

The Mescal Project

Developing Architectural Platforms

By Randall Evans

Topics

- The need for a disciplined approach
- The key elements of the methodology
- The different views within the tool
- The importance of the mapping methodology

Introduction

- Four Challenges – Increasing complexity, large number of devices, integration of heterogeneous elements, speedy time to market
- Also must consider NRE costs
- Current solution – Programmable, multifunctional integrated circuit
- Problems exist – increased delay and power, and sometimes difficult to program
- Proposed solution – The Mescal project – producing reusable architectural platforms

Disciplined Approach

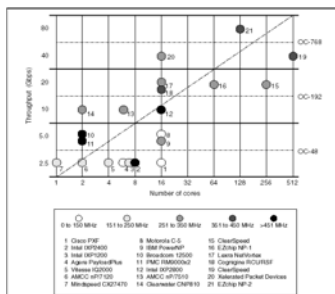
- There is great architectural diversity of modern ASIPs (see next slide)
- Therefore, a technique that addresses heterogeneity in computation and communication is required
- Mescal's methodology defines and explores the design space using tools
- They use computation and architectural models with a disciplined mapping methodology

Network Processor Diversity

■ This figure shows the diversity of network processors, even in the same performance classes

■ The architectural design space for network processors is very large, implying a larger design space for all ASIPs

■ Intuition is insufficient for the exploration of the design space



Key Elements [1]

- Developing representative benchmarks
- Defining the architectural design space
- Explore the design space using disciplined approach through a tool framework
- Formally verifying consistency between various aspects of an architecture
- Mapping applications onto architectures in a disciplined fashion

Benchmarking

- Allows designers to draw quantitative conclusions which aid design
- Helps in identifying performance bottlenecks
- Three main characteristics – representativeness, precise specification, and workload characterization

Defining the Design Space

- Several Axes –
 - Degree of parallel processing
 - Special-purpose hardware used
 - On-chip communication
 - Memory architectures
 - Peripheral integration

Parallel Processing

- ASIP architects must exploit several levels of parallelism:
 - Processing element level
 - Instruction level
 - Bit level

Special-Purpose Hardware

- Computational kernels are often chosen to be hardware accelerated on an ASIP
- Two common approaches – Coprocessors and special functional units
- Most network processors use both approaches to accelerate difficult computational tasks

Memory Architectures

- Significantly affect performance
- Its basic parameters are size, type, location, and connectivity
- Reducing latency is either hardware based or software based

On-Chip Communication

- Heterogeneity of many components requires a complex communication mechanism on the chip
- Three or four major parameters – topology, connectivity, link type, and programmability

Interaction with Peripherals

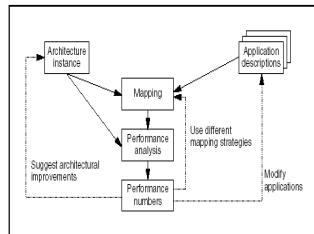
- ASIPs all interact with peripherals
- ASIP architects sometimes want to integrate peripherals on chip
- If a peripheral is integrated, the designer may want programmability

Mescal's Framework

- Mescal's goal requires clear distinction between architecture and application
- This allows for examination of an architecture's suitability for a given application

Architecture/Application Independence

- Design space exploration Y-Chart
- Feedback paths make this Y-Chart iterative, enabling greater and greater improvements
- Mescal architectural development framework (Teepee) is based on this Y-Chart

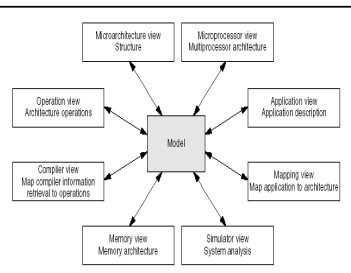


Multiple Views

- Modern architecture description languages combine both structural and instruction set semantics
- Designers restrict the flexibility to manage tool complexity
- Problem: architects, compiler designers, and simulator writers concern themselves with different aspects of the architecture

Multiple Views Methodology

- Categorization of the different views
- The different views encapsulate the architectural properties that apply to the given viewpoint
- This enables different designers to choose the best method of viewing their design



Processing Element Modeling

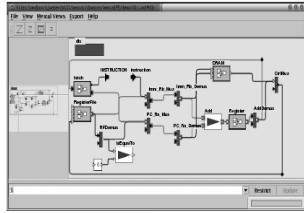
- HDLs generally describe bit level operations with little difference between control logic and data path
- Teepee can describe a circuit at the computational level, where 'actors' represent primitive computational components
- It can also specify special behavior for these actors (doesn't work unless both inputs are integers, etc.)
- Teepee currently simulates with a global clock, but plans have been made to support multiple clocks
- They are also working on developing cycle accurate compiler and simulator views

Microarchitecture View

- In Teepee, the user can choose different views

- The microarchitecture view enables designers to connect various actors explained in different ways

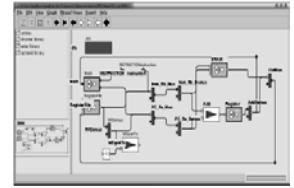
- The actors in this view obviously relate to the microarchitecture



Operation View

- In the operational view of Teepee, the actors display extracted operations

- The operations are displayed in the terms of their ports and connections



Memory Modeling

- Goal – model large memory design space at high levels of abstraction
- Includes elements from registers to memory hierarchies
- Two benefits – automated exploration during initial design and flexible enough to explore new designs without starting over
- Utilizes three basic actors – table, access and arbitrate actors

On-Chip Communication Modeling

- This view portrays the coarsest level of system architecture
- The basic blocks are processing elements and interconnecting hardware
- It's formal computational model makes the communication architectures abilities well defined
- This view allows the exploration of the tradeoffs associated with communication programmability

Application Environment

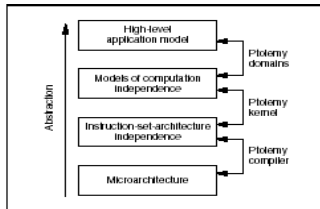
- This view is based off of the Ptolemy II framework
- Application design is difficult, requiring the juggling real-time constraints, concurrency and heterogeneity
- The formal models Mescal uses make semantic complexity implicit in the application model
- In the application view, Teepee adds new computational models to Ptolemy II

Mapping Environment

- Mapping applications onto architectures is the most important part of the Y-chart design methodology
- C-like languages are insufficient to meet requirements like concurrency and heterogeneity
- Mescal allows high level computational models to be directly applied to the system implementation
- This methodology builds on existing application design abstractions to provide a framework

Spectrum of Abstractions

- Tools provide paths between adjacent abstractions
- Toolkits like Ptolemy II enable the design of heterogeneous applications using computational models



Mapping (continued)

- Mescal's mapping methodology allows for an implementation path between high level application models to architecture
- This methodology reduces the mapping problem to matching communication and control semantics of the application with those of the architecture
- The mapping view describes architecture and application interactions
- Integrating the mapping and compiler views allows the production of executable code for each processing element

Conclusion

- The Mescal mapping methodology is the core of the project
- It reduces design time at the cost of reduced performance
- Enforcing several abstraction layers causes the loss of cross-layer optimizations
- However, this allows for a much larger design space to explore
- The Mescal project believes that the benefits outweigh the performance loss cause by using the Y-chart methodology

References

- "Developing Architectural Platforms: a Disciplined Approach"
A. Mihal, C. Kulkarni, M. Moskewicz, M. Tsai, N. Shah, S. Weber, Y. Jin, K. Keutzer, K. Vissers, C. Sauer, S. Malik,
IEEE Design and Test of Computers, volume 19, issue 6,
Nov/Dec 2002, pages 6- 16.