

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

Mark A. Richards, Daniel P. Campbell, and Kenneth M. Mackenzie

Georgia Institute of Technology

*Presented at GOMAC 2003
2 April 2003*



The PCA Program

- **DARPA effort for high performance embedded platforms with strong, rapid, reactive in-mission configurability**
 - Support dynamic and multi-mission requirements
 - Support collaborative, information-centric missions
- **PCA will develop processing architectures that “morph”**
 - Hardware and software resources reconfigure to balance resource requirements and availability
 - at multiple levels: microarchitecture, network, system
 - at multiple time scales: in-mission, between-mission
- **Five “core” projects in Phase I developing highly capable microarchitectures**
 - M3T, MONARCH, Raw, Smart Memories, TRIPS
 - Based on replicated, tiled structures to address the “wire delay” problem
 - Supported by eight additional hardware and software technology projects

Generic PCA Microarchitecture

■ Tiled structure

- fully capable computing cores

- **reconfigurable memory** and cache

- rich set of **reconfigurable data paths**, network interfaces, and I/O

- **streaming and threaded** modes

- **methods for aggregating tiles** into larger processing units

- performance on the order of (per chip)

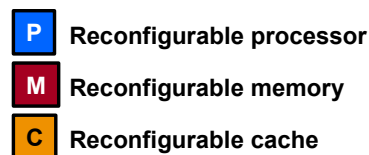
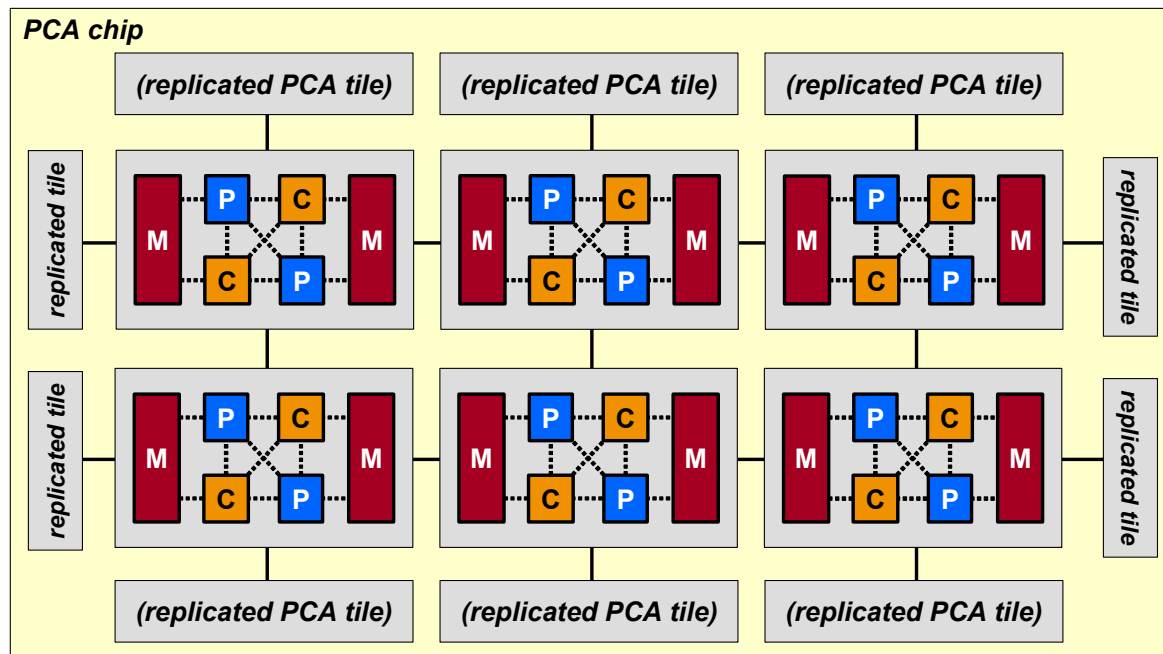
- **4 – 64 GFLOPS and 4 – 16 GOPS**

- **25 – 32 GB/s off-chip I/O**

■ Core projects differ in

- aggregation mechanisms

- relative emphasis on processor, memory, or comm design



Morphing

- **Morphing of embedded resources is the key capability of PCA systems that enables us to meet the goal of “strong, rapid, reactive in-mission configurability”**
- **Can occur in response to multiple sources of stimuli**
 - User command, application command
 - Run-time system
 - Compiler
- **Morphing occurs at various levels and scales**
 - substitution of alternate implementations of same functionality
 - optimize performance vs. power
 - reallocation of chip resources from streaming to threaded processing
 - reallocation of resources to adapt to varying loads
 - request/release an ALU, reallocate a memory block from general to cache
 - fault tolerance
- **Morph decisions divided among user app, run-time system, and compilers**

Software and PCA

- **Increased hardware capability and complexity brings increased software complexity**
 - **If we build target platform reconfigurability and performance info into the app, we will lose scalability and portability**
 - **If we don't, the build and run-time systems will be entirely responsible for leveraging the platform capability, and we still lose fine-grain morphability**
 - **Applications must be reactive to feedback from the hardware**
 - **resource collisions, SWEPT, faults**
- **Solution: the Morphware Stable Interface (MSI)**

The Morphware Stable Interface (MSI)

- **Application Development Framework for PCAs**
- **Comprised of a software architecture and a suite of open standard APIs**
- **Goals**
 - **Dynamically optimize PCA resources for application functionality, service requirements, and constraints**
 - **Obtain nearly optimal performance from PCA hardware**
 - **Be highly reactive to PCA hardware and user inputs**
 - **Manage PCA software complexity**
 - **Leverage existing and already-developing technologies**
- **Cross-project effort, developed in parallel with the hardware**

The Morphware Forum

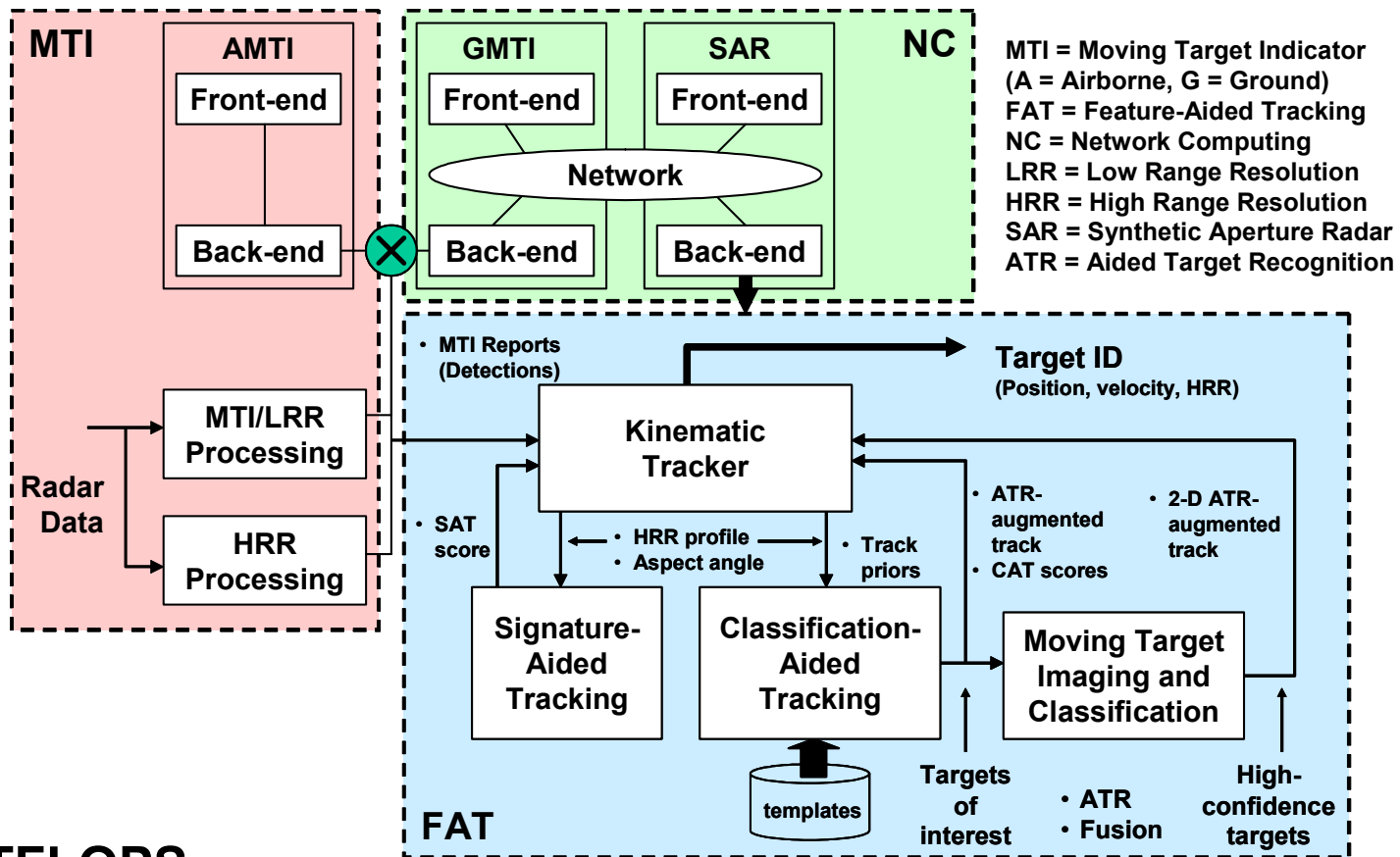


- **Informal consortium of the PCA contractors and other selected participants**
- **Organized and led by the Georgia Tech/SPAWAR team**
- **Meets quarterly**
 - **interim meetings and activities as required**
- **Propose, debate, develop, test, validate, document, and demonstrate standards that define the MSI**

Canonical DoD Application

■ Integrated Radar Tracker PCA benchmark under development by MIT/LL

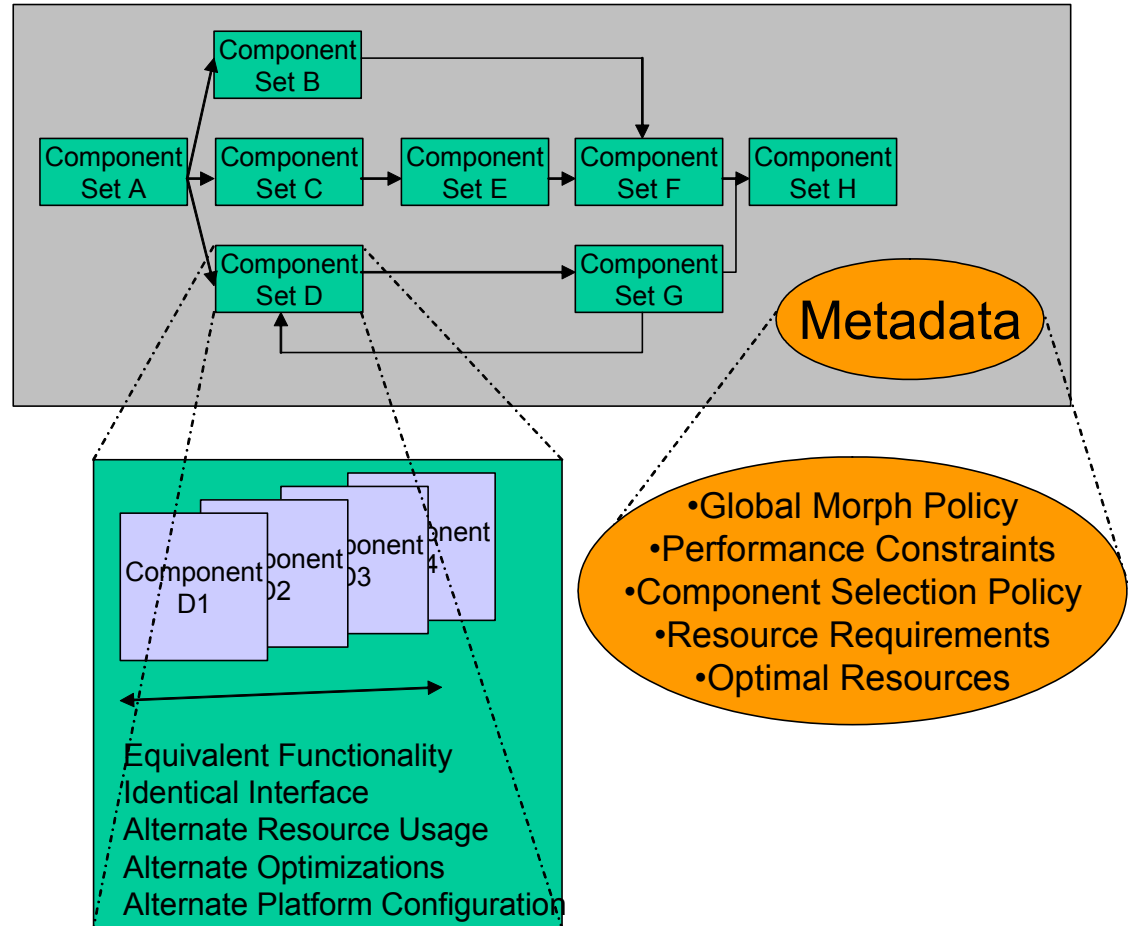
- combines streaming (MTI, NC) and data-dependent/decision directed (FAT) processing
- current base version ~100 GFLOPS
- full version will be multi-TFLOPS



(Figure courtesy of MIT Lincoln Laboratory)

MSI Architecture

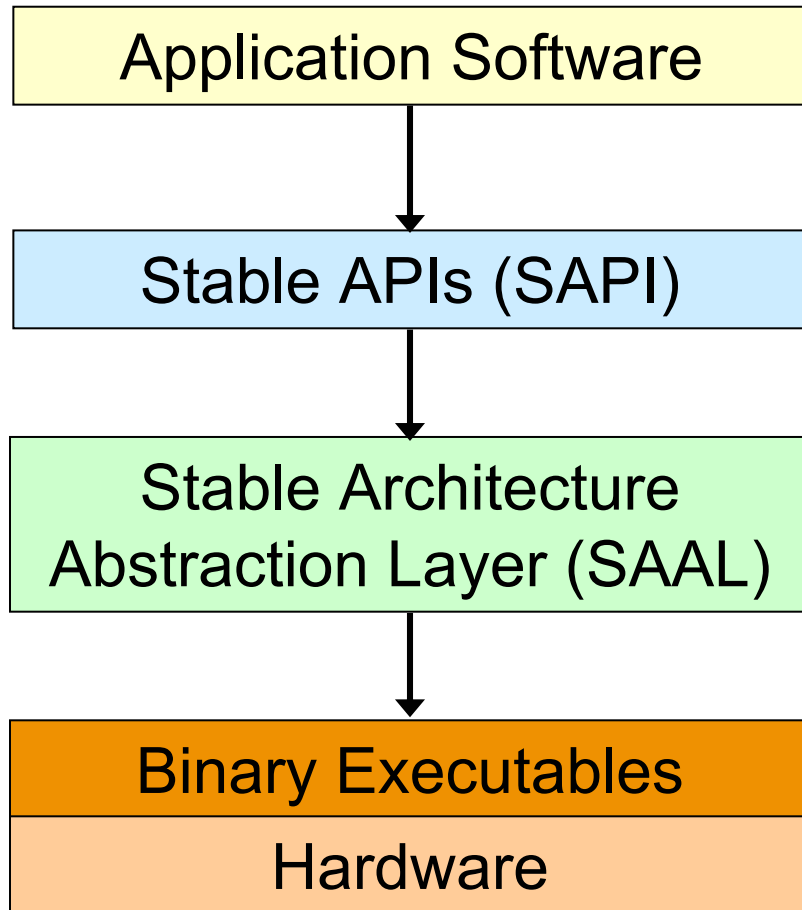
- The MSI architecture is evolving towards a high-level **component-based approach** to support scalability and morphing, incorporating ...
- ... an advanced **stream VM**, coupled with a relatively conventional **thread VM**, for building components; supported by ...
- ... a **metadata system** for expressing constraints and controlling morph behavior



Why a Component Architecture?

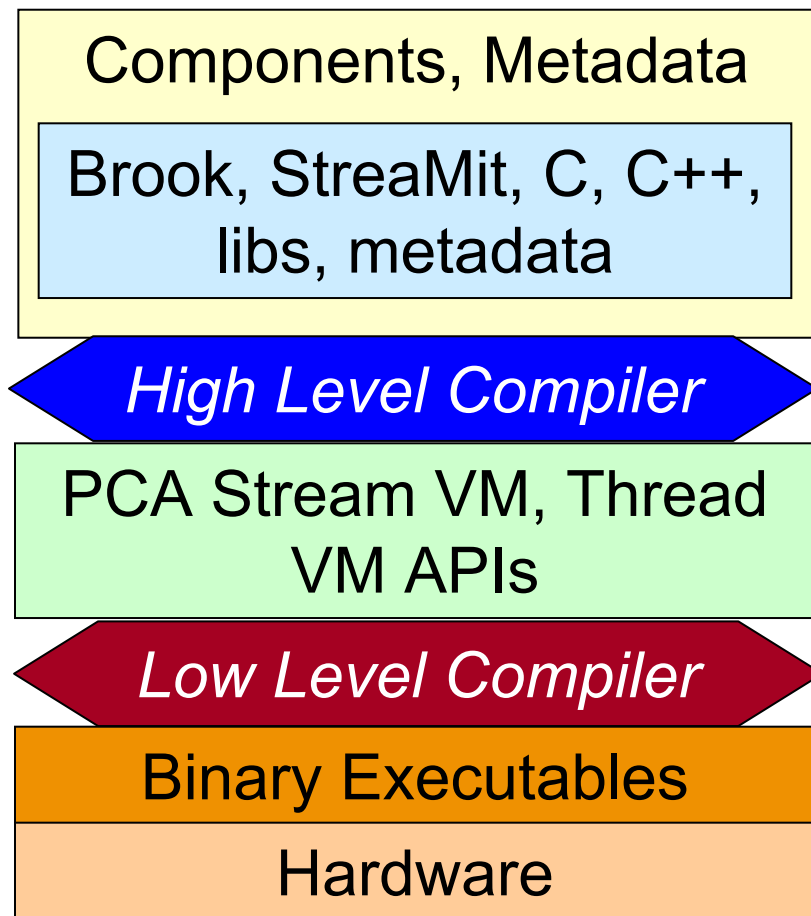
- **Encapsulate run-time changes**
 - Reconfiguration of hardware
 - Reload/restart of software elements
 - Redeployment of software elements
 - Redistribution of software elements among hardware
- **Reduce resource footprint**
 - Load only the subset of application that is demanded
- **Intra-deployment interoperability**
 - Allow subsets of applications to interoperate within an application, regardless of source, deployment time
- **Heterogeneous resource support**
 - Allow subsets of an application to make use of disparate resources by encapsulating functionality
- **Life-cycle development & maintenance cost management**
- **Deployment modularity**
 - Deploy components independently
 - Update system piecemeal
- **Encapsulate communication / computation boundaries**

Building Components: SAPI and SAAL



- **Stable API (SAPI) and Stable Architecture Abstraction Layer (SAAL) provide dual portability layers**
- **Application SW describes functionality, constraints, and performance requirements**
- **SAPI is PCA-aware collection of standardized language and service APIs**
- **SAAL is PCA-aware abstracted low-level machine representations**

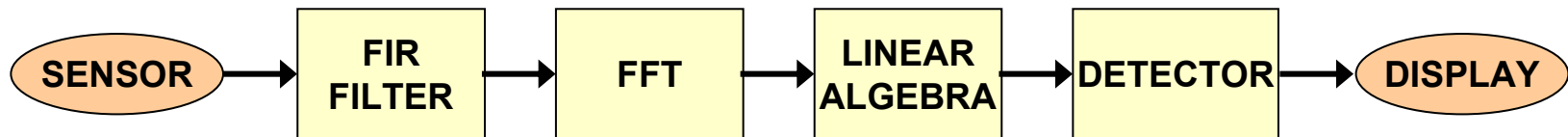
Two-Level Compile



- **SAPI and SAAL implemented through two-level compile**
- **Application SW combines C/C++ and stream language with metadata**
- **Vendor-neutral high-level compiler produces SVM/TVM code and more metadata**
- **Vendor-specific low-level compiler targets specific PCA machine**

Why Stream/Thread Distinction?

- Major compute-intensive portions of many embedded signal processing applications have characteristics of stream operations
 - fixed data flow graph
 - large, possibly infinite, data stream
 - functional kernels not data-dependent
 - but parameterized
 - functional kernels independent of one another
 - little or no retained data or state



- Representation that recognizes/captures these characteristics significantly aids compiler optimization, scheduling, resource allocation

PCA Stream Virtual Machine Concepts

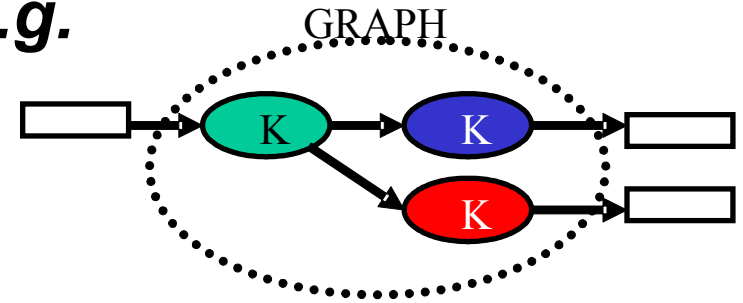
(courtesy of Peter Mattson and Reservoir, Inc.)

- PCA SVM built on just 4 major concepts:
 - Kernel
 - Computation- and data- intensive function executed on a single stream processor (one at a time)
 - Generated by high-level compiler, may not correspond to a “conceptual” kernel
 - Stream and Block
 - Data stream/block operated on by kernel, bound to a specific resource
 - Graph
 - Set of kernels which concurrently read/write streams in a synchronized fashion
 - Defined and controlled by a thread



PCA Stream VM Progress

- Development of the stream VM (SVM) API has been very active
- Consensus on many details, *e.g.*
 - Stream definition, attributes, processing order, ...
 - Kernel as a graph, multi-I/O, state retention, control by TVM, ...
 - Kernel restrictions: no dynamic memory, pointers, GOTO, recursion, ...
- Still plenty more details
 - Thread/stream interaction: passing data, *etc.*
- New update of SVM due at June 2003
Morphware Forum meeting

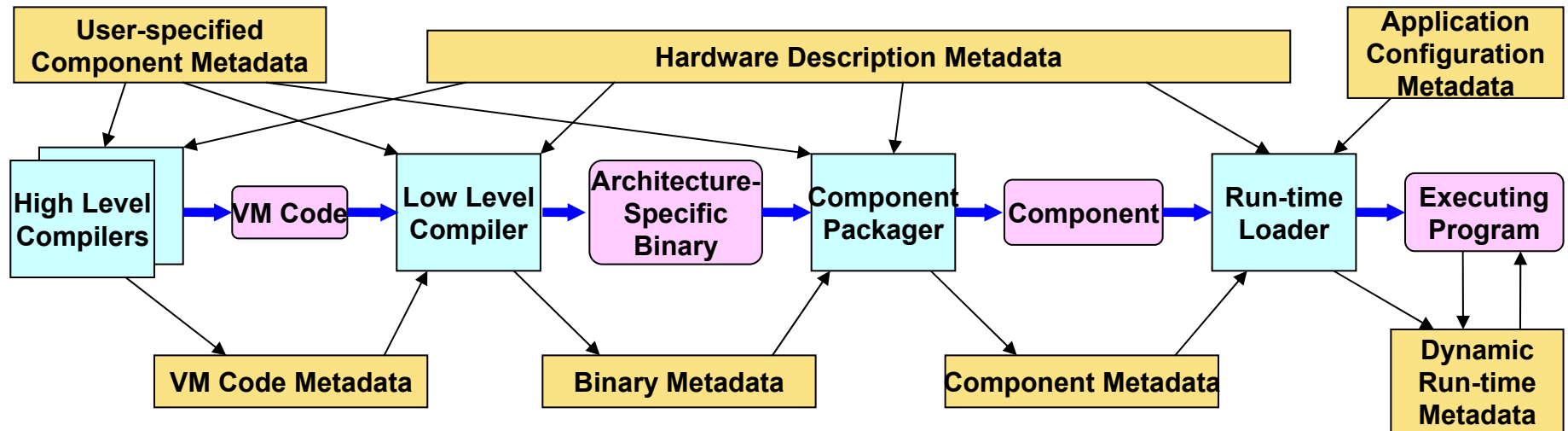


PCA Thread VM Progress

- **Current model is essentially a POSIX-based conventional thread machine**
 - pthreads, VIA communication architecture
- **Currently evaluating a proposed restructuring of the TVM into two elements:**
 - simpler, lower-level “hardware architecture language” (HAL) with low-level network inspired by RapidIO as the SAAL VM, paired with ...
 - ... a higher-level PCA run-time system that builds POSIX, MPI, *etc.* as desired/required on top of the HAL
- **Also considering an alternative processor scheduling modeling based on “scheduler activations”**
 - related to IBM K42

Metadata System

- Metadata needed throughout the PCA system
 - One method of representation preferable to many
 - Needed to enable processor and compiler developers to progress
- VM definition efforts to date have resulted in some sample VM metadata implementations and dictionaries
 - but not comprehensive enough
- Currently considering standardizing the XML representation and dictionary instead of the APIs
 - would leverage many existing university and commercial toolchains
 - accommodates procedural or static representation



Sample Metadata Types

<i>Available Resource Descriptions</i>	<i>Component Performance Commitments</i>	<i>Component Resource Requirement</i>
<ul style="list-style-type: none"> • Power • Cost and requirements for reconfiguration • I/O pins, bandwidth, capability • Memory footprint, availability, configurations • Computation unit size, types, quantity • Communication pathways 	<ul style="list-style-type: none"> • Throughput and latency • Elapsed time between checkpoints • Load balancing between branches or modules • SWEPT metrics • Temperature • Synchronization 	<ul style="list-style-type: none"> • Memory footprint, access rates, latency • Cache footprint, access rates, latency • I/O capabilities • Power



Morphware Forum Major Milestones

- **Summer 2003:**
 - SVM & TVM API “0.1”
 - VM Metadata structure
- **Winter 2003/4:**
 - Metadata system “0.1” (structure + content)
 - IRT benchmark in SAPI (stream+thread languages)
 - Core architectures described in metadata system
- **Summer 2004:**
 - PCA run-time system “0.1” and document
 - Simple app build-and-run demos
 - Component architecture “0.1”
- **Winter 2003/4:**
 - Multiple app + morphing build-and-run demos
 - Integrated PCA component and run-time system “0.2”
- **Continuing updates to MSI components throughout ...**



- The Morphware Forum web site provides some public information
 - Selected public papers & briefings
 - Links to PCA project sites and related links
 - Link to DARPA PCA
 - This paper and presentation, soon
- In the future, it will provide one-stop public dissemination of MSI documents

