DARPA Polymorphous Computing Architectures (PCA) Program

ECE 4100/6100  Advanced Computer Architecture

November 25th, 2003
Kenneth M. Mackenzie
Reservoir Labs, Inc

(includes slides from many sources)
“Polymorphous” ?!

Idea: processors that can be tailored (“morphed”) to suit application domains.
   specifically: DSP-style vs. general-purpose style

Inspiration: flock of “post-superscalar” architecture projects
   all attempting to build scalable processors out of replicated components (“tiles”)

DARPA (Defense Advanced Research Projects Agency): recognize application opportunity
   provide umbrella application goal
   also demand an actual software infrastructure
Outline

Why/how post-superscalar wire delay

DARPA PCA program how to herd cats

Tour of five core architecture projects
- MIT "RAW"
- Stanford "Smart Memories"
- UT Austin "TRIPS"
- UIUC/IBM "M3T" (tied to IBM’s "Blue Gene")
- ISI "MONARCH"

Software structures
- portability layer
- common "high level" compiler
- possible morphing scenarios
Outline

Why/how post-superscalar wire delay

DARPA PCA program
  how to herd cats  

Tour of five core architecture projects
  MIT “RAW”
  Stanford “Smart Memories”
  UT Austin “TRIPS”
  UIUC/IBM “M3T” (tied to IBM’s “Blue Gene”)
  ISI “MONARCH”

Software structures
  portability layer
  common “high level” compiler
  possible morphing scenarios

GTRI Sudha, CREST

Reservoir Labs
THE BIG SIMPLIFICATION
more speed, less power

Avoid kitchen-sink approach

In future, wiring $\geq$ gates

Data *transport* is as important as computation: “networking” issues arise even on-chip...
Abstraction Layers for Computation

```plaintext
foo(int x) { .. }
```

Computation
Language / API
Compiler / OS
ISA
Micro Architecture
Layout
Design Style
Design Rules
Process
Materials Science
Physics

Fortran
IBM 360 /RISC/ Transmeta/x

Mead & Conway

Illustration: adapted from Michael Taylor, presentation to HotChips, 2001
Abstractions protect us from change but must eventually change as the world changes.

Language / API
Compiler / OS
ISA
Micro Architecture
Layout
Design Style
Design Rules
Process
Materials Science

Changes in physical constraints

More Resources:
Wire
Delay
Gates
Wires
Pins

Illustration: adapted from Michael Taylor, presentation to HotChips, 2001
Wire delay is *crashing* through the abstraction layers

<table>
<thead>
<tr>
<th>Language / API</th>
<th>Partitioning (21264)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler / OS</td>
<td>Pipelining, trace cache (P4)</td>
</tr>
<tr>
<td>ISA</td>
<td>Timing Driven Placement</td>
</tr>
<tr>
<td>Micro Architecture</td>
<td>Fatter wires</td>
</tr>
<tr>
<td>Layout</td>
<td>Deeper wires</td>
</tr>
<tr>
<td>Design Rules</td>
<td>Cu wires</td>
</tr>
<tr>
<td>Process</td>
<td></td>
</tr>
<tr>
<td>Materials Science</td>
<td></td>
</tr>
</tbody>
</table>

Illustration: adapted from Michael Taylor, presentation to HotChips, 2001
The future of wire delay handled in micro-architecture...

Illustration: adapted from Michael Taylor, presentation to HotChips, 2001
Tiled Architectures (ex: MIT “RAW”)
Coupling tiles into “processors”?

Many approaches that permit dynamic scheduling
  RAW: fast, register-level network
  Speculative threading (Stanford Hydra; many others)
  Hydra rhetoric: either have “lots” or “little” ILP.

Note: static styles for DSP problems also
  VLIW/SIMD
  systolic (pipelined parallel)

Rich space for exploration
Also makes you wonder: can you pick the coupling mechanism on an application-by-application basis?
Raw: Example use of tiles

Illustration: adapted from Michael Taylor, presentation to HotChips, 2001
DARPA PCA Program
Polymorphous Computing Architectures

Idea: make a formal goal that we can use these architectures for multiple targets

Performance within x% of dedicated designs

Bonus: can “morph” on the fly.

Rub: actual compilers and software systems to start *now* designing applications that use these systems
    get real user feedback to architects
Enable reactive multi-mission and in-flight retargetable embedded information computing systems that will reduce mission computing payload adaptation, optimization and verification from years to months to minutes.

Polymorphous Computing

- Application Software
- Malleable SW/HW Abstractions & Stable Interfaces
- Micro-Architecture HW

Polymorphic - Adj. Having, taking, or passing through many different forms or stages. (Greek polus many + morphe form)
Example Application: Radar

good model for any sensor-driven application
Example Application: Radar

good model for any sensor-driven application

sensor → signal processing task → tracking task → operator

normal mode: vision: computing resources are tiles you can devote to DSP or threaded problems

high detail

many targets
PCA Importance to DoD

**Multi-Mission/Multi-Sensor**
- Mission Driver Sensor “Plug-and-Play”
- Flexible Sensor/Payload Applications
- Improved EO/IR
- SAR/MTI
- SIGINT
- HSI

**Tactical UAV (TUAV)**

**Common Evolvable Computing Platform**

**Lower Cost Mission Critical Processing**

**Rapid Upgradability**

**Missionized Avionics**

**Future Combat Systems**

**DoD Existing Platforms**

B52 platform to remain in service for 70+ years

**OODA Loop**
- Observe
- Orient
- Decide
- Act

Missionized Avionics

Pilot’s Decision Loop

CPU Mem

0 %

100 %

0 %

100 %

Updated EO/IR

SAR/MTI

SIGINT

Slide: Bob Graybill, DARPA
PCA Program Overview

Phase 1
FY01 – FY03

Research Phase

Mission Requirements Analysis & Application Requirements Decomposition
- USC/ISI (East)
  - BAE
  - Northrop Grumman
  - Lockheed Martin
- MIT/Lincoln Laboratory
- DoD Contractors
- NSWC

Polymorphous Computing Research & Proof-of-Concept
- University of Illinois – Urbana Champaign
  - IBM (Watson)
  - BAE
  - Vanderbilt University
  - South West Research
- MIT Lab for Computer Science
- Stanford University
- University of Texas – Austin
  - IBM (Austin)
- USC/ISI (West)
  - Mercury Computing
  - Raytheon

Focused Technology Dev
- Northrop Grumman (High confidence)
- USC/ISI (Free Space Optics)
  - George Mason Univ
  - Applied Photonics
- Lockheed Martin (V&V)
  - Lockheed Martin Aero
  - University of Pennsylvania
- MPI Software (Morphware)
  - Mississippi State University
- Mercury Computing (Morphware)

Phase 2
FY03 – FY05

Development & Demonstration Phase

Mackenzie 18

slide: Bob Graybill, DARPA
Outline

Why/how post-superscalar wire delay
DARPA PCA program how to herd cats
Tour of five core architecture projects
  MIT "RAW"
  Stanford "Smart Memories"
  UT Austin "TRIPS"
  UIUC/IBM "M3T" (tied to IBM’s "Blue Gene")
  ISI "MONARCH"
Software structures
  portability layer
  common high-level compiler
  possible morphing scenarios
MIT RAW

Idea: small (one ALU/FPU) tiles connected by a very fast network
Parallelize superbblocks onto tiles
Illustration: adapted from Michael Taylor, presentation to HotChips, 2001
Memory Mapped Communication is Not a First Class Citizen

To other tiles, through memory system that happens to go over a network.

Slide: David Wentlaff, presentation to HPEC Workshop, 2001
Raw’s First Class Register-Mapped Communication

Ex: add r26, r25, r24

Network Input FIFOs

Network Output FIFOs

Slide: David Wentlaff, presentation to HPEC Workshop, 2001
Raw: Example use of tiles

Illustration: adapted from Michael Taylor, presentation to HotChips, 2001
source: RAW website
RAW Summary

Focuses on interconnect

I think the tiles are so small in part to force the interconnect story

Multiple software systems

gcc  (MIMD mode or manual control mode)
RawCC  (parallelize-at-block-level)
SUDS  (speculative threading support)
StreamIT (streaming language generates systolic programs)

Status: back from fab as of 02Q4

Best source: http://www.cag.lcs.mit.edu/raw/
Stanford Smart Memories

Processor tiles connected to configurable memory units

Threaded mode: memories support speculative threading, a la “Hydra”

Stream mode: memories are stream register files with local DMA, a la “Imagine”

Interesting observation: tuck FPGA logic into repeaters and muxes at negligible cost
Smart Memories Layout

A tile

A quad

Quad networks

Processing tile or DRAM block

Crossbar interconnect

Processor

Memory system of 16 8KB SRAMs

Quad interface

processor is dual issue + FPU

Illustration: Smart Memories paper, ISCA, 2000
**Streaming Memory System**

- Imagine memory system
  - Virtually multi-ported stream register file
  - DMA engine for gather/scatter of streams to/from DRAM

**Cached Memory System**

- Hydra memory system
  - 2-level cache hierarchy
  - Fine grain speculation support needs extra bookkeeping bits

---

slide: Bob Graybill, DARPA
Smart Memories Summary

Emphasis on morphing via reconfigurable memory units
   FPGA-type logic means you stop to reconfigure
   Many hardware goodies like associative tag check logic

Processors are more conventional (and heavyweight) than RAW

Multiple software systems
   gcc
   hydra compiler
   “Brook” (stream mode compiler)
   DSP-C

Best source: Paper in ISCA 2000
UT Austin TRIPS
Tera-something-something-something-something

“Grid Processor” made up of many (16-64) ALUs

Idea is to spread superblocks across multiple ALUs like a place-and-route problem. ALU fires dynamically when data is available

Systolic mode fixes instructions in place and fires them repeatedly

Memory units (L1 and L2) configurable as explicit memory or as cache
35 nanometer target design

- 15x15 mm chip
- 8 processors, mapping 8K instructions
- 256KB of L1 I and D storage
- 2MB of L2 storage
- 256 GB/s off-chip bandwidth
- 4 Teraflops peak (FP codes)
- 512 GIPS (integer codes)

Burger, UT-Austin, 2/13/02
Grid Processor organization

Diagram showing the components of a grid processor:
- Instruction Sequencer
- Register File
- Memory Interface
- Block Termination Control
- Memory
- Input ports
- Op1, Op2
- Inst Storage
- ALU
- Router
- Output ports
Grid Processor Architectures

- Three goals to keep in mind:
  - Expose much ILP (large window)
  - Minimize physical distances
  - Avoid large centralized structures
TRIPS Summary

Focus is on the grid processor

Threaded execution by place-n-route of dataflow

Stream execution by fixing instructions at ALUs and repeating them

Best source: paper in ISCA 2003
UIUC M3T
“Morphable Multithreaded Memory Tiles”

M3T emulated on IBM BlueGene chip “Cyclops”

Thread units are integer processors in “quads”
FPU for every four TUs
10 cycle TU-to-TU interconnect time

units tied together at the level of mini-tasks
of 10-45 instructions (commeasurate
w/interconnect time)
M3T Emulation on Blue Gene

- **Blue Gene chip architecture:**
  - 128 threads of execution
  - 32 FPUs @ 500 MHz / 1 Gflop
  - 512KB shared cache in 32 banks
  - 8 MB of memory in 16 banks
  - switches for interchip operation
  - access to off-chip DRAM
  - fast thread synchronization
  - atomic memory operations

*Slide: Josep Torrellas, presentation at PCA PI meeting, June, 2001*
Some Compiler/Synthesizer Knobs

**Superscalar:**
- Instruction issue width
- Types and latencies of functional units

**Systolic array:**
- Connectivity between processors
- Communication cost between processors

**SIMD:**
- Number of processors working in lockstep
- Grain size of the step
**M3T Summary**

Focus is on the dynamic scheduling mechanism

Cyclops (IBM) focus is raw data parallelism

Dynamic scheduling by “mini-tasks”

Stream execution in ordinary MIMD modes

Best source: website
USC/ISI MONARCH

MOrphable Network ARCHitecture

ALUs laid out with small intervening memory units. Sequenced by a microcontroller.

Memory on-chip with local connections and a global crossbar

Grew out of two projects
ISI "DIVA": processor-in-memory w/MMX/SSE-type unit
Raytheon "HPPS": systolic array
MONARCH Single Chip Architecture

- External Memory Interface
- High Speed I/O
- XBar
- Parcel Logic
- Selectable Buffer ("FPGA")
- I/O Adaptor
- DRAM 2 MB (3 copies)
- Mem Ctrl
- \(\mu\)Controller
- I Cache
- FPU
- 12 ports
- 3.2 GBytes/sec I/O rate
- 24 GBytes/sec I/O rate
- 800 MHz Clock
- 512 ops/clock
- 12 GFLOPS
- 400 GOPS
- 32 MBytes DRAM
- 320 KBytes SRAM
- 256 MALU
- 36 Watts

Slide: John Granacki, presentation at PCA PI meeting, June, 2001
Monarch Modes

1. Each slice is a processor; 2x A units per processor (4x 32-bit FPUs per A unit) are like MMX/SSE units.

   slice boundaries don’t matter in this mode
   M units (register files) include address generation

3. arbitrary combination of above.
Architecture Summary

Tour of five core architecture projects
  MIT “RAW”
  Stanford “Smart Memories”
  UT Austin “TRIPS”
  UIUC/IBM “M3T” (tied to IBM’s “Blue Gene”)
  ISI “MONARCH”

Various points of focus (interconnect, memory, processors)
Different mechanisms to tie tiles together
Different “knobs”

All tiled, all with threaded- and stream- modes
Outline

Why/how post-superscalar
  wire delay
DARPA PCA program
  how to herd cats
Tour of five core architecture projects
  MIT “RAW”
  Stanford “Smart Memories”
  UT Austin “TRIPS”
  UIUC/IBM “M3T” (tied to IBM’s “Blue Gene”)
  ISI “MONARCH”

Software structures
  portability layer
  possible morphing scenarios
  common “high-level” compiler
Polymorphous Computing
Architectures Program Goal

Enable reactive multi-mission and in-flight retargetable embedded information computing systems that will reduce mission computing payload adaptation, optimization and verification from years to months to minutes

Polymorphous Computing

- Application Software
- Malleable SW/HW Abstractions & Stable Interfaces
- Micro-Architecture HW

Polymorphic - Adj. Having, taking, or passing through many different forms or stages. (Greek polus many + morphe form)
Insist on a Portability Layer

languages:

c/c++ w/ threads, c/c++ w/MPI, StreamIT, Brook, other

portability layer (“SAAL”)

platform-indep. compilers

platform-dependent compilers

architectures:

RAW  SM  TRIPS  M3T  MONARCH  other...
Morphing Scenarios
using sensor processing as the motivating application

sensor → signal processing task → tracking task → operator

- **normal mode:**
  - high detail
  - many targets

vision: computing resources are tiles you can devote to DSP or threaded problems
Top View of PCA Application Development

Available tools are a standard language, e.g., C, and component APIs.

Select Component Sets (boxes)

Define Data and Control Flow (arrows)

Determine Global Morph Policies, Constraints, Resource Requirements

Define System I/O

Metadata

- Global Morph Policy
- Performance Constraints
- Component Selection Policy
- Resource Requirements
- Optimal Resources

Equivalent Functionality
- Identical Interface
- Alternate Resource Usage
- Alternate Optimizations
- Alternate Platform Configuration

Component Set A
Component Set B
Component Set C
Component Set D
Component Set E
Component Set F
Component Set G
Component Set H

Component Set A
Component Set B
Component Set C
Component Set D
Component Set E
Component Set F
Component Set G
Component Set H

Slide: Dan Campbell, GTRI
Monolithic Binary: Optimized composition of optimized components

Palette of Monolithic Binaries: Compositions of components, optimized for various scenarios/constraints - selected at run time or load time

Components optimized for various scenarios/constraints - selected at run time or load time

more detail on next page
Application Build & Deploy – 2 of 2

**Build System**

- **Monolithic Binary**: Optimized composition of optimized components

**Runtime System**
- Binary Loader
- Resource Manager

**Compatibility Info (Metadata)**

**Palette of Monolithic Binaries**: Compositions of components, optimized for various scenarios/constraints - selected at run time or load time

- **Runtime System**: Binary Loader (+) Resource Manager (+) Morph Selector

**Component-Specific Metadata**

**Morph-Specific Metadata**

**Deployment Options - Increasing Flexibility/Complexity**

- **Components optimized for various scenarios/constraints** - selected at run time or load time

  - **Runtime System**: Binary Loader and/or JIT/Dynamic Compiler (+) Dynamic Resource Manager (++) Component/Morph Selector (+)

**Slide: Dan Campbell, GTRI**

Mackenzie 50
High Level Compiler
Reservoir Labs

languages:

c/c++ w/ threads, c/c++ w/MPI, StreamIT, **Brook**, other

*portability layer ("SAAL")*

architectures:

RAW  SM  TRIPS  M3T  MONARCH  other...
High-Level Compiler

Generic compiler for streaming architectures

Outline:
- gross structure of a compiler
- input languages (initially "Brook")
- key problem: mapping and selecting the configuration status
Structure of a Compiler

Front End
- Brook Parser
- StreaMIT Parser

Internals
- AST
- Elaboration

Back End
- Code Generation

Extract/Enhance Parallelism
- Mapping

IR

λ

a

b

*

*

+

return

app

printf

"%d"

π1

π2

I/O

app

app

app

app

πi

πj
Input Language?

C?

Fortran?

Brook...

C with explicitly parallel “kernels”
In use now as a programming model for graphics cards
(c.f. Pat Hanrahan’s talk here last week)
**Brook Example**

Kernels perform computations on streams. This kernel computes pair-wise sum.

```
void kernel streamAdd(float s1<>, float s2<>, out float s3<>) {
    s3 = s1 + s2;
}
```

```
void kernel weightedSum(float image_in<>[3][3], out float image_out<>) {
    image_out = 1.0 / 9.0 *
        (image_in[0][0] + image_in[0][1] + image_in[0][2] +
         image_in[1][0] + image_in[1][1] + image_in[1][2] +
         image_in[2][0] + image_in[2][1] + image_in[2][2]);
}
```

```
int main() {
    float image[100][100][3][3];
    float imageOut[100][100];
    float st<>[3][3];
    float stOut<>;
    float stOutDouble<>;
    ... 
    streamRead(st, image, 2, 0, 100, 0, 100);
    weightedSum(st, stOut);
    streamAdd(stOut, stOut, stOutDouble);
    streamWrite(imageOut, stOutDouble, 2, 0, 100, 0, 100);
    ...
}
```

Stream is 1D, but elements can be arrays. This is a stream of 3x3 arrays.

Use of stream operator to read from array into stream.

Use of streamAdd kernel to double stream.

Represents `s3.push(s1.pop() + s2.pop())`. 
Goals of Mapping

Exploit parallelism in all forms (data, task, pipeline, ILP)

Co-optimize mapping and configuration

Optimize given constraints
  extremely limited local memory for instance

Optimize given alternative objectives
  latency instead of throughput
  power
Mapping Problem: Input is IR

consider only task mapping for simplicity

for (;;)
   for (;;)
   for (;;)
      while ()
         for (;;)

while ()
   for (;;)
      for (;;)

source program
(tasks are the loops/kernels)

task graph from the IR
(at some granularity)
Mapping Problem: Choose from many Possible Mappings
circles are tasks, rectangles are processors

Data Parallel

Task Parallel

Pipelined

Combinations

Aside: edges are storage or communication
Configuration Problem

1. **Static Morphing**: pick the CONFIGURATION and the mapping (a co-optimization problem)

   *E.g. pick # tiles/processor*

2. **Dynamic morphing**: pick a SEQUENCE of configurations and orchestrate morphing between them.

   *E.g. suppose (A) is data-parallel but ILP-poor.*
Searching the Solution Space

Search given constraints and an objective.
Many-dimensional search space

Tools:
- map problems to known frameworks (DAGs, ILP)
- use heuristics to prune the search tree and to limit backtracking
- fallback: solve subproblems in phases
Example Strategy

Phase 1: partition program into reasonable-size pieces.

Phase 2: schedule pieces as if they were “instructions” in a VLIW.

Pros: draw on vast body of work in VLIW scheduling

Cons?
Status

Just released version 1.0 of the compiler targets a fixed configuration (i.e. just does mapping) does a lousy job! provides a base for LLC and application developers

Plan: develop a sequence of compilers
1.x fixed configuration
2.0 compiler selects the fixed configuration
3.0 compiler selects sequences of configurations
High-Level Compiler Summary

Generic compiler for streaming architectures

Targets multiple languages, initially Brook

Crux problem: simultaneously select mapping and configuration

Plan: develop a sequence of compilers
PCA Summary

“Post-Superscalar”: tiled architectures

DARPA PCA vision: tailor archs to applications

Tour of five core architecture projects
  MIT “RAW”
  Stanford “Smart Memories”
  UT Austin “TRIPS”
  UIUC/IBM “M3T” (tied to IBM’s “Blue Gene”)
  ISI “MONARCH”

Software structures
  portability for usage; portability to crossfertilize ideas