



Emerging Software Frameworks for Exploiting Polymorphous Computing Architectures



Sponsored by

Program Goals

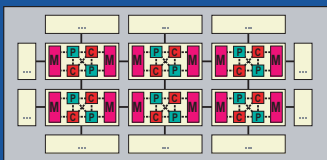
Develop embedded computing systems that can rapidly adapt and reconfigure themselves to changing application requirements

- Micro architecture elements malleable and reactive
- Computing cores
- Cache elements
- Memory elements
- Data paths
- Network interfaces
- Input/Output

Develop a software framework for PCA systems for applications that:

- Dynamically optimize hardware
- Is highly reactive
- Obtains nearly optimal performance
- Manages complexity
- Abstracts configurable computing elements
- Supports dynamic resource allocation
- Leverages existing technologies

Generic PCA Micro-Architecture



- Replicated Tile
- Configurable Processing Element
- Configurable Memory Element
- Configurable Cache Element
- Fixed Communication Path
- Configurable Communication Path
- PCA Chip

Morphware Stable Interface Forum

- Goal:** Develop the PCA software framework Quarterly meeting of PCA participants
 - First meeting June, 2001
 - Sixth meeting held in October, 2002
- Initial focus:** present existing software approaches, find common aspects, establish a common lexicon, and consider the range of target applications
- Recent focus:** establish and develop this framework
- Ongoing effort:** this framework depicts the progress to date in finding common aspects among the participating projects, leveraging existing software development tools, and exploiting the best capabilities of PCA hardware

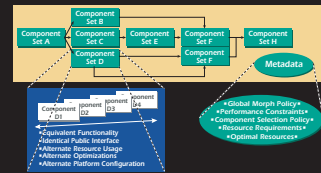
Participants This framework is the synthesis of Forum participant efforts:



Defense Advanced Research Projects Agency

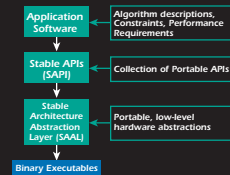
- | | |
|--|--|
| <ul style="list-style-type: none"> Applied Photonics Georgia Institute of Technology George Mason University IBM Lockheed Martin Company Massachusetts Institute of Technology MIT/Lincoln Laboratory Mercury Computing Systems Mississippi State University MPI Software Technology, Inc. | <ul style="list-style-type: none"> Northrup Grumman Raytheon SPAWAR South West Research Stanford University University of Texas - Austin University of Illinois University of Pennsylvania University of Southern California Vanderbilt University |
|--|--|

Global and Local Optimization



- MSI Forum framework divides task of resource optimization into two portions:
 - Top level application:
 - Resource, constraint, and application-requirements aware
 - Chooses PCA system global configuration
 - Is made up of one or more component sets
 - Chooses component from among sets to attain requirements and meet constraints
 - Component Sets
 - One or more objects with identical public interface, built for different constraints, resource requirements, optimal use, and hardware configurations
 - Responsible for meeting constraints, may use whatever computing resources required
 - Relatively static hardware state facilitates low level optimization
 - Similar to the proxy design pattern, with heavier-weight mechanisms for picking instantiation, and client program having explicit knowledge of the proxy
- Factors the software research into two areas
 - How to get good configuration specific performance (languages and compilers)
 - How to best decide what configuration to use, and how to best react to changing conditions, resources, and requirements (morphware)
- Defines clear, complementary roles for the various tool-building efforts
 - most participants focus primarily on one of the portions
 - allows morphware research and development independent of compilers

Dual Portability Layers



- Allows dynamic compilation
- Increases specialization opportunities in the builder/ middleware marketplace
- Reduces development for build systems by providing a common middle layer
- Allows addition of new top level approaches without breaking existing systems
- Provides a stable, platform independent target for top level build tools

Why Component Sets?

- Component Sets is a more powerful concept for PCA systems than proxies
- Explicit exposure of proxy mechanisms allows clients to express constraints and preferences, and negotiate dynamically with selection mechanisms
- Allows a variety of application construction implementations transparently to application code
- Exposes the special, PCA specific nature of component and morph selection to build systems and optimization tools

Component and Morph Selection Mechanisms

- Explicitly directed by application
- Resource availability driven
- Global application requirements driven
- Localized requirements driven

Metadata

- Non functional descriptions of requirements, constraints, desired resource management policy, or any other information which expresses preferences about system operation independent of algorithm description

Available Resource Descriptions

- Power
- Cost/ Requirements for Reconfiguration
- I/O pins, bandwidth, capability
- Memory footprint, availability, configurations
- Computation Unit size, types, quantity
- Communication pathways

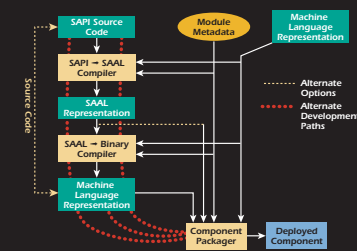
Component Performance Commitment Metadata

- Throughput/ Latency
- Elapsed time between checkpoints
- Load balancing between branches/ modules
- SWEPT
- Temperature
- Synchronization

Component Resource Requirement Metadata

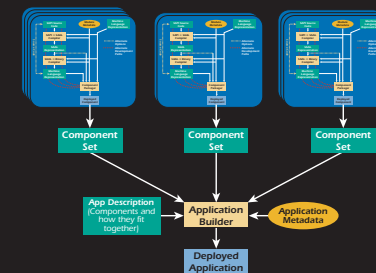
- Memory footprint, access rates, latency
- Cache footprint, access rates, latency
- I/O Capabilities
- Power

Component Development Process



- Normal development process begins with source code in one of several SAPI native languages (stream-centric C derivative, thread centric C derivative, others), a component interface description and component metadata describing resource requirements, performance commitments, optimal use situations, and optimization axes; passes through the two-stage compile, and is packaged into a deployed component
- Framework supports dynamic compile systems by allowing component development to bypass the SAAL to Binary compile layer, and the deployment of SAAL based components
- Framework supports portable components via SAPI or SAAL level deployed components
- Framework allows component vendors to bypass the entire component development process and deploy micro-optimized, platform and configuration specific components

Application Building



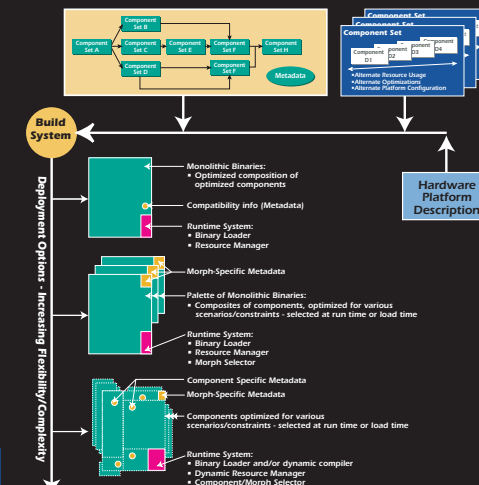
- Build system combines component sets according to application description, builds communication and flow interconnects, and builds morph policy and selection mechanisms, producing deployed application
- MSI Framework allows deployed components constructed in various ways:

Future of PCA-MSI

- MSI Forum is an ongoing effort, with details remaining to be finalized.
- Continued quarterly meetings
- Finalize SAPI language selections from among existing project languages
- Finalize SAAL syntax and structure from among existing dual layer compile approaches
- Complete metadata specification syntax descriptions
- Detailed description of software framework : June, 2003

Application Build & Deploy Options

- MSI Framework supports varying deployment options – from the static case of a single, monolithic binary executable to reactivity through a palette of binaries built for specific metadata conditions to fully dynamic applications consisting of run-time selected components
- More flexible configurations require more robust build systems, and run time systems, but provide more agile applications



GTRI morphware website :
<http://seal.gatech.edu/morphware> – links to PCA project websites and background info

DARPA PCA program website:
<http://www.darpa.mil/ipto/research/pca/> – detailed PCA program mission, background, and status

Contact: Dan Campbell dan.campbell@gtri.gatech.edu
 Ken MacKenzie kenmac@cc.gatech.edu
 Mark Richards mark.richards@ece.gatech.edu

