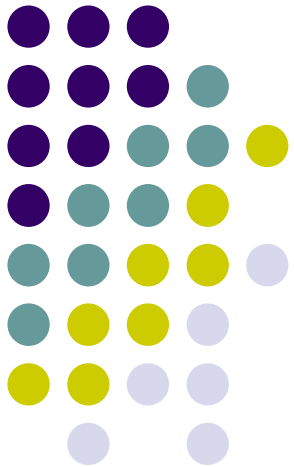
Fast And Very Accurate Computer System Simulation Via FPGA-Based Acceleration

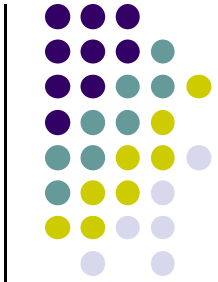
Derek Chiou,
Dam Sunwoo, Nikhil Patil, Hari Angepat,
Joonsoo Kim, Bill Reinhart, D. Eric Johnson,
and Gene Wu

University of Texas at Austin
Electrical and Computer Engineering



Supported in part by DOE, NSF, SRC,
Bluespec, Intel, Xilinx, IBM, and Freescale

First, Some Terminology



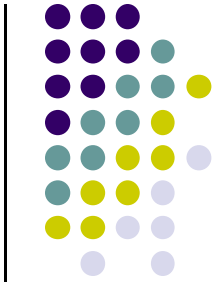
- Host
  - The system on which a simulator runs
    - Dell 390, a 1.8GHz Core 2 Duo, 4GB RAM, Windows XP
    - A Xilinx XUP board
    - A Nallatech ACP system
- Target
  - The system being modeled
    - Alpha 21264 processor
    - Dell 390, a 1.8GHz Core 2 Duo, 4GB RAM, Windows XP

Simulation

Computers enable modern simulation
As computers get faster, so do simulators of the real world
Real world does not increase in complexity

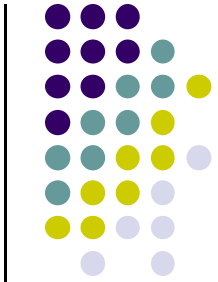
Simulators are fantastically capable, and getting better, quickly

Except For Simulating Computers



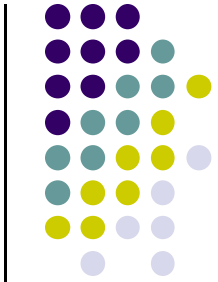
- Accurate simulators are slow
  - Accurate simulators between 1KIPS-10KIPS
    - Source: Emer, Intel / Barnes, AMD
    - 2min of execution takes years
- Getting slower
  - Unlike physical world, computers grow in complexity faster than they get faster
    - More complex cores, more cores, more features, etc.
  - Slowing down by a factor of 2, relative to target, per year (Murkherjee, Intel)

Why Faster Simulators?



- Why not run many short simulations in parallel?
  - Cannot run full, unmodified, unbenchmarked, software
- Better architect computer systems ***before*** they are built
  - Speed enables longer , heavier evaluation of current software on future hardware
    - what architectural mechanisms improve database/game speeds?
- Facilitate ***during*** co-design of hardware and software
  - Intel and Microsoft making a system that works well together
    - Need to be able to execute sufficient cycles to run Microsoft software
- Tune software for correctness, performance and power ***after*** real system exists
  - Provide full visibility at useable speeds
  - Difficult/impossible to achieve on a real system

Simulator Parallelization Required

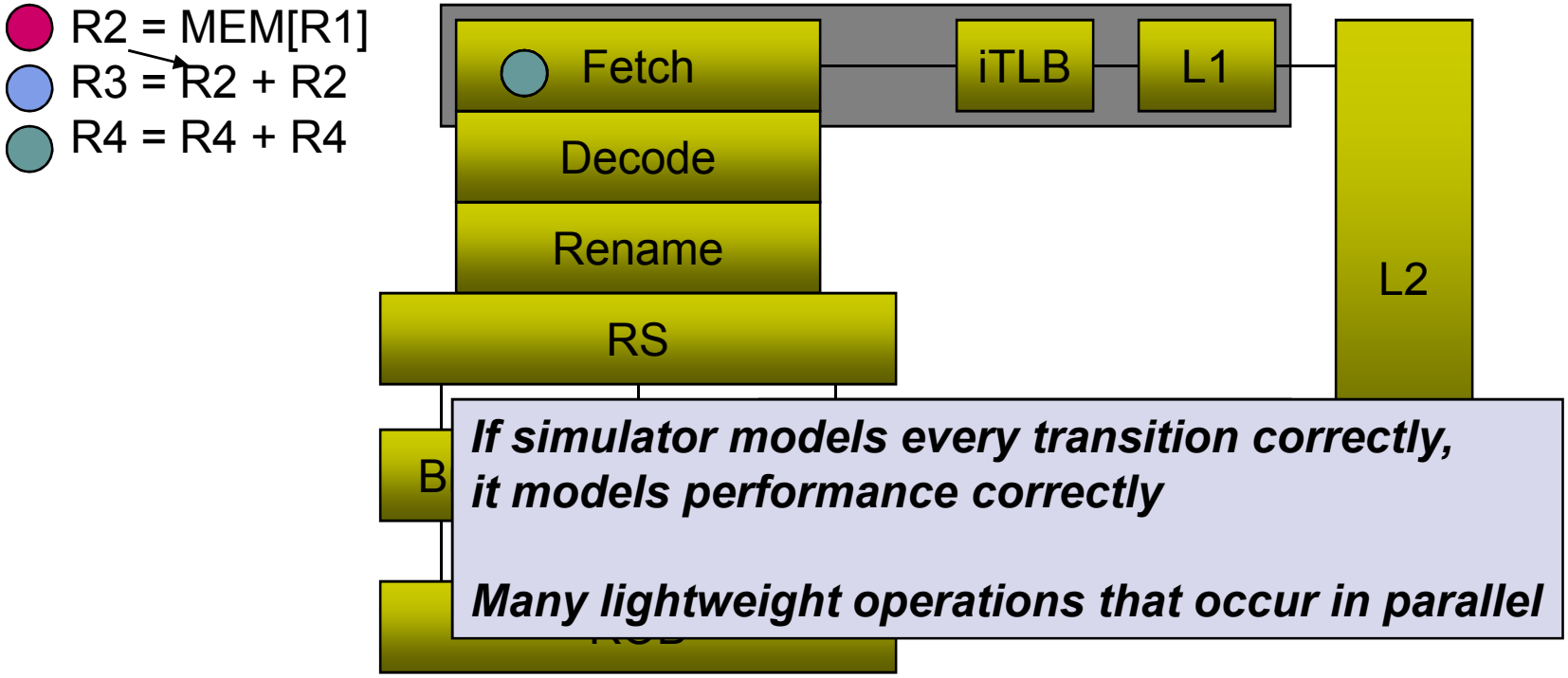


- Improve squential target performance
  - 1KIPS-10KIPS is too slow to simulate virtualization, full software stacks, etc.
- Scalable performance for multicore targets

- Pure software parallelization insufficient
  - To improve performance to 10MIPS, need 1K+ processors

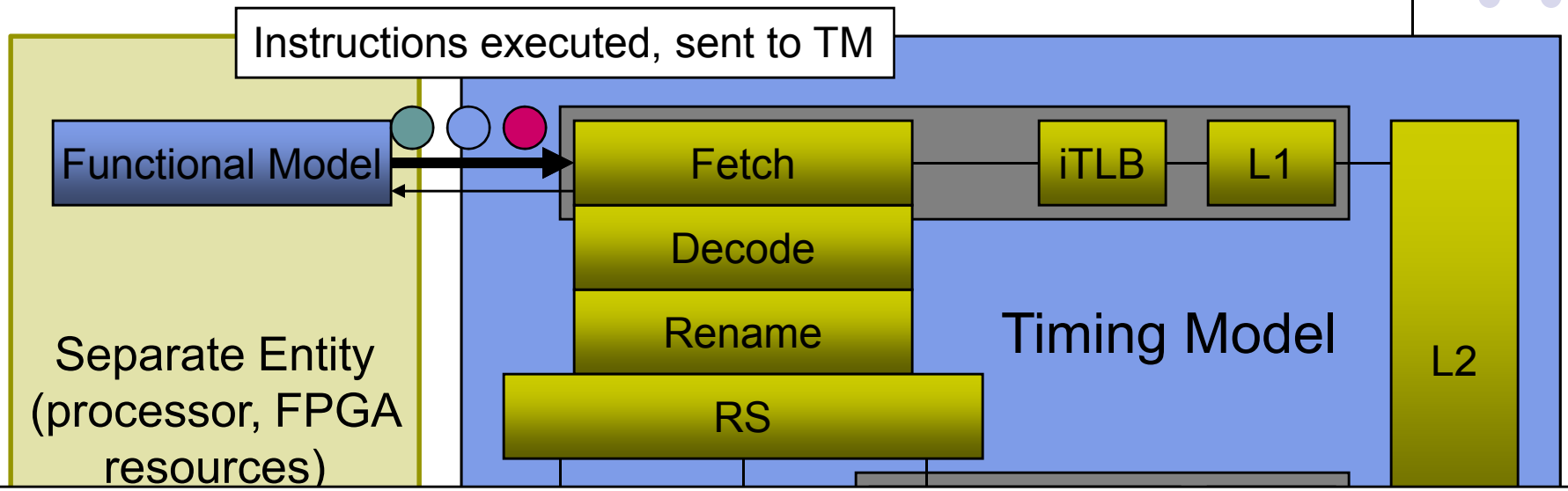
- Hardware is essential
  - Need a new ***simulator architecture***

Cycle-Accurate Simulators



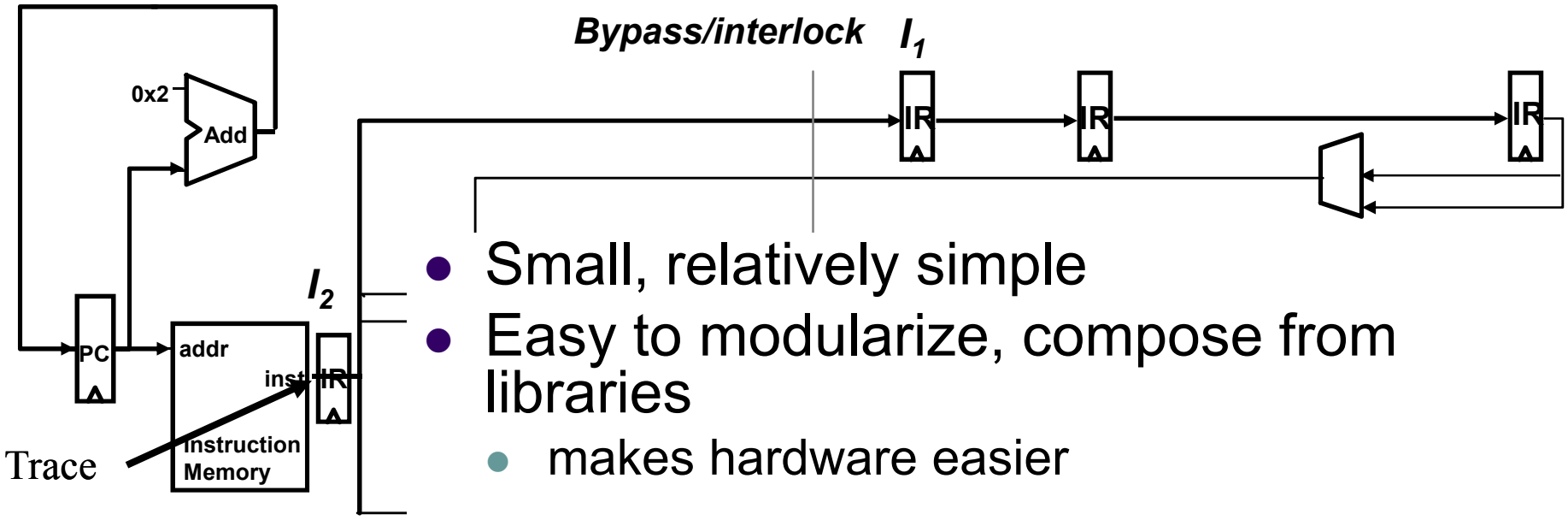
FPGA-Accelerated Simulation Technologies (FAST) Simulator Architecture

MICRO 2007



- Functional model speculates at ***simulator level***
  - Roll back required if functional path != timing path
    - Branch misprediction likely to require 2 rollbacks
  - Functional model runs ahead, improves parallelism
- Parallel slowdown due to communication?
  - FM runs ahead, speculatively, round-trip communication infrequent
  - **Round-trip communication only when functional path!=timing path**
- The better the target micro-architecture, the faster the simulator

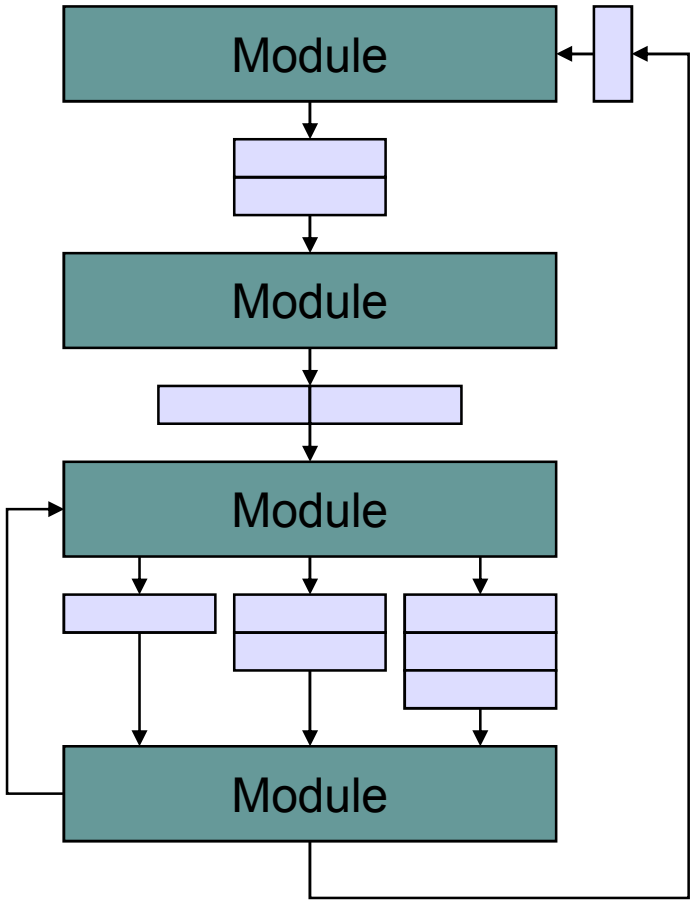
Timing Models are Simple Compared to Implementation



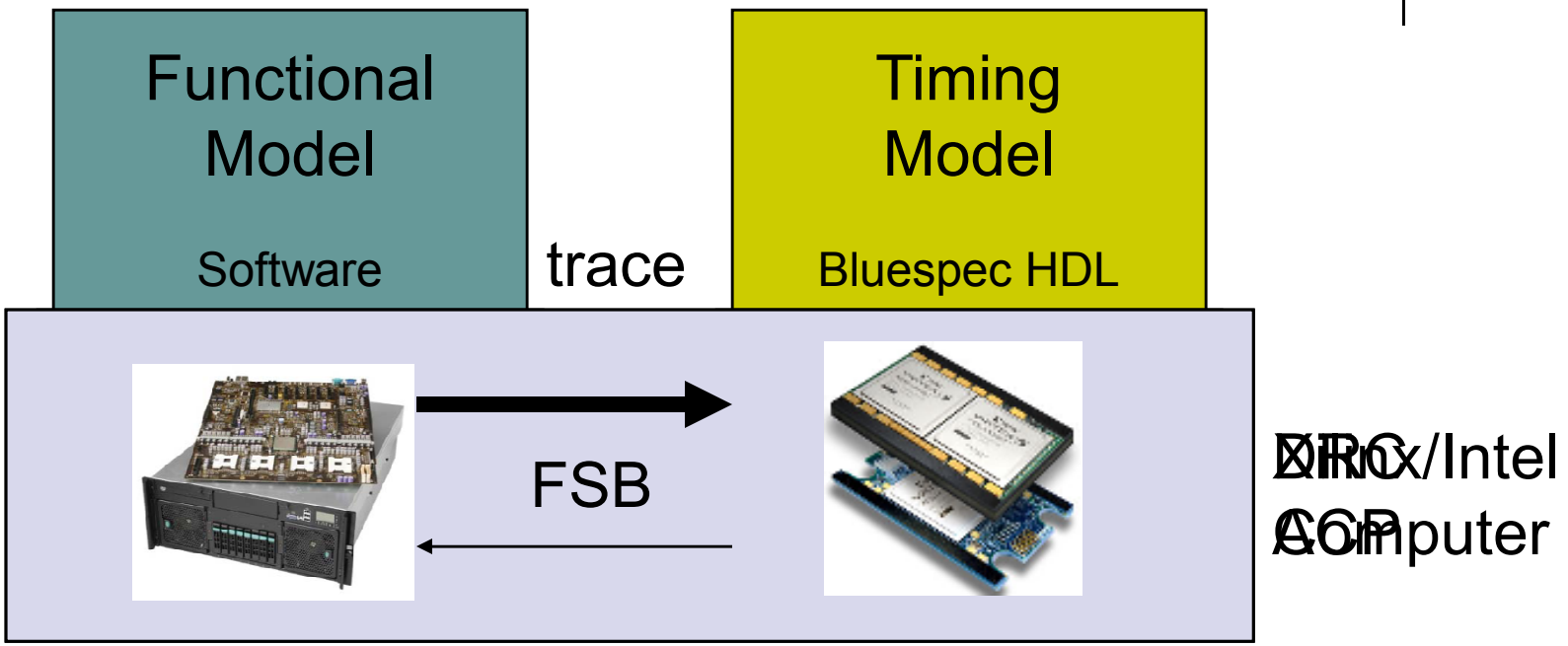
- Small, relatively simple
- Easy to modularize, compose from libraries
  - makes hardware easier
- Can do things like cache studies
  - Have all functional information

Modular Timing Models Reduces Development Effort

- Currently written in Bluespec
- Modules model behavior
- Connectors
  - Abstract performance information from modules
    - Latency, throughput, capacity
    - Easier to vary parameters
  - Simplifies modules
  - Provide a stats & trace gathering platform
    - Search for performance issues
- ***Multiple host cycles per target cycle***
  - ***16 reads of dual-ported BRAM to model 32-way associative cache***

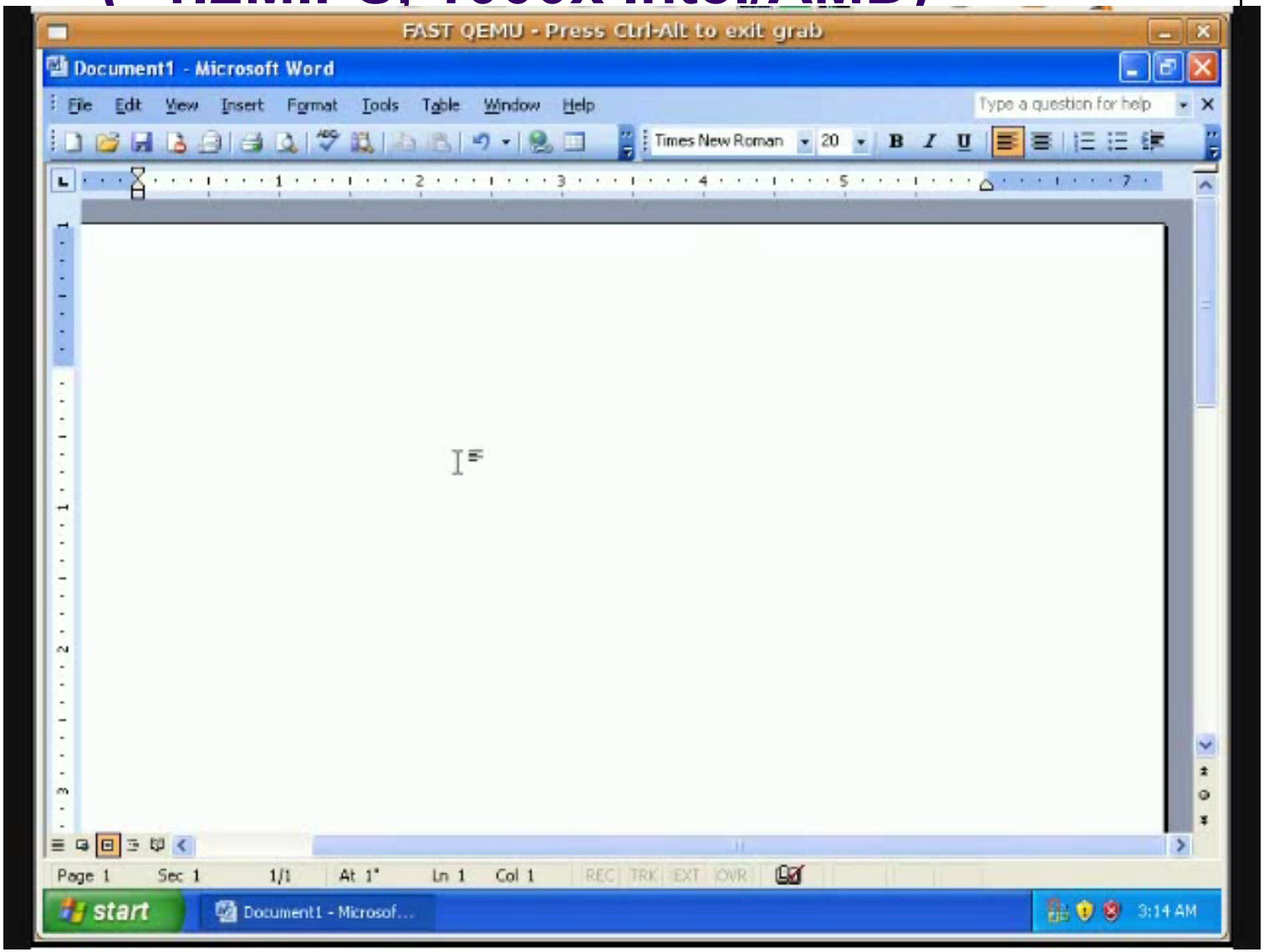


FAST Prototype Overview

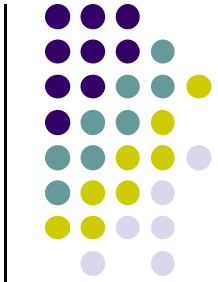


- Software (QEMU modified with trace/rollback) functional model
  - Eventually hardware functional model, but software sim exists
- FPGA-based timing model written in Bluespec
  - Complex OoO micro-architecture fits in a single FPGA

Old FAST Prototype in Real Time on DRC (~1.2MIPS, 1000x Intel/AMD)



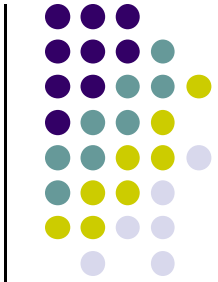
FAST Is …



- **Not** Prototyping
  - Does not require RTL
    - E.g., Palladium, Quickturn, Centaur/ARM implement full RTL
- **Not** RTL Acceleration
- **Not** software simulation
  - Simplescalar, Asim

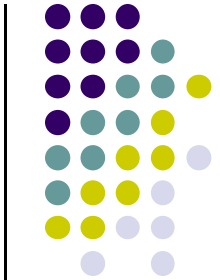
- A ***simulator*** accelerated by FPGAs
- The goals of FAST are to be as fast or faster than prototypes, much more cost effective (1 FPGA models a processor), early stage, full visibility

Some Similar Projects



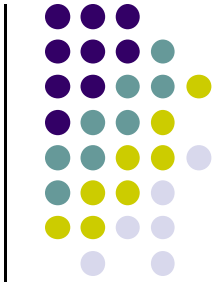
- Protoflex (Hoe, CMU)
  - Goal: Improve ***functional*** simulation speeds
  - FPGA implementation of most common target instructions
  - Transplant to software to handle complex instructions
- HASim (Emer, Intel/MIT)
  - Goal: same as FAST
  - Functional model in FPGA as well
  - ***Simulator layer is not speculative***
    - Timing model controls when functional model does everything
- RAMP-Gold (Asanovic, Berkeley)
  - Goal: Simulating large numbers of cores reasonably accurately
  - Highly multithreaded functional model and timing model, both in FPGA
- RAMP-Red (Stanford) and RAMP-Blue (Berkeley) were both prototypes

Why FPGAs?



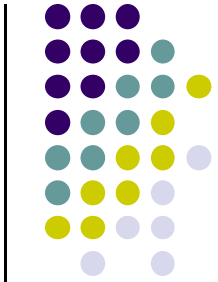
- FPGAs are almost ideal to model many, light interactions in parallel
  - Interactions can be very different
  - Can't do this with GPU (data parallel)
  - Multicore host is not parallel enough
    - Number of target cores grows too
- Want flexiblity
  - Rules out ASICs
- Need programmable hardware
  - Cannot do this with anything but FPGAs!

What Could Be Improved?



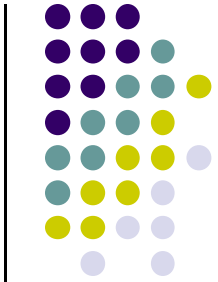
- FPGA visibility
  - Why can't we single step an FPGA and read out all of the state? (gdb for FPGAs)
    - FPGA-accelerated simulator could look identical to software
    - Concern: currently need to stop much of FPGA to read out state
  - Remove external interfaces (eg. memory) from FPGA fabric, using hard macros?
- High speed, easy-to-use exporting of state
  - Statistics, logging
- Tool speed
  - Compile speeds comparable to software
  - 10MHz clock frequency in a Virtex5 is fine when debugging
  - Currently target 100MHz-133MHz host speed for production to reduce/eliminate timing tuning
- Tool cost
  - Web-based toolchain available for free IF source code made public-domain?
    - Even safer is tool output is run on controlled machines
    - Simulator-as-a-service
  - Pay-for-proprietary
- Tight coupling with processors + memory
  - Nallatech/Xilinx ACP is a great step in the right direction
  - Washington/Xilinx CHiMPS?

Some Key Open Questions



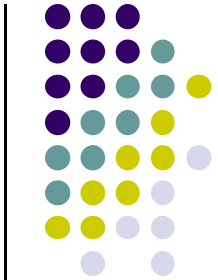
- Best way to write timing models
  - Structurally, functionally
- Make writing timing models easier?
  - C/C++2FPGA?
    - Software timing model
- FAST2Imp
- Imp2FAST

Potential Market



- Processor design houses
  - E.g., Intel, AMD, IBM, Freescale, ARM, …
  - Intel has tens of thousands of machines running simulations
- High-end software developers
  - Performance/power important
  - E.g., Microsoft, Oracle, etc.
- System companies
  - E.g., Nokia, Cisco, etc.

Conclusions



- FPGA-Accelerated/Based simulation is significantly faster than software
  - 100-10,000+ times faster
- Fast simulators are useful
- FPGA-Accelerated simulation could be a huge market for FPGAs and tools
  - Every high-end software developer may need multiple FPGAs
  - Processor architects may need thousands to tens of thousands