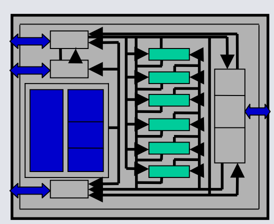
MESCAL Application Modeling and Mapping: Warpath

Andrew Mihal
and the MESCAL team
UC Berkeley

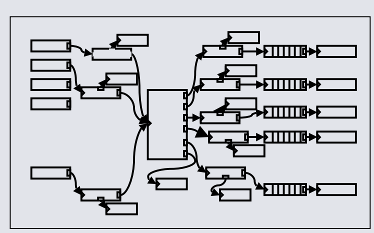
5th Biennial Ptolemy Miniconference
Berkeley, CA, May 9, 2003

Complex Systems

- Heterogeneous Architectures
  - Diverse computational resources
  - Diverse communications architecture
  - Diverse memory architecture



- Concurrent Applications
  - Multiple flavors of concurrency
  - Models of Computation



Three Challenges

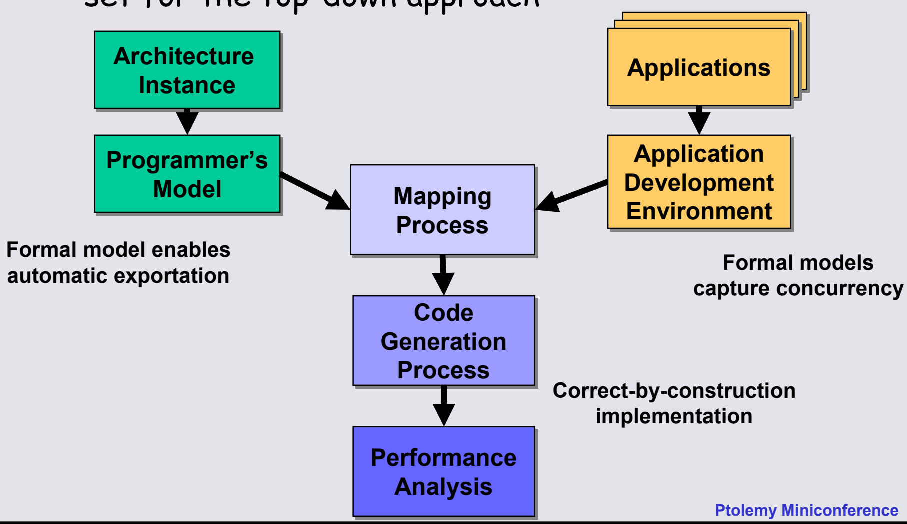
- Choose the right *application development environment*
  - Capture application's requirements
  - Useful high-level abstractions
- Find a good *programming model* for the architecture
  - Capture architecture's capabilities
  - Right mix of opacity and transparency
- *Transition* between application development environment and programmer's model (implement)
  - Enable efficient design space exploration
  - Correct results
  - Meet performance goals

MESCAL Approaches

- Bottom-up
  - Start with a specific application domain and a specific architecture
  - Develop useful abstractions of the device
  - Aspire to achieve hand-coded performance in a fraction of the design time
- Top-down
  - Consider heterogeneous applications that use combinations of MoCs
  - Develop a mapping discipline
    - Correct-by-construction implementation
    - Target a broad class of architectures

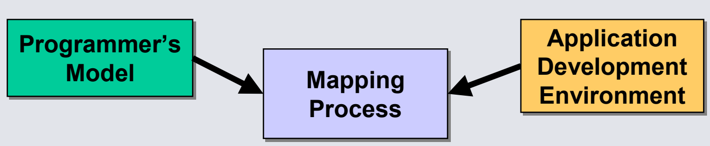
Warpath

- Disciplined methodologies and a supporting tool set for the top-down approach



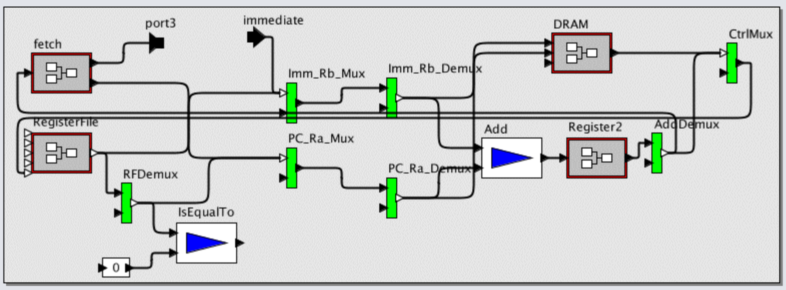
Outline

- Target Architectures
  - Exporting programming models
- Target Applications
  - Characteristics of application development environments
- Mapping

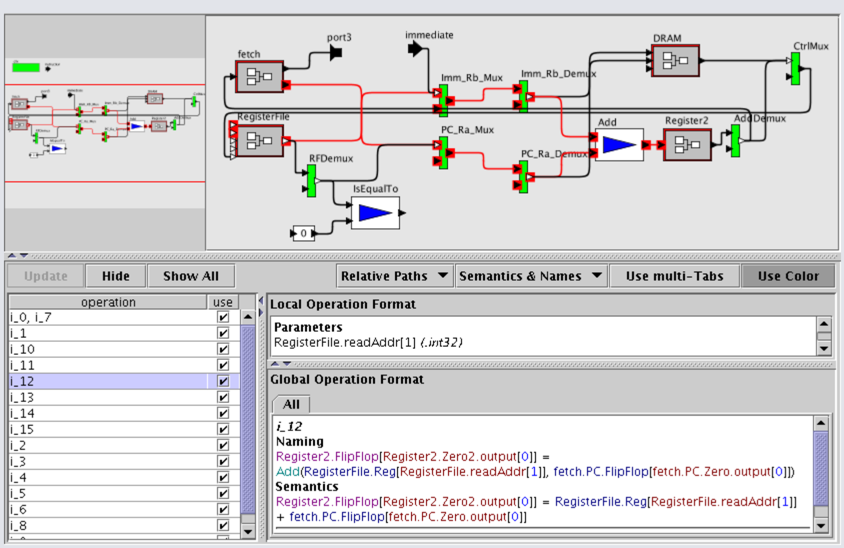


Target Architectures

- Teepee Processing Element Architecture View
  - Successor to Architecture Description Languages
  - Library of components
  - MoC captures register-to-register data transformations
  - Formal analysis finds "operations"
  - Not limited to RISC-like datapaths
- DLX-like machine:



Operation Extraction



Programming Models

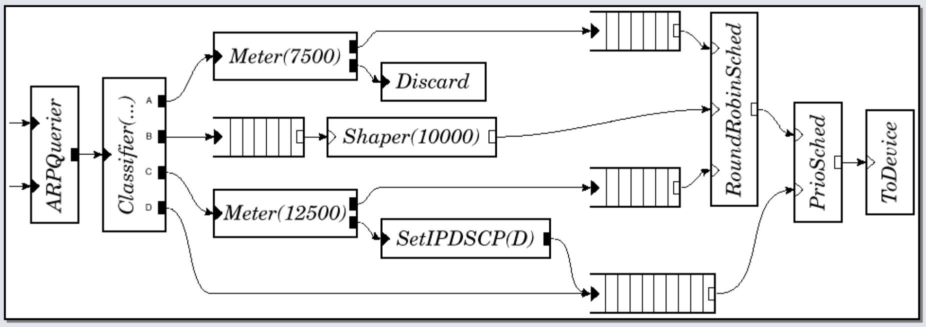
- Teepee architectures are fundamentally different from conventional RISC machines
- RISC datapath features:
  - Instruction fetch/decode units
  - Program counter
  - Part of the computation in each cycle is to figure out the next instruction to run
- Runs sequential programs with jumps
- C language
  - Arithmetic operations
  - Loops
  - Function calls
  - 20% of the architectural details, 80% of the performance

Teepee Processing Elements

- Control structures are implicit in the model
- Control synthesis strategies:
  - Hardcoded state machine
  - Horizontal/vertical microcode
  - Reconfigurable
  - RISC/VLIW
  - None of the above
- ~~Runs sequential programs~~ Executes one or more operations each cycle
- Opportunity to customize processing element control to the style of computation the application uses

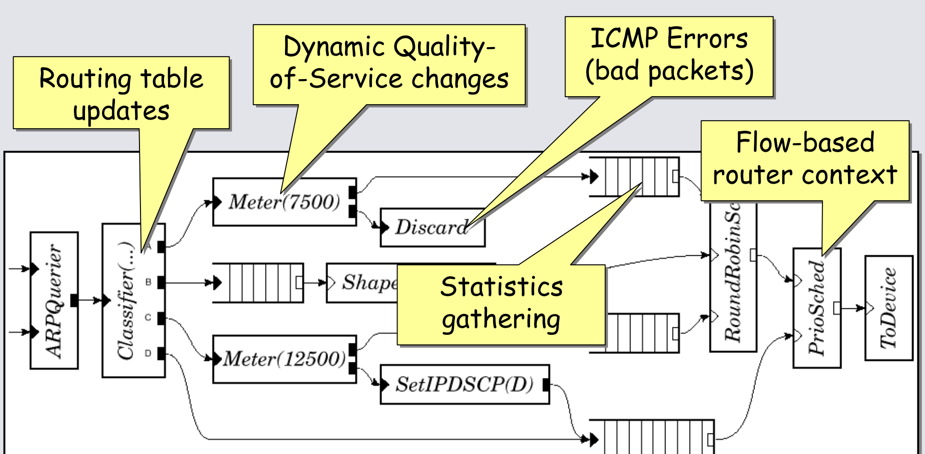
Target Applications

- Heterogeneous, concurrent applications
- Click network processing apps
- Data plane:



Click Applications

- Control plane:

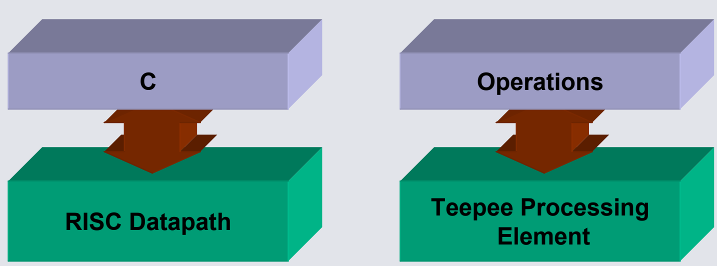


Warpath Application Development Env.

- Good ideas from Ptolemy II
  - Models of Computation
  - Orthogonalization of computation, communication, and control
  - Library of domain-polymorphic components
  - Hierarchical heterogeneity
- Targeted for implementation on a Teepee architectural platform
  - Strict software interfaces for computation, communication, control
  - Separate implementation and visualization
  - Get rid of Java
  - Don’t assume RISC-like datapaths

Application Mapping

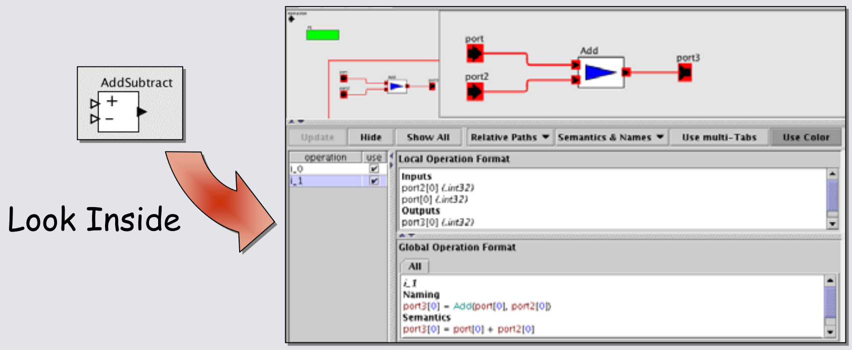
- Common abstractions on each end of the implementation transition



- Warpath application development environment describes application computation in terms of operations

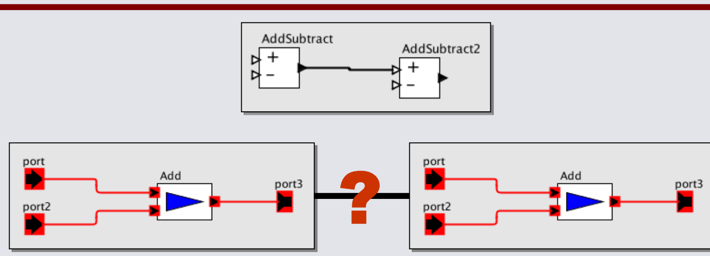
Add Actor

- Model an abstract machine that has an operation that performs the add mathematical function



- Thinking of this as a software model instead of a hardware model
- Operational semantics

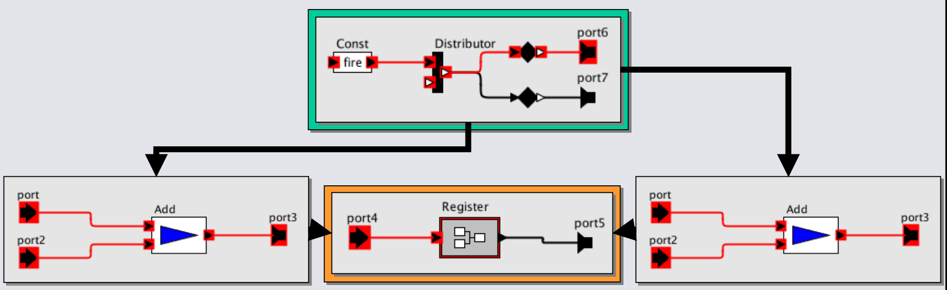
Compositions of Actors



- Obtain a graph of abstract machines
- Lacks semantics of control and communication
- Adding a model of computation makes this concrete

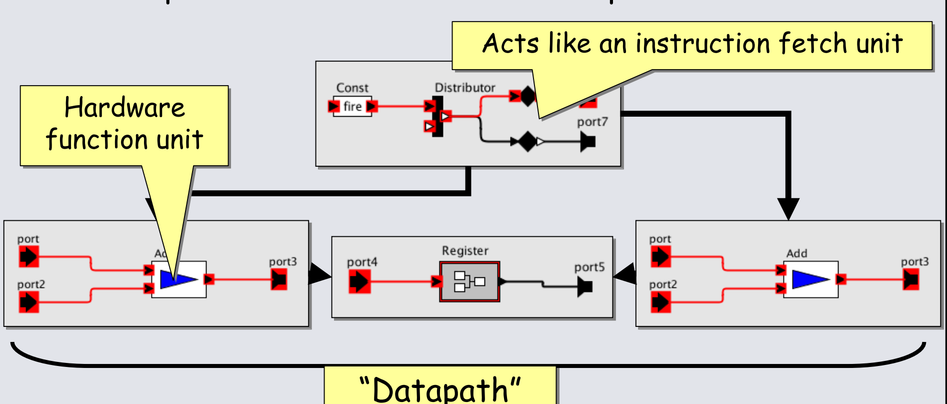
Compositions of Actors

- Receiver: operations to read and write a token to a storage element
- Director: operations to invoke operations on other abstract machines



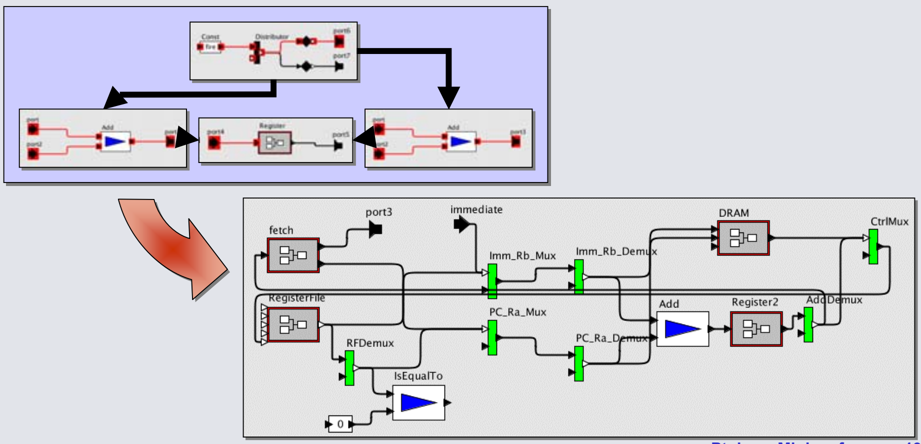
Implementation

- Base case: think of the application model as a hardware model
- One-to-one relationship between application components and architecture components



Implementation

- Next: Programmable platform
  - Compile programs for one or more PEs so that they execute the operations specified by the application model



Summary

- We maintain that the key is to have common abstractions on each end of the implementation transition
- Actors and domain components described in terms of operations
  - Operational semantics for an abstract architecture
  - Retargetable compilation process
- Designers can tune architectures to match the application
  - Application MoC influences PE control logic
  - Program counters, stacks in memory, etc. *optional*
  - Add special function units that perfom domain-specific operation
  - Explore customization/programmability tradeoffs

The MESCAL Team

- Kurt Keutzer
- Matthias Gries
- Christian Sauer
- Kees Vissers
- Yujia Jin
- Andrew Mihal
- Matt Moskewicz
- Will Plishker
- Kaushik Ravindran
- Niraj Shah
- Scott Weber