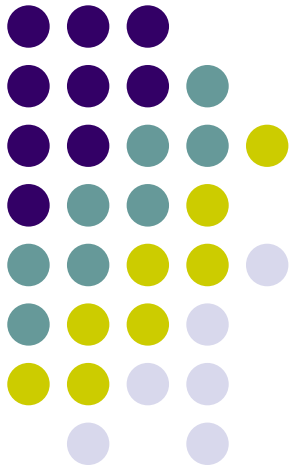
# Splitting Functional and Timing Models: The Role of Bluespec

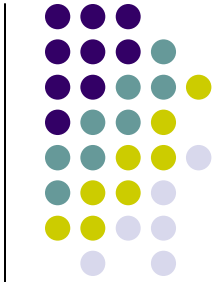
Derek Chiou

University of Texas at Austin

Electrical and Computer Engineering

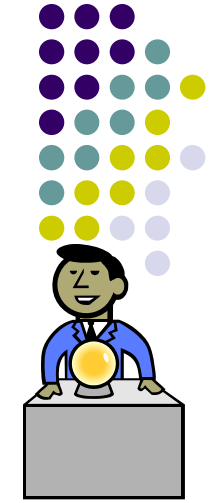


# First, Some Terminology

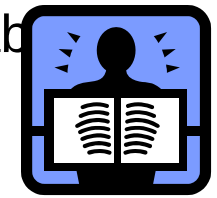


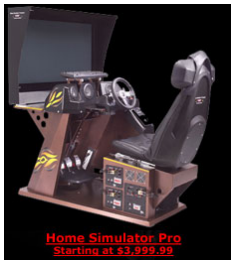
- Host
  - The system on which a simulator runs
  - E.g.,
    - Dell 390 with a single Core 2 Duo running at 1.8GHz with 4GB of RAM
    - A Xilinx XUP board
- Target
  - The system being modeled
  - Eg.,
    - Alpha 21264 processor
    - A Dell 390 with a single Core 2 Duo running at 1.8GHz with 4GB of RAM

# Simulators



- Why?
  - Predict performance metrics of not-yet-available targets
  - Provide better visibility (even after hardware availab
  - Provide early execution platform for software
- Must/Should haves
  - Available before hardware
  - Easy to modify
    - To explore space, make corrections
  - (Significantly ) less effort than the real implementation
  - Should provide better visibility (–g mode)
  - *Be fast enough for software (1MIPS-100MIPS+)*
    - *Unified simulator used by hardware and software*
    - *Find more bugs before hardware*
- *How to simulate in hardware to improve simulation speed*





# Issues of Hardware Simulators

- Cannot be significantly harder than software simulators
- Reuse as much as possible
  - Simplify hardware implies easier to share
  - Reuse software components?

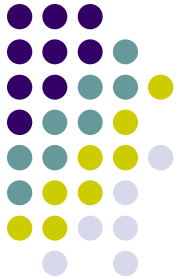
- Folded/Unfolded: how resources are reused
  - Amount that work reuses underlying host hardware structures
    - Multiple host cycles (Michael already mentioned)
  - Trading time for space for implementation time
- Factorization: Smart partitioning
  - Effort is additive, Effect is multiplicative

# Role of Bluespec in Simulation

- Hardware simulators practical only if development effort is reasonable
- Much faster to develop than standard RTL
  - Abstraction level approaching software
  - Easier to partition and assemble
  - Easier to abstract
    - Incorporate less accurate (perhaps unsynthesizable) code

- Bluespec-derived scheduling
  - Fine-grained control often not needed assuming task done in target cycle
- Difficult to simulate things map well onto guarded atomic actions
  - Cache coherency protocols
- Easier to fold/unfold
  - Vectors
  - Loops
- Much easier to parameterize (passed types)
  - Modularity, generality

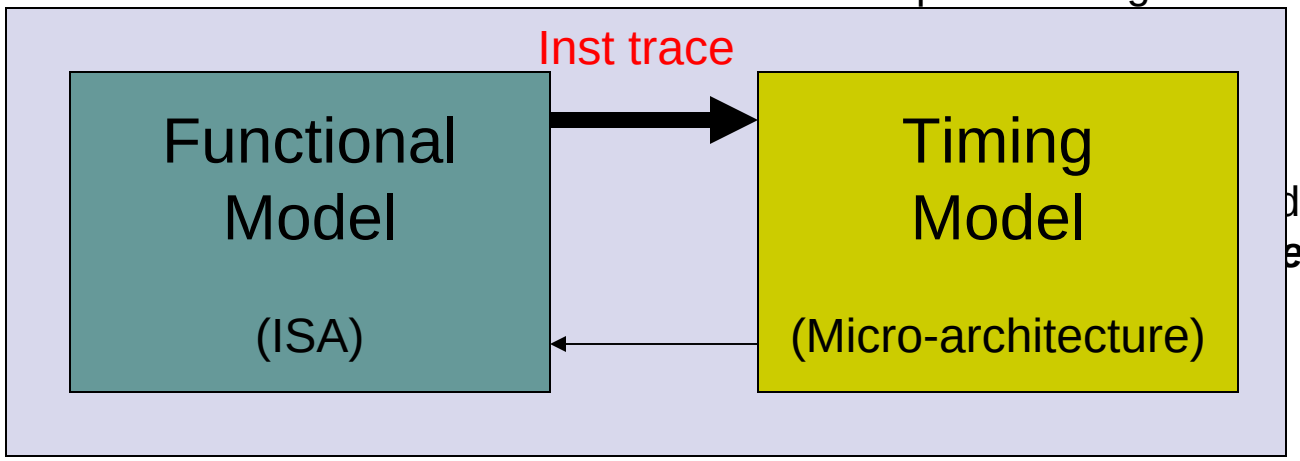
# Our Factorization: Functionality/Timing Boundaries



Instructions
Architectural registers
Peripheral functionality
…..

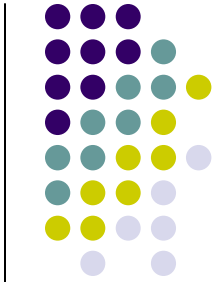
Caches
Arbitration
Pipelining
Associativity
…

- Proven Software Partitioning
  - Asim, FastSim, etc.
- Good factorization
  - Functional simulators exist
  - Timing model becomes very simple
- Promotes reuse
  - Functional/Timing
  - Simplifies timing model



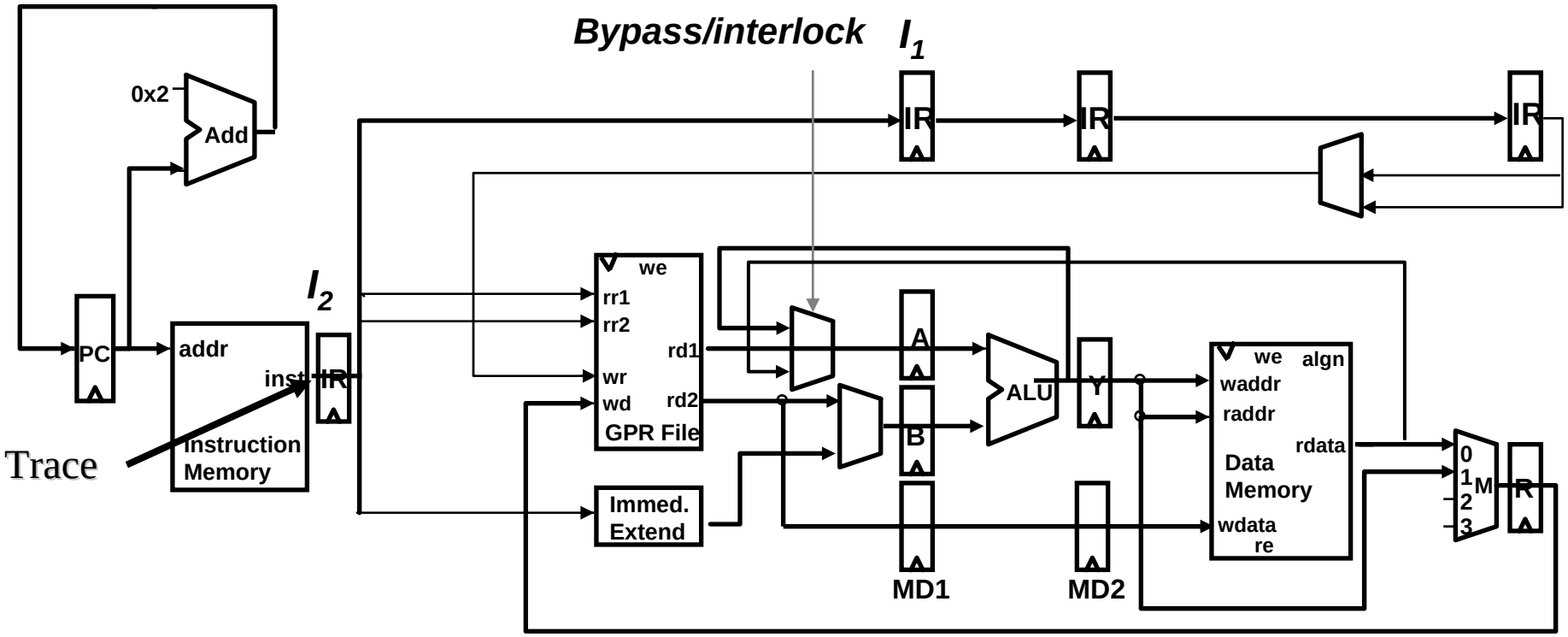

del?
*el?*

# What is in a Trace?

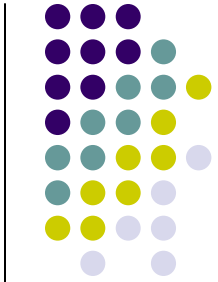


- Conceptually, everything a functional model can produce
  - Flattened opcode, virtual/physical address of instruction, virtual/physical address of data, source registers, destination registers, condition code source and destination registers, exceptions, etc.
- Can be heavily compressed
  - Eg., simulator TLB to avoid physical address

# What is a Timing Model?

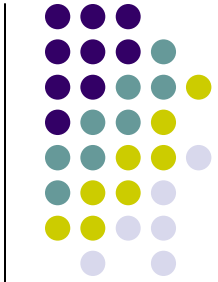


# Timing Models



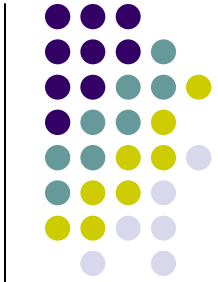
- Sub-modules combined to create larger modules
- Comprised of
  - CAMs, FIFOs, arbiters, memories
  - Caches
  - Branch predictors
  - Fetch, Decode, Rename, Reservation Stations, ALUs, ROB
- Each module (should be) very simple
- Extensive sharing of modules/sub-modules
- Major interfaces use parameterized Bluespec types
- Parameterized folding
  - E.g., our CAM takes a “parallelism” argument that indicates how many RAMs to use, trading space for performance

# Functional Models



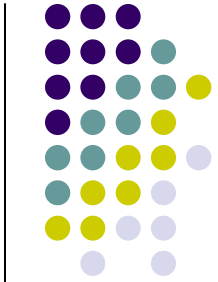
- Software
  - Many full-system functional simulators exist
    - Simics, SimNow, Bochs, QEMU, Mambo, etc.
- Hardware
  - ISS
  - Full superscalar processor
  - How folded

# Real Partitioned Simulators



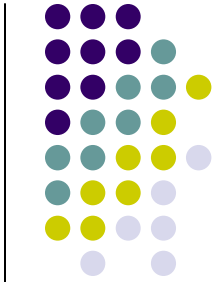
- UT FAST (Chiou et. al.)
  - X86, boots unmodified Linux, almost Windows XP
    - Runs unmodified apps
    - Porting to PowerPC soon
  - Speeds
    - Runs at 1.2MIPS today
      - Intel & AMDs cycle-accurate simulators run between 1KHz-10KHz
    - 5-10MIPS soon (next 6 months), long term goal of 10MIPS-100MIPS
  - Heavily-modified QEMU functional model
  - Abstract OOO superscalar timing model, RTL-level cycle accurate
    - Written completely in Bluespec
- Intel HASim (Emer, et. al.)
  - SMIPS functional model, R10000 timing model
  - Difference in how functional model executes
  - We have developed methods to share functional and timing models between FAST and HASim

# FAST: Speculative Simulation



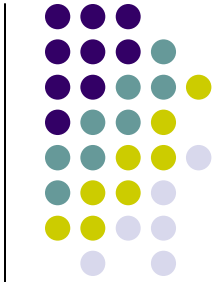
- FAST uses full-system software ISS as functional model
  - Instructions each executed to completion before going to next instruction
- Thus, FAST simulators are speculative!
  - Speculate functional path == target path
  - Timing model detects functional path != target path
    - Forces functional model down wrong path OR
    - Returns functional model to functional path
- Speculation reduces need for round-trip communication
  - *Makes FAST latency tolerant*
- Timing model determines target behavior
  - Functional model completely reusable

# How Could Bluespec Help More?



- Pack/unpack software functions (nag, nag, nag)
  - Would provide interface between hardware and software
- Virtual to physical channel mapping
  - Additional folding capabilities
  - Would automatically provide complex interfaces
  - Hardware-to-hardware and software-to-hardware
  - Within a chip and across chip boundaries
  - Reduces/eliminates that mapping process
  - Generalizes the models

# A Plug



- First Annual University of Texas at Austin ECE Computer Architecture and Circuit Design Open House
  - August 28th, all day
  - Will showcase architecture and circuit design research being done in UT's ECE department
- Everyone welcome
  - A great way to learn more about UT research