CHOMP: A Framework and Instruction Set for Latency Tolerant, Massively Multithreaded Processors

THE WORLD'S FIRST HYBRID-CORE COMPUTER.

John Leidel, Kevin Wadleigh, Joe Bolding, Tony Brewer, Dean Walker

IA^3 Workshop on Irregular Applications: Architectures and Algorithms

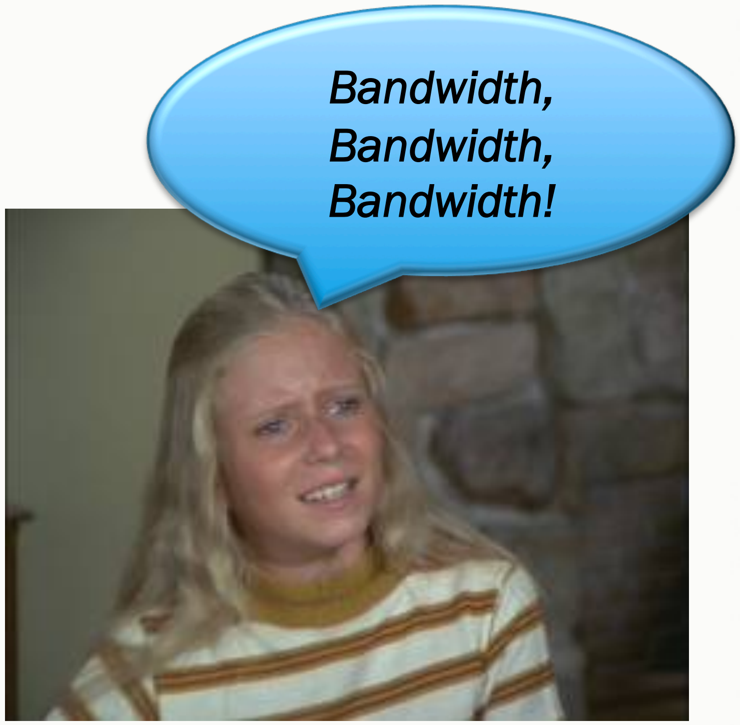
Overview

- Motivations
- MX-100 Hardware Overview
- CHOMP Personality Overview
- CHOMP Instruction Set
- Application Examples

MOTIVATIONS

An HPC Programmer's Wish List

- **Globally shared, single word memory access**
- **Lightweight synchronization and barrier operators**
- **Latency hiding techniques**
- **Simple parallel scalability**
- **Bandwidth!**
- **...all with familiar and/or commodity programming models**
  - Not all programming models are created equal
  - None are perfect, but industry adoption is paramount



CONVEY computer™

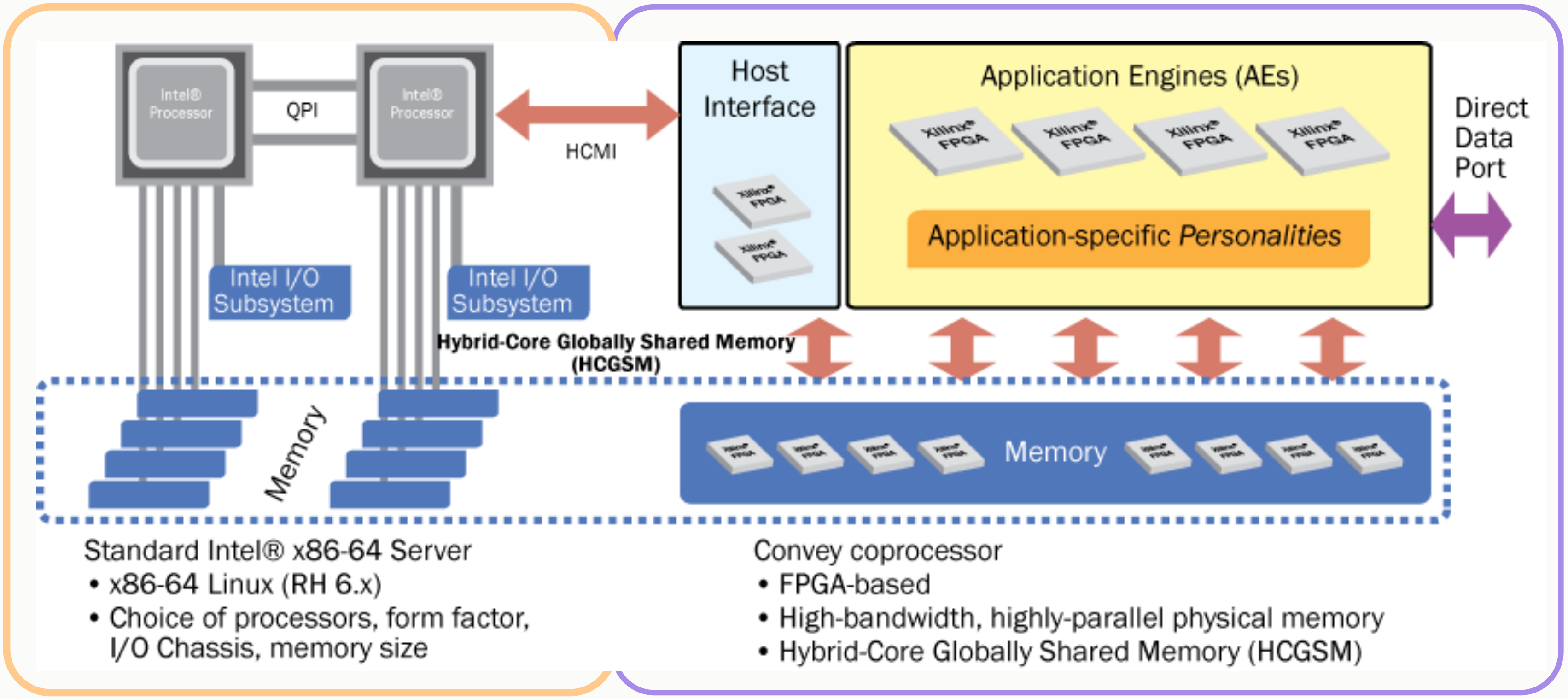
Architectural Wish List

Globally Shared Memory	Memory Interconnect	
Single Word Memory Access	Scatter/Gather Memory	
Lightweight Sync/ Barriers	Atomic Memory Operations	Tag-Bit Semantics
Latency Hiding	Memory & Compute Overlap	
Bandwidth	Wide, Concurrent Memory Bus	
Simple Scalability	Develop on your laptop, execute on the super	
Commodity Programming Model	Portable Code	

CONVEY computer™

MX-100 HARDWARE OVERVIEW

MX-100 Platform



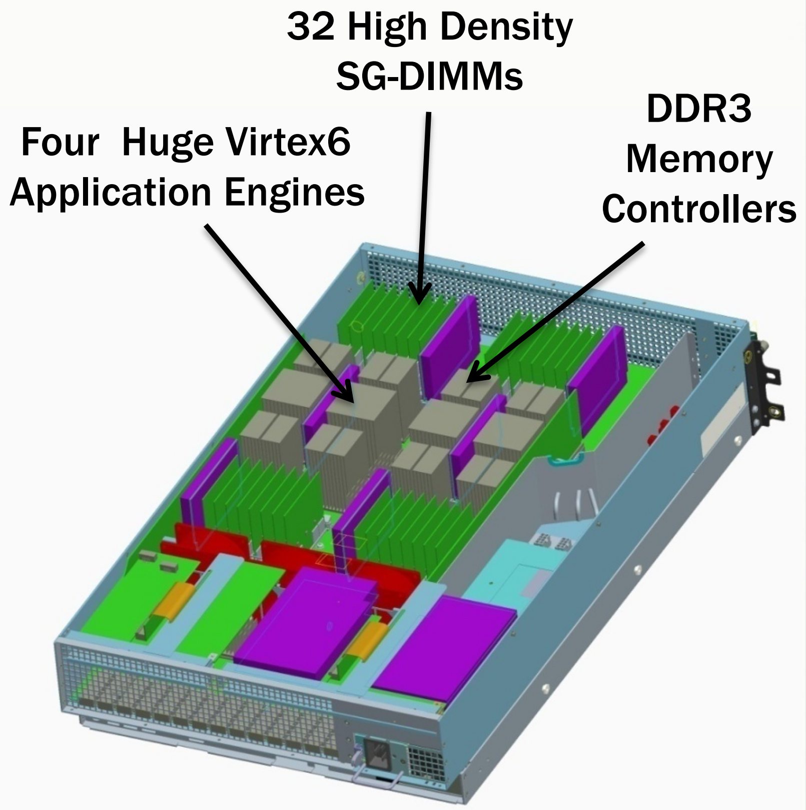
Intel® Processor
QPI
Intel® Processor
HCMI
Host Interface
Xilinx® FPGA
Application Engines (AEs)
Application-specific Personalities
Direct Data Port
Intel I/O Subsystem
Intel I/O Subsystem
Hybrid-Core Globally Shared Memory (HCGSM)
Memory
Memory
Standard Intel® x86-64 Server
• x86-64 Linux (RH 6.x)
• Choice of processors, form factor, I/O Chassis, memory size
Convey coprocessor
• FPGA-based
• High-bandwidth, highly-parallel physical memory
• Hybrid-Core Globally Shared Memory (HCGSM)

•HCMI = Hybrid Core Memory Interconnect; PCIe Gen2 X8 link
•All memory, host and coprocessor, is globally shared and virtually addressable

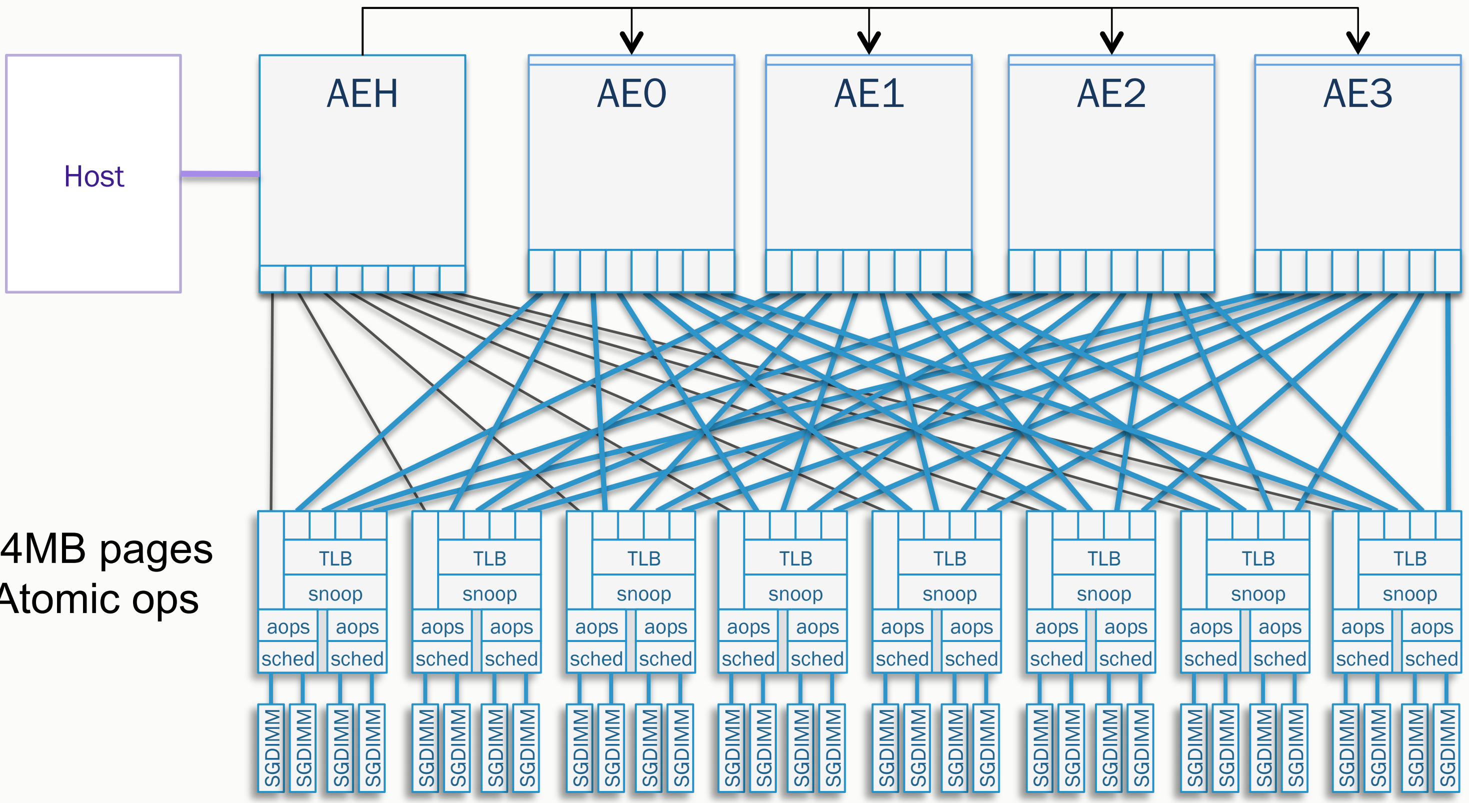
MX-100 Coprocessor

Shared, Virtual Memory System

- **Scatter/Gather**
  - 1TB of coprocessor memory
  - 128 GB/s bandwidth to coprocessor memory
- **Atomic Memory Operators**
  - Native to memory controllers
  - 8-Byte Accessible
    - Does not require fetching cachelines
- **Tag Bit Semantics**
  - lock or “tag” bit for each 8-byte word



MX-100 Single-node Block Diagram



1024 TIDs per link (32K OR)

32TB (45 bit) physical address space

64MB pages
Atomic ops

32 DDR3 SG-DIMMs
HalfDIMM up to 4GB - 256GB system
FullDIMM up to 32GB - 1TB system

CONVEY computer™

Atomic Operations & FE Memory Bits

- **Atomic operations avoid round trips to memory to acquire lock, update data, release lock**
  - Accessible from coprocessor instruction space
  - Accessible from host via lock engine and compiler intrinsics

- **Full/Empty Bits**
  - Extra bit stored with each word
  - Can be used to signify when data is valid/ready
  - Accessible from host via lock engine & compiler intrinsics

Atomic Operations
Add, Sub
Min, Max
Exch
Inc, Dec
CAS
And, Or, Xor

Tag Bit Semantic Operations
WriteEF, WriteFF
WriteXF, WriteXE
ReadFE, ReadEF, ReadFF
ReadXX
IncFF

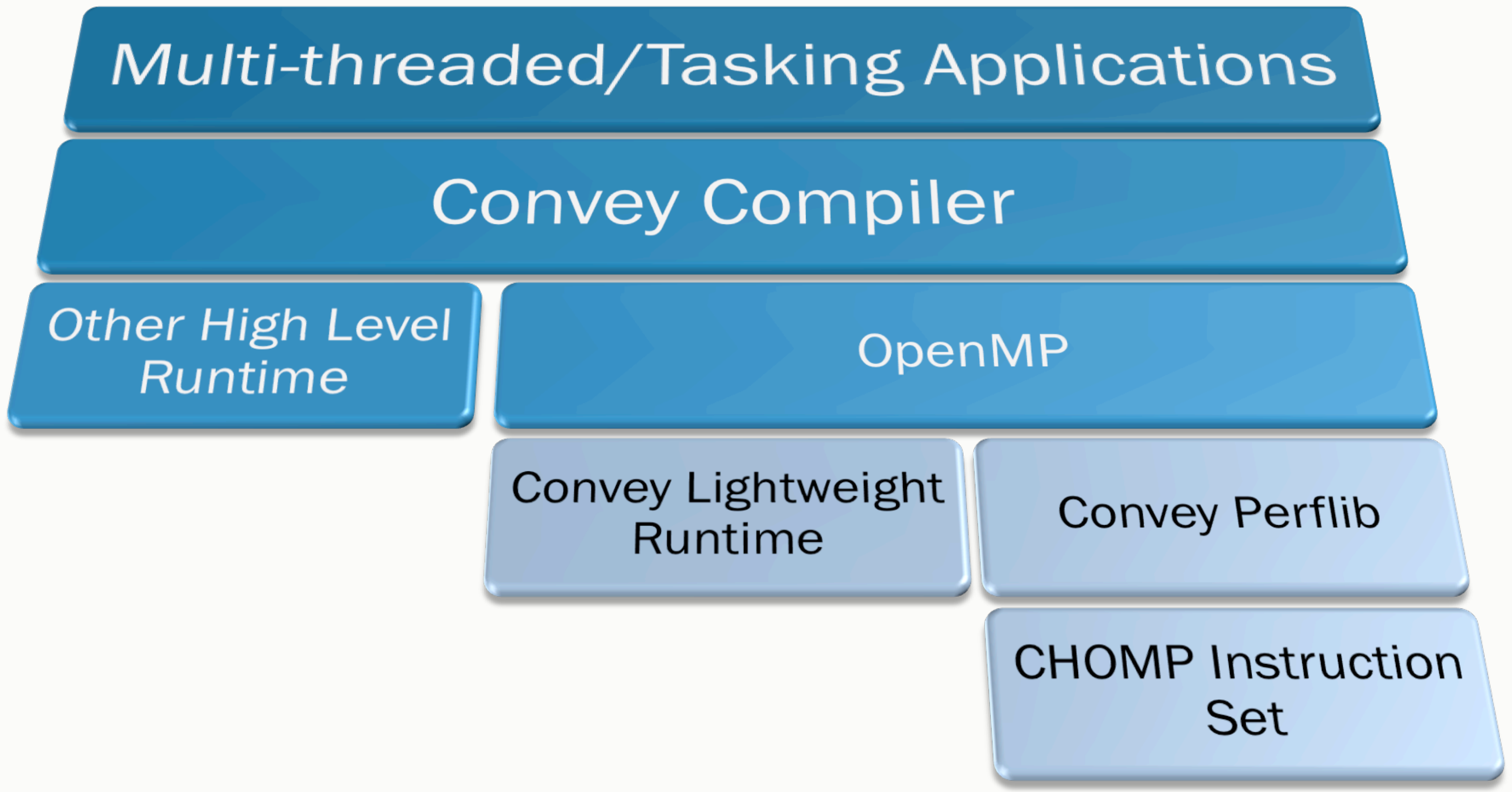
CONVEY computer™

CHOMP PERSONALITY OVERVIEW

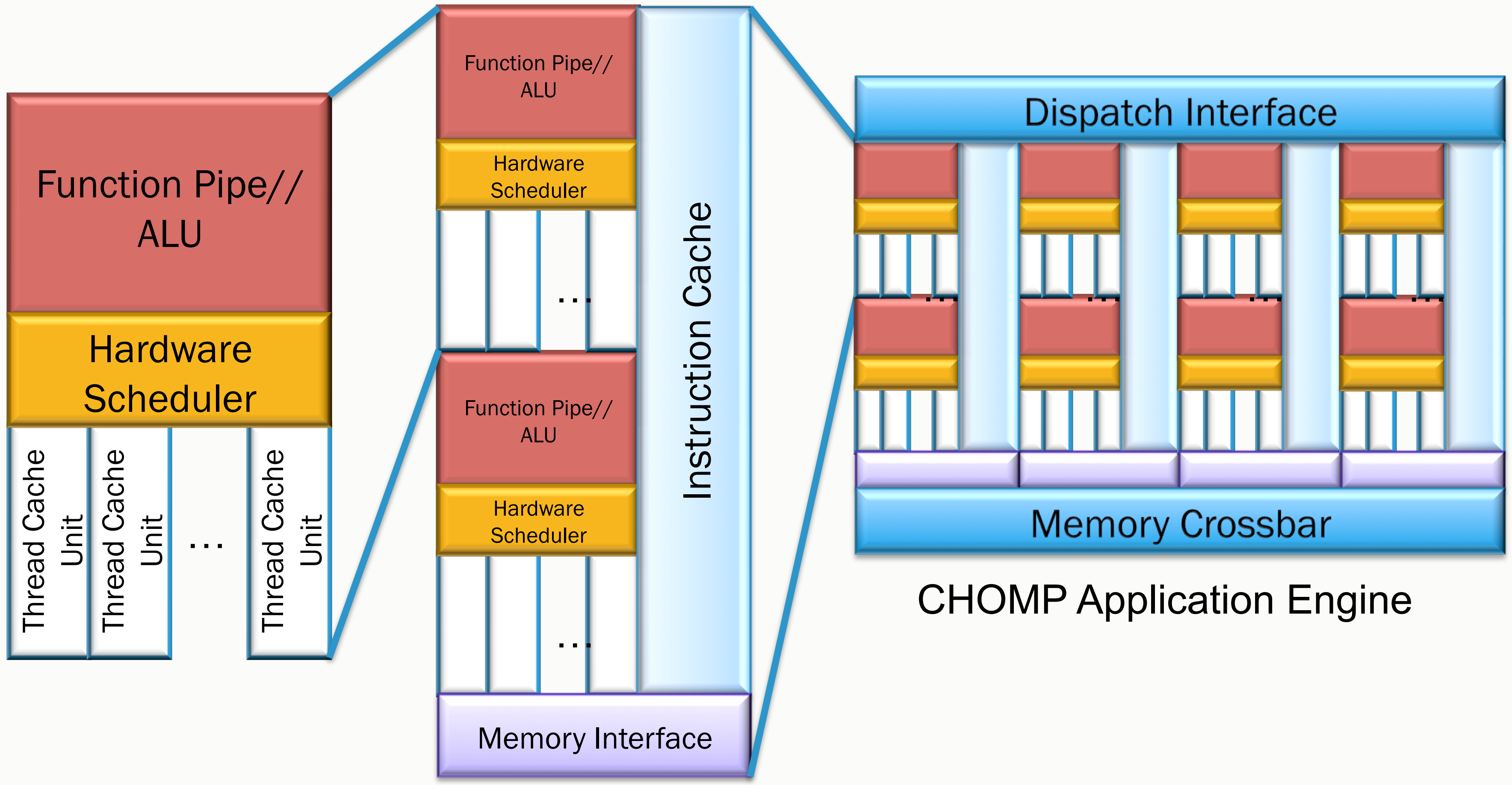
CHOMP: Convey Hybrid OpenMP

- **Scalable, MIMD Personality Framework**
  - Simple, RISC-like instruction set
  - Atomic memory operations
  - Fine-grained synchronization using tag-bit semantics
  - Software driven, hardware-based low-latency thread/task scheduler
  - *Every instruction is treated as a first class citizen*
- **First Programming model is OpenMP 3.0**
  - Support for vanilla OpenMP code directives
  - Simple, portable parallelism
  - Convey has joined the OpenMP Architecture Review Board
- **Interest in other language constructs [DSL's]**

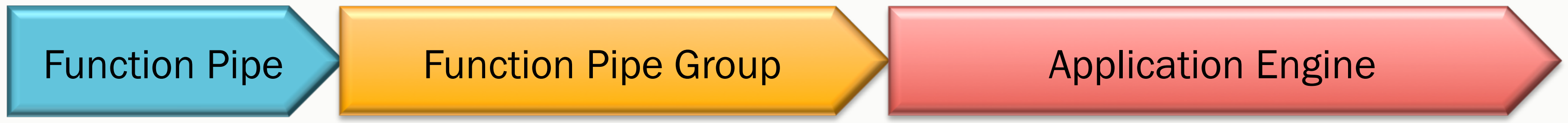
CHOMP Architecture Hierarchy



CHOMP Personality Infrastructure



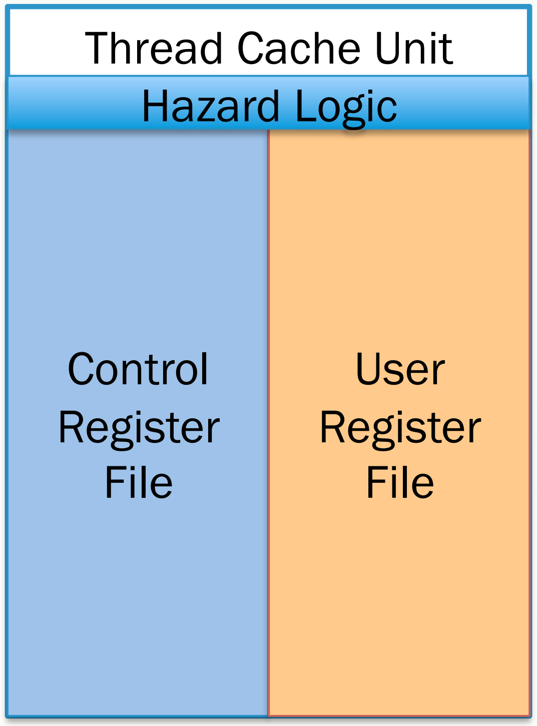
CHOMP Application Engine



CONVEY computer™

CHOMP Thread Cache Units

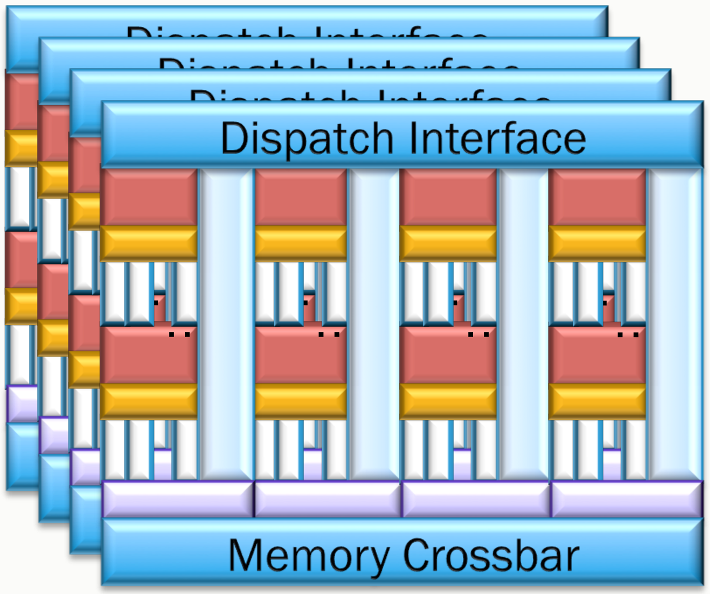
- **Smallest divisible unit of parallelism in hardware**
  - Control register file
  - User register file
- **A single TCU maps to some autonomous unit of software parallelism**
  - Thread, task, fiber, pebble, etc
  - In OpenMP, each TCU represents a thread
- **Scheduling decisions and context switching is performed on a TCU by TCU basis**
- **The current personalities have 64 TCU's per Function Pipe**
  - 63 are available to the user
  - 1 is used for the Workload Manager



CHOMP Function Pipe/Function Pipe Group

- **Function Pipes**
  - Contains arithmetic unit(s)
    - First personality design will contain:
      - Integer Mul, Add, Misc
      - Double Precision Floating Point Add, Mul
  - Multiple Thread Cache Units share a Function Pipe
    - The first personality contains 64 TCU's per FP
  - Workload Manager
    - Manages the scheduling on TCU's access to Function Pipe resources
- **Function Pipe Groups**
  - 8-way Set Associative Instruction Cache
    - ICACHE shared amongst all FP's+TCU's
  - Contains one or more Function Pipes
  - Memory interface to AE crossbar

CHOMP Personality Infrastructure



4 x CHOMP Application Engines per coprocessor

MX-100 DP Floating Point Personality

- 4 Application Engines per Coprocessor
- 12 Function Pipes per Application Engine
- 64 Threads per Function Pipe
- **3024 total threads per Coprocessor**

CONVEY computer™

CHOMP TCU Scheduling

- **Any time a Workload Manager finds empty TCU's, it will attempt to fetch them from the runtime work queues**
- **The Workload Manager will fetch:**
  - *MIN(Thread_Cache_Count, Empty_TCU_Count)*
  - The Thread Cache Count [TCC] values will affect the hardware's ability to naturally balance the load
  - The default TCC value is the number of TCU's per FP [in this case 64]

CHOMP TCU Scheduling cont.

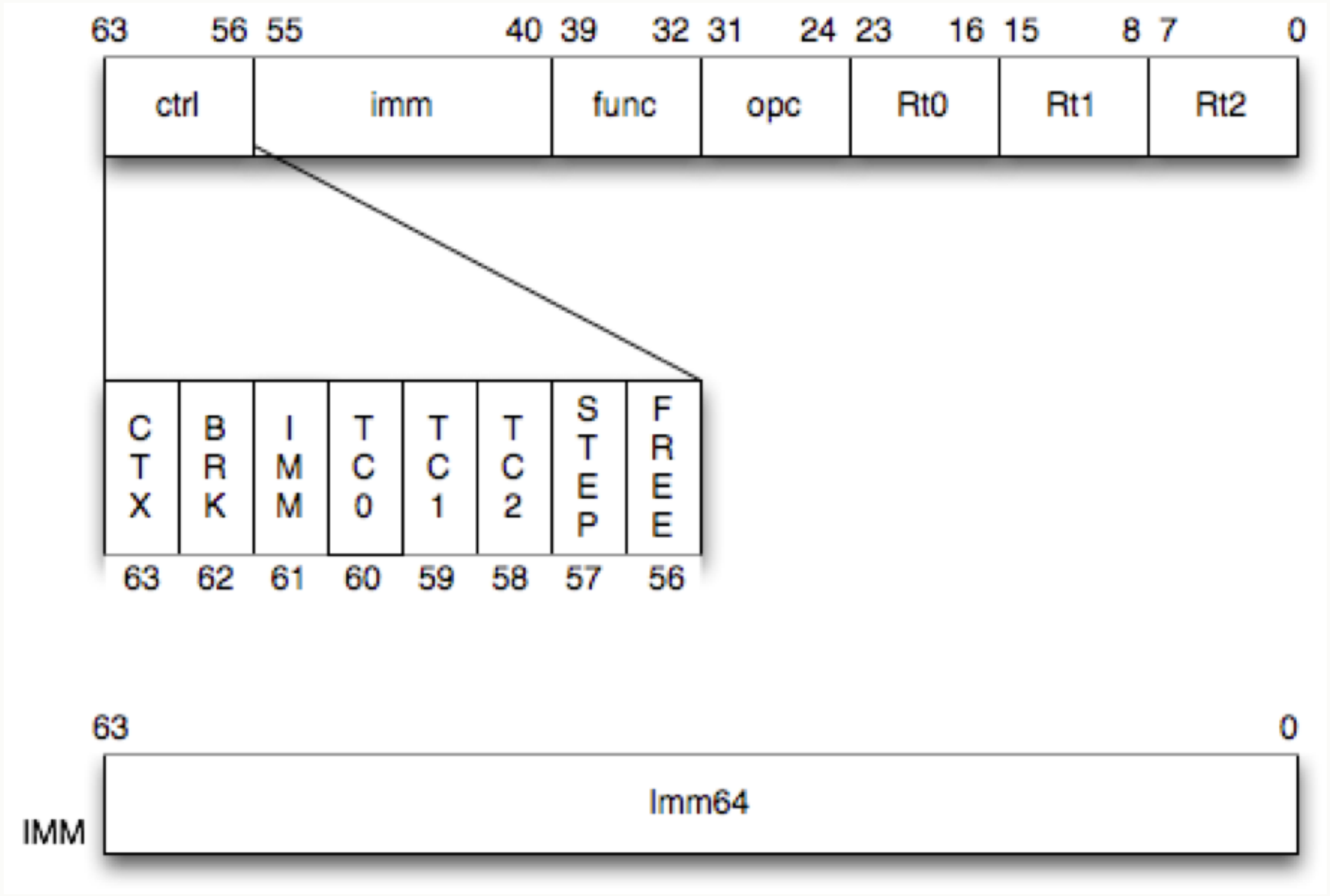
- **The hardware enforces a TDM round-robin policy on a single cycle between TCU's**
  - A minimum of 16 TCU's per FP must be active in order to not stall the Function Pipe
- **The hardware will not select TCU's for execution that are in the following state:**
  - Register hazarded
  - Waiting on an ICACHE miss to complete [fill]
  - Forcible context switch [via setting the context switch bit]
  - Fence [equivalent to setting the context switch bit]

CHOMP INSTRUCTION SET

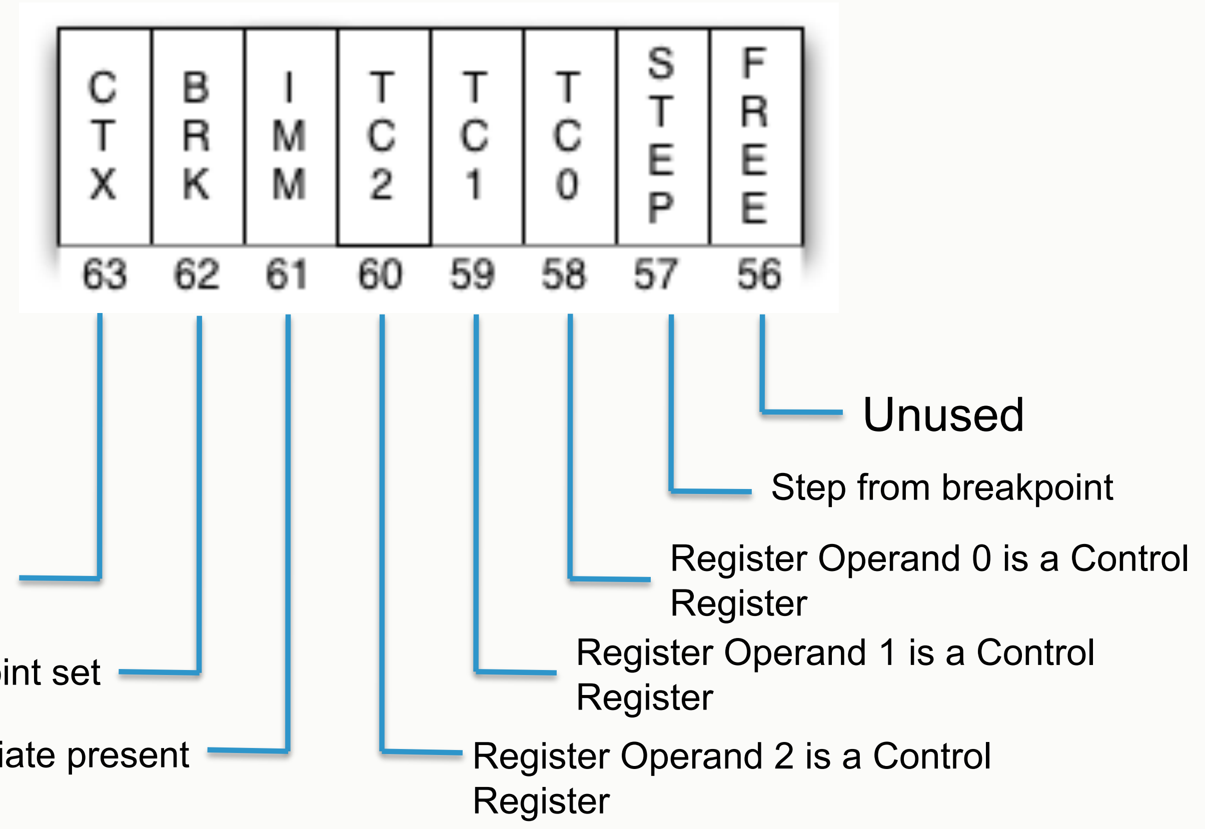
CHOMP Instruction Format

- **CHOMP Base ISA includes one RISC format**
  - Three eight-bit register operand fields
  - Opcode Field [8-bits]: defines the instruction “class”
  - Function Field [8-bits]: defines the instruction
  - 16-bit immediate field
  - 8-bit control field
  - Optional 64-bit immediate value in the next instruction word

CHOMP Instruction Format



CHOMP Instruction Format cont.


CTX
BRK
IMM
TC2
TC1
TC0
STEP
FREE
63
62
61
60
59
58
57
56
Unused
Step from breakpoint
Register Operand 0 is a Control Register
Register Operand 1 is a Control Register
Register Operand 2 is a Control Register
Context switch
Breakpoint set
64-bit immediate present

CHOMP Operation Classes

Operation Code	Function	Required
0x00	Load/Store	Yes
0x01	Arith-Misc	Yes
0x02	Arith-Integer	Yes
0x03	Arith-Float	No
0x04	Arith-UDEF	No
0x05	Arith-Flow Ctrl	Yes
0x06	Arith-Atomic Full/ Empty	Yes*
0x07	Arith-Thread Ctrl	Yes

- **User-Defined Arith Operation Class**
  - Permits customer architects to define their own arithmetic instructions
  - Zero, one, two, three operand arith's with predefined function codes
  - All user-defined arith's obey the standard context-switch and hazard mechanisms
  - Ability to attach these user-defined instructions to user-defined performance counters

*required for the workload manager

CONVEY computer™

CONVEY computer™

Application Examples

CHOMP Code Example

```
#pragma cny coproc {
#pragma omp parallel for shared(p) private(tmp,j,k)
    for( i=0; i<NUM_PAGES; i++ )
     {
      /* accumulate PR of incoming links */
         for( j=0; j<p[i].ni; j++ )
         {
             k = p[i].in[j];
             tmp+= ( p[k].rank/p[k].no  );
         }

         /* normalize the PR */
         p[i].rank = (1-DAMP)+DAMP * tmp;
     }
}
```

Spawns 3K+ lightweight threads

Spawn N threads for N cores

Load all incoming page ranks for adjacency arrays. Very high cache miss rate.

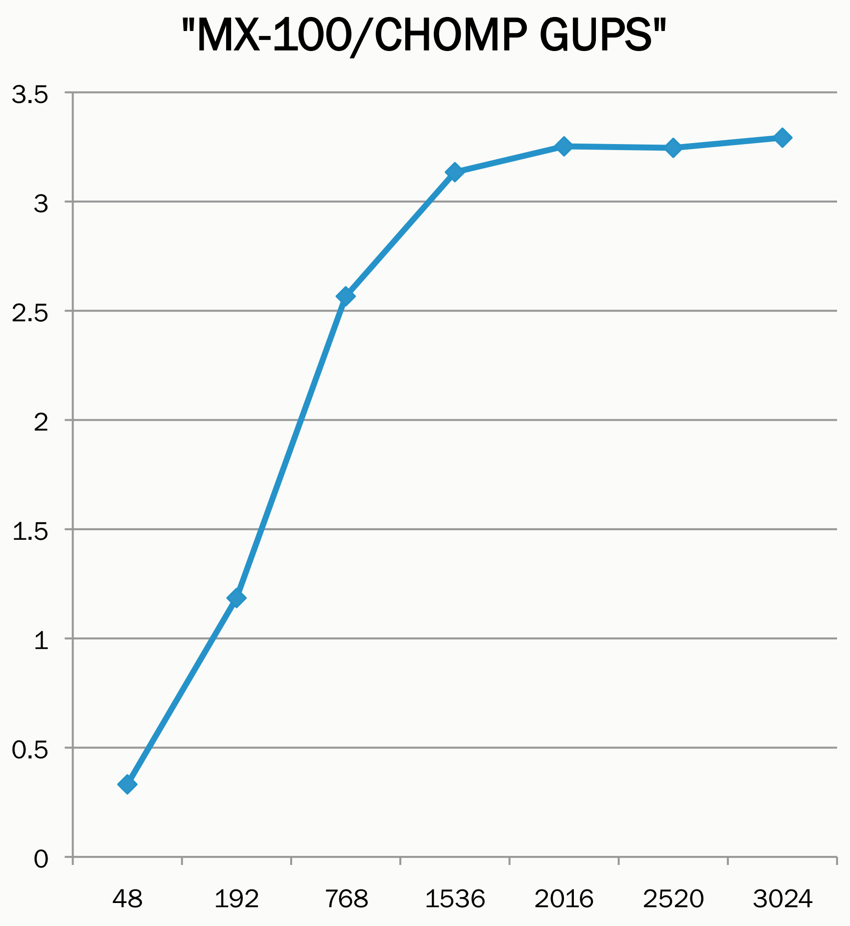
Normalize Result

CONVEY computer™

Pointer Chasing Example

- **Pointer chasing**
  - Multi-source graph searches
  - Multi-source shortest path searches
  - Vertex coloring
  - Some community detection algorithms

```
//-- Parallel for N starting points
cur = *start;
for( i=0; i<iters; i++ ) {
        visited[i] = cur;
        cur = cur->next;
}
```



*benchmark represents 65K vertices per thread; vertices randomized using LCG

Acknowledgements

- **Co-authors**
  - Kevin Wadleigh
  - Joe Bolding
  - Tony Brewer
  - Dean Walker
- **RTL Team**
  - Dean Walker
  - Mike D’Jamoos
  - John Amelio
  - Ryan Akkerman
  - Mike Dugan
- **MX-100 Platform Team**
- **Compiler Team**
  - Daniel Palermo
  - Geoff Rogers
  - Jason Eckhart
  - Randy Meyer
  - Rich Bleikamp
  - Mike Carl

Questions/Comments?

jleidel<at>conveycomputer<dot>com

THE WORLD'S FIRST HYBRID-CORE COMPUTER.

CONVEY computer™