A Model-Integrated Design Tool for Polymorphous Embedded Systems

Brandon Eames
Esteban Osses
Vanderbilt University
Overview

- Polymorphous Embedded Systems
- A Model-Integrated Approach:
  - Metadata Authoring Toolkit
    - Capturing Machine Model Metadata
    - Application Representation
  - Design Space Exploration Tool
    - PCA Mapping as a Design Space
    - Semantic Translation onto Desert
Proliferation: Cell Phones, Drive-by-wire technologies, Avionics, toaster ovens, etc

Moore’s law: more transistors on a chip
  ▪ How to best utilize the computing power?

Application requirements and feature sets complicate designs

Increasingly difficult to implement correct and efficient systems
  ▪ Platform utilization efficiency
  ▪ Application development efficiency

Verification through testing and simulation increasingly difficult

More sophisticated approaches to system design must be adopted
Polymorphous Computing
Architectures

- Computing is inherently heterogeneous
  - Performance varies across classes of architectures
  - Different algorithms “prefer” certain architectures

- The Curse of Moore’s Law
  - Superscalar/dynamic approaches bring diminishing returns as issue width increases
  - Highly-scalable approaches increasingly difficult to program

- Polymorphous Computing: create a “morphable” architecture
  - Single chip that can be configured play the role of each class of traditional architectures
  - Reconfigurable at design time or runtime
  - Tailor the architecture to fit the shape of the application
Polymorphous Application Development

- Applications specified in high-level programming languages
  - StreamIt & Brook
  - Posix-like C thread API
- Streaming and Threaded Virtual Machines
  - Provide runtime system support for programming model abstractions
- Compiler-based infrastructure to map applications to architectures
  - High-Level Compiler for global mapping and optimization
  - Low-Level Compiler for binary translation
A Model-Integrated Approach

- Metadata Authoring Toolkit (MAT)
  - Model application and architecture metadata
  - Facilitate tool tool integration

- Design Space Exploration Toolkit (DSETool)
  - Model application composition, architecture configuration, and hw/sw mapping as a Design Space
  - Search the space for near-optimal configurations
PCAMachineModel

- PCAMachineModel: a GME-based modeling paradigm for Machine Model metadata
- Visual specification of metadata
  - Icons for Processors, Memories, Networks
  - Lines to represent resource connectivity
  - Morphs & Morph interactions specified
  - Numerical metadata captured in attributes
2.1. Root Object

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mm_Root</td>
<td>Single root object that represents the entire machine model.</td>
</tr>
</tbody>
</table>

2.2. Primary Objects

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mm_Ingredient</td>
<td>An architecture-specific hardware ingredient.</td>
</tr>
<tr>
<td>mm_Proc</td>
<td>A potential processor resource.</td>
</tr>
<tr>
<td>mm_Mem</td>
<td>A potential memory bank resource.</td>
</tr>
<tr>
<td>mm_Net</td>
<td>A potential network link resource.</td>
</tr>
<tr>
<td>mm_Morph</td>
<td>Defines a set of hardware ingredients that can be configured to produce a set of resources.</td>
</tr>
</tbody>
</table>

Manual mapping of Spec to Metamodel
PCAMachineModel Metamodel → PCAMachineModel Schema

UDM-based Automatic Translation

```
<xsd:complexType name="mm_RootType">
  <xsd:sequence>
    <xsd:element name="mm_Ingredient" type="mm_IngredientType" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="mm_Mem" type="mm_MemType" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="mm_Morph" type="mm_MorphType" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="mm_Net" type="mm_NetType" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="mm_Proc" type="mm_ProcType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="BigEndian" type="xsd:boolean" use="required"/>
  <xsd:attribute name="WordSize" type="xsd:long" use="required"/>
  <xsd:attribute name="AddressSize" type="xsd:long" use="required"/>
</xsd:complexType>
...```
Tool Integration: Machine Models for High Level Compiler

UDM Translation

PCA Machine Model.xsd-compliant XML

HLC-compliant XML

PCA High Level Compiler

October 6, 2004
A Software Model

- Application composed of a set of concurrent, interdependent tasks
- Tasks consume data, produce data
- Application modeled as a directed graph
  - Vertices represent software tasks (or “components”)
  - Edges represent data dependence
- Execution behavior of a task captured as attributes
  - Worst-case execution time, latency, throughput, etc

Diagram:

```
Task A --- Task B --- Task C --- Task D
```

October 6, 2004
Hierarchical Composition of PCA Software

- Polymorphous Software composition
  - Components model Tasks
  - Port Associations model data dependencies
  - Distinguish between Streaming and Threaded Tasks

- Hierarchical abstraction
  - Complex components composed from simple components

- Metadata
  - Component execution properties captured as model attributes
“The system shall exhibit latency of no more than ten milliseconds”

OCL syntax

```
constraint latencyConstraint() { system.latency() < 10 }
```

Constraints model

- Application requirements
- Component and resource dependencies

**Performance**: Latency from A through C less than 10 ms

**Compatibility**: Task C must be co-located with Task D

**Compatibility**: Logical Processor P0 requires Logical Memory M0

**Resource**: Task B requires a processor with at least one floating point unit
Polymorphous Computing
Mapping Problem

- Goal: Coarse-grained mapping of PCA applications onto resources, subject to constraints
  - Resolution to component implementation alternatives
  - Tasks mapped to & statically scheduled on processors
  - Communications mapped to network links, local memories
  - Temporal & spatial partitioning of morphable resources

Diagram:
- Task A
- Task B
- Task C
- Task D
- Latency(B,D)
- Constraint: \( \text{latency()} \{ \text{Latency(B,D)} < 5 \} \)

Resources:
- DCache
- P0
- Network Link
- P1
- Local RAM
Neema, Vanderbilt University
DEsign Space ExploRation Tool
- Map multi-modal dataflow-based application to heterogeneous architecture
- Constraints formally model design requirements
- Uses OBDD-s to symbolically represent and prune large design space

Symbolic representation of design space
Symbolic representation of constraints
Symbolic constraint application
Modeling Design Spaces in DESERT

- Domain-independent design space modeling language

- Hierarchical tree structure
  - Space contains Elements
  - Elements composed of Elements
    - AND decomposition: part/whole relationship
    - OR decomposition: models choice
    - LEAF decomposition: leaf node in the tree
  - Constraints apply at a particular context
DSE Semantic Translation

- Constraint
  - Expression: String
- Port
  - PortNumber: Integer
- PortAssociation
  - PacketSize: Integer
- AlternativeComponent
- StreamComponent
- ThreadComponent
- AlternativeComponent
- StreamComponent
- ThreadGroup
  - ThreadFunctionName: String
  - NumThreadsForked: Integer
  - DoNotDistributeFlag: Boolean
  - LatencyEstimate: Double
  - ThroughputEstimate: Double
  - WCETEstimate: Double
  - ThreadMemoryFootprint: Integer
  - ThreadWorkingSetSize: Integer
- Kernel
  - KernelName: String
  - ImplementationFile: String
  - MemoryFootprint: Integer
  - WorkingSetSize: Integer
  - ThroughputEstimate: Double
  - LatencyEstimate: Double
- DesertSystem
  - id: Integer
  - SystemName: String
  - name: String
  - externalID: Integer
- ConstraintSet
  - Constraint: String
  - content: String
- Relation
  - type: String
- Element
  - expression: String
  - decoupled: Boolean
- ElementRelation

October 6, 2004
AlternativeComponent: OR-Decomposition

Thread & Stream Component: AND-Decomposition

ThreadGroup & Kernel: LEAF nodes
Summary

- **Polymorphous Embedded Computing**
  - Advanced, scalable computer architectures
  - Use architecture reconfiguration to tailor platforms to fit applications

- **Model-Based System Specification**
  - Intuitive, formally analyzable visual system specification language
  - Capture applications, architectures, metadata and constraints

- **Model-Based Tool Integration**
  - Semantic translators map models in one language onto a different language
  - Machine model mapping to XML
  - Mapping of polymorphous design space onto DESERT