The Morphware Stable Interface: A Software Framework for Polymorphous Computing Architectures

Daniel P. Campbell
Mark A. Richards
Georgia Institute of Technology
Atlanta, GA
770-528-7541 / dan.campbell@gtri.gatech.edu

Dennis M. Cottel
Randall R. Judd
SPAWAR Systems Center
San Diego, CA

Kenneth M. Mackenzie
Reservoir Labs, Inc.
New York, NY

Abstract

Polymorphous Computing Architectures (PCAs) are computing devices that are capable of significant, rapid reconfiguration directed by software. Composed of several groups of computing elements, PCA devices can be configured to achieve high performance on a wide variety of problem types and processing demands. We describe an emerging concept called the Morphware Stable Interface (MSI), a portable, scalable software development framework to harness the power and complexity of PCAs, while allowing productive software development, and rapid adoption of PCA devices.

1. INTRODUCTION

The Polymorphous Computing Architectures (PCA) program is a Defense Advanced Research Projects Agency (DARPA) effort to develop new embedded computing platforms with very strong, rapid in-mission reconfigurability. Target applications range from military platforms that must adapt to rapidly changing mission parameters, to embedded network controllers, whose optimal configuration of hardware resources will change in response to the traffic and environmental conditions it faces.

The PCA program "core projects" working to develop microprocessors that implement polymorphous capabilities include Smart Memories [1], Raw [2], M3T [3], TRIPS [4], and MONARCH [5]. The chips under development in these projects have several characteristics in common. These are typically tiled structures composed from replicated, fully capable computing cores, reconfigurable memory and cache elements, and a rich set of reconfigurable data paths, network interfaces, and I/O paths. Each can operate in streaming or threaded modes. Each has mechanisms for aggregating individual processor tiles into larger compound processor units. They differ in their approach for aggregating processors and in their emphasis on processor, memory, or communication design. Figure 1 illustrates a generic PCA microarchitecture.

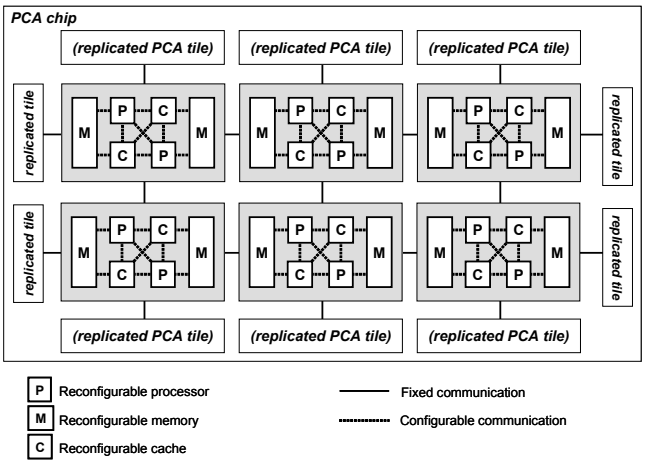

Figure 1. Generic PCA chip micro-architecture.

The increased capability of PCA systems comes at the expense of increased software complexity. Applications written with knowledge about the platform embedded into the structure of the application can make use of the reconfigurability of such resources, but suffer from a lack of scalability and portability. Applications written with no such knowledge are completely dependent on build and run-time systems to utilize the capabilities of reconfigurable systems.

An important goal of the PCA program is therefore to create an application development framework, called the Morphware Stable Interface (MSI), that will exploit the capabilities of PCA hardware while retaining as much portability and performance as possible. The MSI should:

- Support dynamic hardware reorganization and optimization
- Obtain nearly optimal performance
- Abstract configurable computing elements
- Abstract hardware reconfiguration
- Mitigate development and runtime complexity
- Leverage existing technologies.

This paper presents the early results of the MSI design effort.

2. THE MORPHWARE FORUM

To facilitate the development of the cross-project consensus and design effort needed to realize the MSI, the PCA program has formed the Morphware Forum, an informal consortium of the PCA contractors and other selected participants. Organized and led by Georgia Tech under DARPA sponsorship, the Forum meets quarterly, with other interim activities as required, to develop and debate proposals for the MSI. The Forum will ultimately develop a set of publicly available standards documents that define the architecture and details of the MSI.

Georgia Tech maintains a Morphware Forum web site at http://www.morphware.org that provides a vehicle for meeting planning, collaborative exchange, and public information about the MSI effort. At this writing, the public portion of the site is limited to introductory papers and briefings on the MSI effort and the PCA systems, and links to program participants' web sites. MSI standards documents will be available at this site as the Forum approves them.

3. MORPHWARE STABLE INTERFACE

The MSI is a framework consisting of several major elements. Some of the most significant aspects of the MSI are described.

3.1 The Streaming Virtual Machine

Several of the projects have developed specialized languages to exploit the differences between PCAs and traditional CPUs. The highly parallelized, low unit-to-unit latency of PCA devices exposes a need for compilers to more fully understand data dependencies and control flow within an application than is possible with languages such as C. In order to expose these elements more clearly to compilers, the projects have developed variants of C and C++ that introduce and enforce data-stream, and operational kernel constructs. These language express programs as directed data flow graphs connecting various computational kernels. The restrictions on data and control flow allow compilers to map algorithms very effectively to PCA-style devices, while simplifying source code for a wide range of applications. These languages express a fundamentally different underlying virtual machine than languages such as C, and form an important aspect of the MSI.

3.2 MSI Portability Layers

From its inception, the PCA program has advocated dual portability layers for the MSI. The MSI provides portability via the upper "Stable API" (SAPI) and the lower "Stable Architecture Abstraction Layer" (SAAL). Figure 2 illustrates the concept. It reduces development effort for application build systems by providing a common middle layer; allows addition of new top level application approaches without breaking existing build systems; provides a stable, platform-independent target for top level build tools; allows dynamic compilation, and increases specialization opportunities in the builder/middleware marketplace.

The SAPI consists of multiple platform-independent, high-level languages and metadata representations. These languages will be extensions to existing languages such as C or C++ that express each of the virtual machine abstractions present in the SAAL. The SAAL will be a portable C-based representation of a virtual machine with both streaming and threaded modes. It will be both platform- and SAPI language-independent. Finally, it will be lower level than SAPI languages, and able to support dynamic compilation. The Morphware Forum is currently actively developing detailed proposals for the SAAL VM.

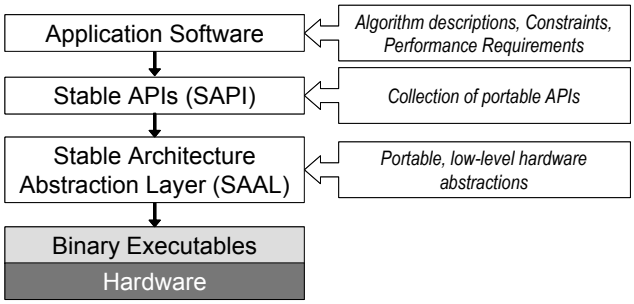

Figure 2. SAPI and SAAL MSI portability layers.

3.3 Component-Based Framework

The ability of applications to intelligently reconfigure host platforms is a critical element of PCA software. In order to facilitate morphing, a component-based software framework has been proposed. Components provide natural and intuitive boundaries for run-time reconfiguration of hardware. Software components will be built for varying hardware configurations, and the appropriate version may be loaded and executed on the fly as portions of the host platform are reconfigured by the PCA system or application software.

3.4 Metadata

PCA systems must to be aware of and reactive to application, resource, and constraint goals and requirements. Examples include SWEPT (size, weight, energy, performance, and time) constraints, quality of service requirements (*e.g.* latency, throughput), morph policy, security requirements, and so forth. Metadata provides a means for the MSI to represent the information needed to implement this capability.

Metadata are non-functional descriptions of requirements, constraints, desired resource management policy, hardware configuration options or any other information that expresses information about system operation independent of the application functional description. Each of the uses of metadata constitutes a unique context that must be standardized and described. A standard method for describing metadata contexts has been proposed and is currently under consideration by the Morphware Forum. Specifications for several metadata contexts have been proposed, and standard appropriate storage, retrieval, and query mechanisms for the metadata are currently being designed.

4. REFERENCES

[1] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. Dally, and M. Horowitz, "Smart Memories: A Modular Reconfigurable Architecture", *Proceedings* International Symposium on Computer Architecture, June 2000.

[2] M. B. Taylor, J. Kim, *et al.*, "The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs", IEEE *Micro*, April/March 2002.

[3] C. Cascaval *et al*, "Evaluation of a Multithreaded Architecture for Cellular Computing", *Proceedings* Eighth International Symposium on High-Performance Computer Architecture (HPCA), February 2002.

[4] R. Nagarajan *et al*, "A Design Space Evaluation of Grid Processor Architectures", *Proceedings* 34th Annual International Symposium on Microarchitecture, pp.40-51, December, 2001.

[5] J. Granacki and M. Vahey, "MONARCH: A Morphable Networked micro-ARCHitecture", High Performance Embedded Computing Workshop, October 2002.