UNified Universal Microprocessor Framework

Nirav Dave, Michael Pellauer, & Arvind
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Lab

E Pluribus Unum
Ex Uno Plures
Wouldn’t it be great if...

Architects (especially budding architects) could:

- Easily Explore several different micro-architectures
  - Get a feel for the performance/area/power tradeoffs

- Try out cool design alternatives
  - New branch predictor, Load-Store buffer scheme, or New cache-coherence protocol

- Have a parameterized design
  - By superscalar width, instruction latencies, etc.

- Easily convince others of their results
One Solution: Simulators

- Simulators written in C/C++/SystemC
  - Fast and extensible \textit{but}
  - Provide too much temptation to cheat (i.e., lower fidelity) to increase the speed of simulation
  - Make it too easy to inadvertently leave out important implementation details \(\Rightarrow\) have difficulty convincing others of the results
    \(\text{(also cannot mapped to FPGAs...)}\)

- RTL simulators written in HDLs (e.g., Verilog)
  - Tied to reality - can be turned into real hardware or simulated on FPGAs \textit{but}
  - Software simulation of large designs \textbf{painfully} slow
  - Error prone and very hard to change
  - Requires greater specification of low-level design
Bluespec: Simplifying RTL Generation

- RTL can be synthesized from high-level architectural descriptions

- Synthesized RTL can be simulated or further mapped onto FPGAs or into ASICs
  - Gives confidence in completeness of design
  - Provides an opportunity to study timing, area and power characteristics

See for example,
- Itanium study by Wunderlich & Hoe (ICCD 04)
- IP Lookup by Arvind, Dave, Nikhil, Rosenband (ICCAD 04)
Bluespec Tool Flow

Bluespec SystemVerilog source → Bluespec Compiler → C

Bluespec C sim ↔ Cycle Accurate

Bluespec C sim → VCD output

Verilog 95 RTL → Verilog sim

Verilog sim → RTL synthesis → gates → Debussy Visualization

Legend:
- files
- Bluespec tools
- 3rd party tools
UNUM: A general framework for processor design in Bluespec

- Abstraction of a general processor which allows designers to:
  - Reuse library modules
  - Insert their own custom hardware blocks
  - Change datatypes & representations
  - Change common module parameters (scalar width)
  - Have a common testbench/verification framework

- Bluespec semantics (guarded atomic actions) preserve correctness
Bluespec Semantics

- Bluespec modules contain:
  - Local state elements
  - Local rules (guarded atomic actions)
  - Methods to access internal state
  - TRS semantic model

- Bluespec Interfaces are collections of methods instantiated *(offered)* by modules, e.g.

```plaintext
interface FIFO#(t);
  method Action enq(t) if (!full);
  method Action deq() if (!empty);
  method t first() if (!empty);
endinterface
```
UNUM Microprocessor Model

- Each unit has an interface
- Standard component library available
UNUM and ROB-based designs

- High correspondence
What about a 5-Stage Pipeline?

- PowerPC 405
  - FET
  - DCD
  - EXU
  - MEM
  - WB

Diagram:
- FET → DCD → EXU → WB → Regs → MMU
Abstract Microprocessor Model

- >85% code reuse from library
Current Progress

- **Complete:** single-issue ROB PowerPC processor
  - Implements ~190 instructions, ~230 mnemonics
  - Used for Decoder/ALU verification

- **Underway:** PowerPC 405, PowerPC 440
  - 8 instructions from running eCos

- **Beginning placing designs onto FPGA**
Future Work

- Multiprocessor Framework
  - Processor/Memory Interface
  - Cache-coherence implementation
  - Multiprocessor testbench & monitoring

- Hope to use this framework for large simulation
  - 4 processors, 2 billion instructions each
  - Testbench measuring: CPI, Cache Misses, Branch mis-predictions
  - Goal: Summer 2005
The End

Thank You

ndave@csail.mit.edu
pellauer@csail.mit.edu
arvind@csail.mit.edu