NanoMesh:
An Asynchronous Kilo-Core System-on-Chip

Jonathan Tse [1] Andrew Lines [2]

[1]Computer Systems Laboratory, Cornell University

[2]Networking Division, Intel Corporation

May 20, 2013

Context

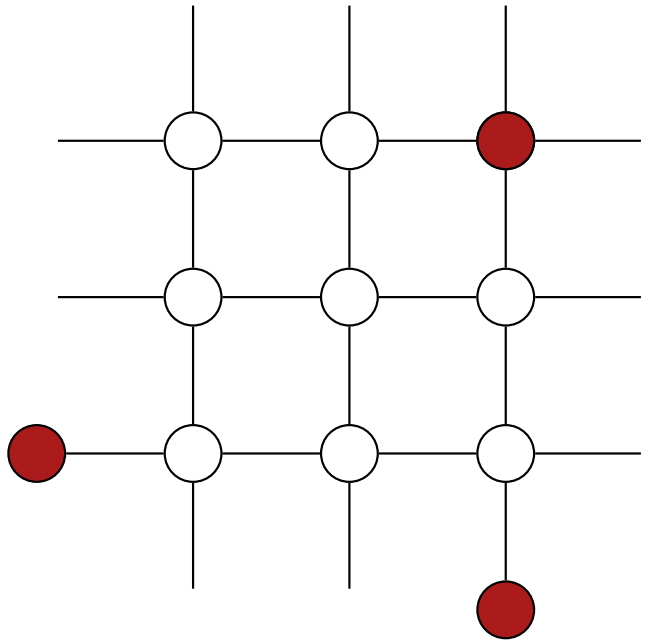
- Ever-increasing demand for performance
- Tighter design constraints and power envelopes
- Efficient use of transistors:
  - More cores
  - Specialization/Accelerators
  - Systems-on-Chip (SoC) for tight integration
- Time-to-Market

SoC Challenges

- The quest for parallelism
  - Workloads
  - Programming model
  - Programmer vs System Architecture
- On-die connectivity
  - Networks-on-Chip (NoC)
  - Topology, routing, flow-control
- Intra-die synchronization
  - Clock distribution
  - Globally Asynchronous, Locally Synchronous (GALS)
  - Fully-asynchronous

NanoMesh SoC Framework

- Lightweight Cores
  - General-purpose compute
  - Push work to cores
  - Instantiate as necessary
- SoC Interconnect
  - Scalable interconnect
  - Multiple protocols
  - Connect IP instances

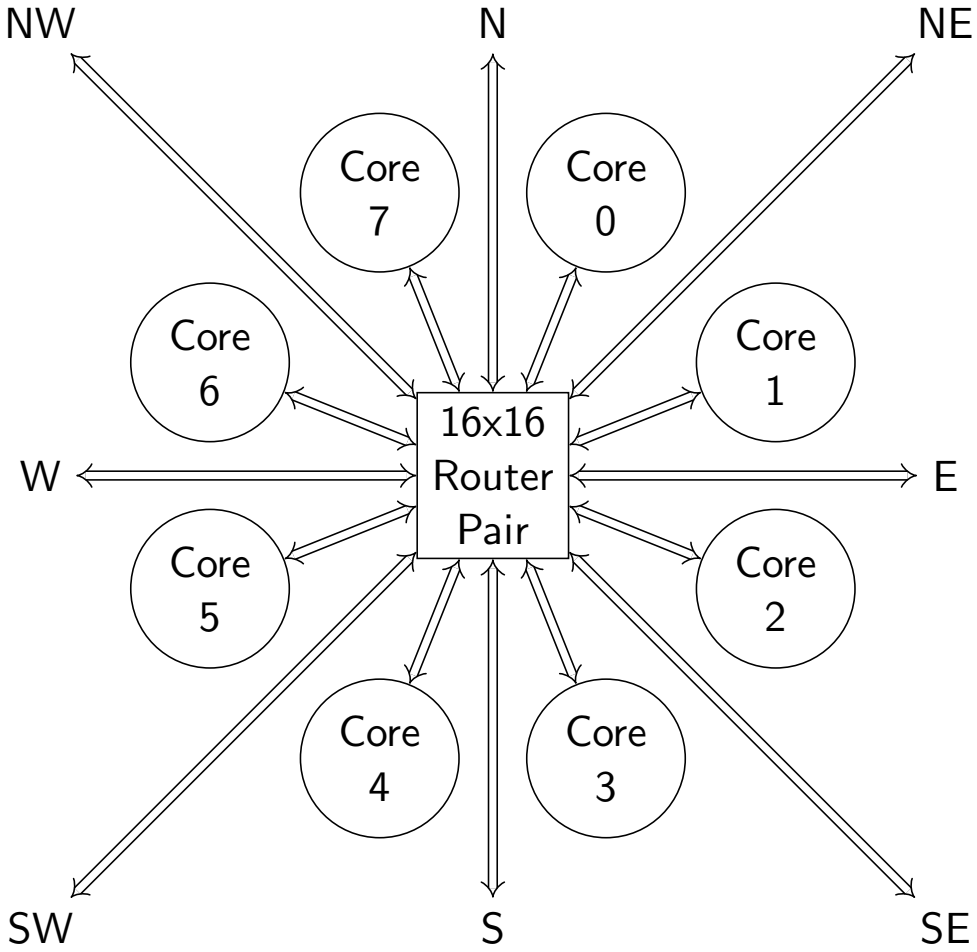


System Overview

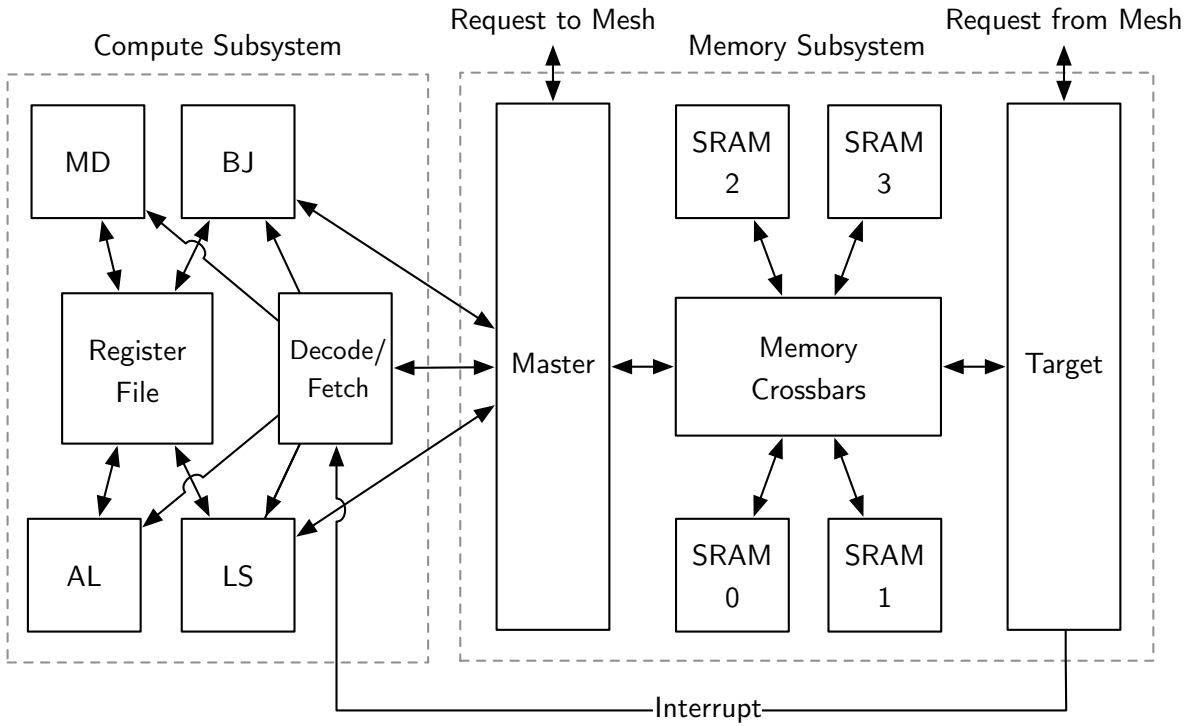
- **Simplicity**
- Proof-of-concept
- Parallelism
  - Shared-memory programming model
  - Programmer-driven parallelism
- Mesh NoC for Connectivity
  - Crossbar-based router
  - 2 parallel mesh networks for Request/Response
  - Dimension-order routing
  - Flow-control with asynchronous handshakes
- Fully QDI Asynchronous Design
  - Easy modularity at channel-boundaries
  - Inbuilt flow control
  - Inbuilt buffering
  - Fine-grained clock gating

NanoMesh Overview

- Mesh of Tiles
  - 16 x 16 Mesh
  - 2048 IP Instances
  - 188 at Boundary
- 8 IP Instances (Cores)
- Cardinal Links
- Diagonal Links
  - Lower latency
  - More bandwidth

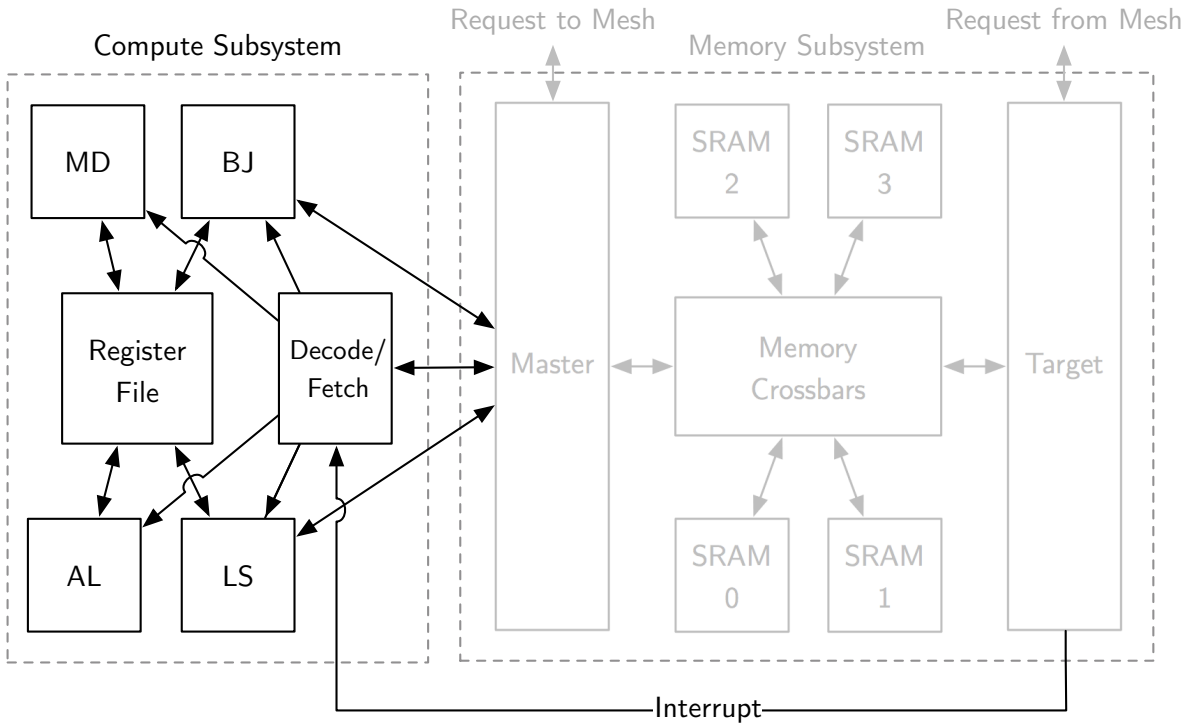


NanoMIPS Core



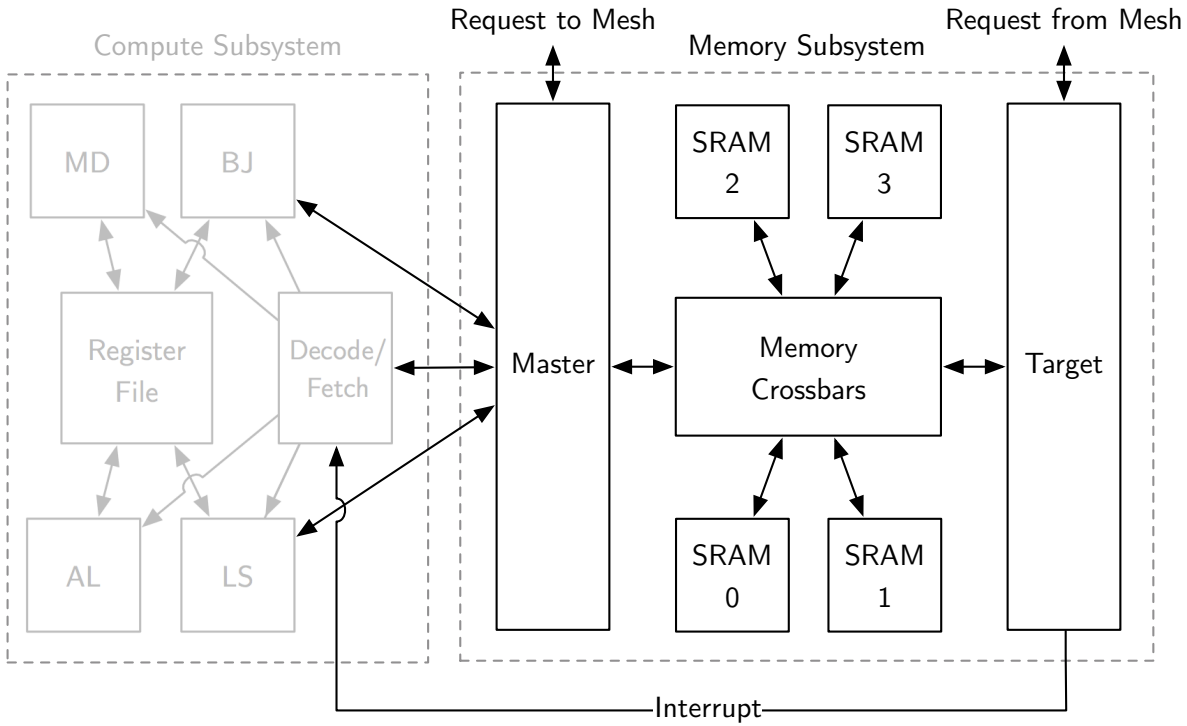
- Single-Issue In-Order 32-bit MIPS Core
  - Removed some instructions (unaligned loads/stores)
  - Added instructions to support multi-core communication
- CSP Specification

NanoMIPS Compute Subsystem



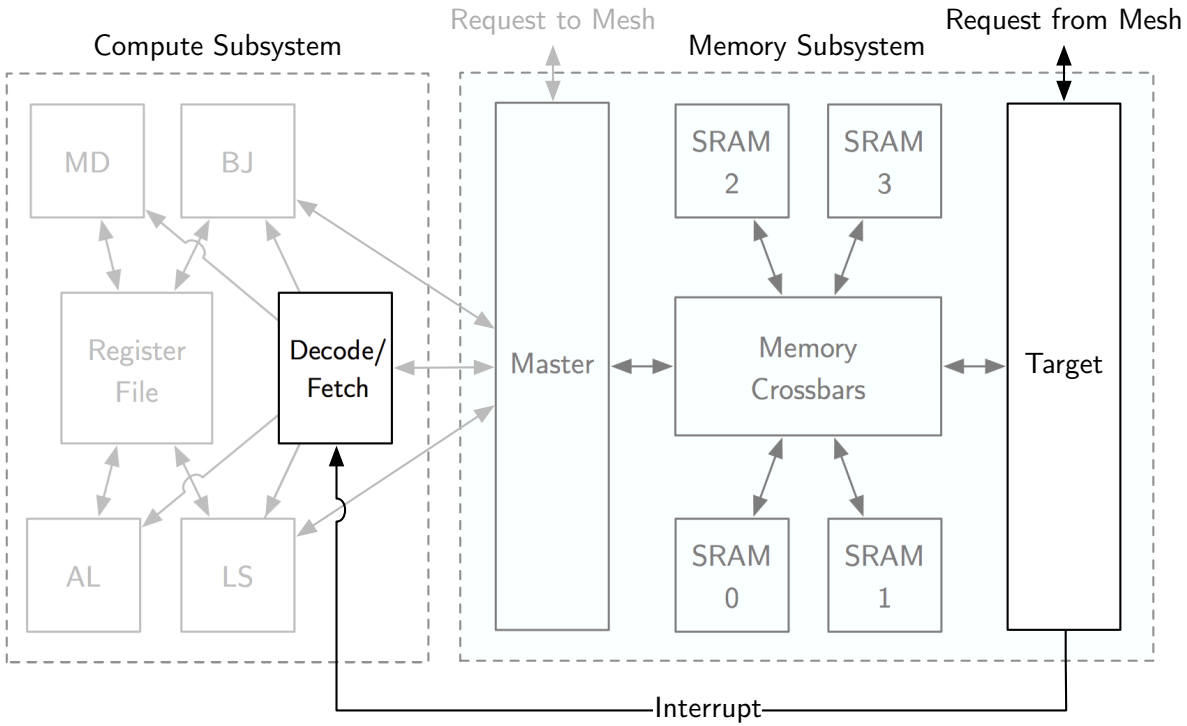
- Simple exception/interrupt handler (No OS support)
- New `WAIT` and `HALT` instructions stall fetch

NanoMIPS Memory Subsystem



- 64KB SRAM, no cache, no explicit Instr/Data memory
  - Full R/W access to every core's SRAM
  - Multiple access modes (contiguous and striped)
- Memory subsystem independent of compute
- New Memcopy Instructions (copy 4 to 16 words)

Shared-Memory Message Passing



- Write to flagged memory address triggers Interrupt channel
  - Memory Dirty — Clears `WAIT`
  - Interrupt — Clears `HALT`
- Supports all memory access modes
- Allows for rudimentary message passing

Sample NanoMesh Execution

Core A — Control

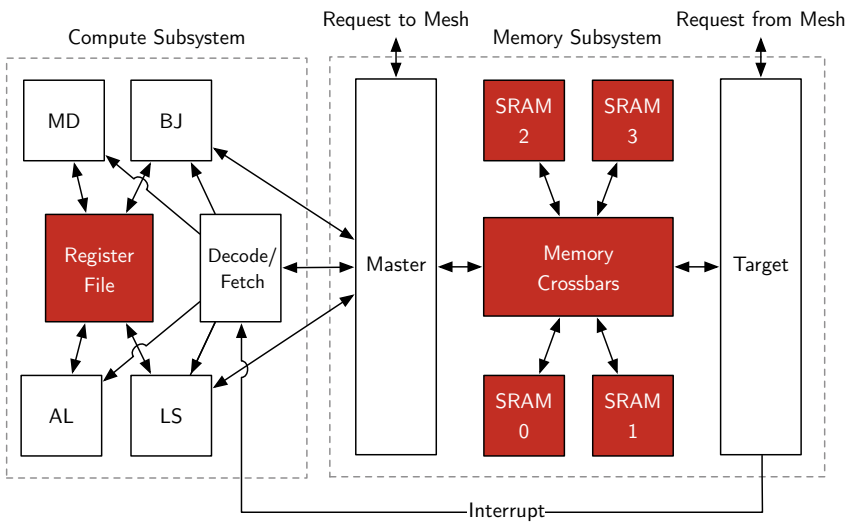
1. Write Program to Core B
2. Fire `Interrupt` on Core B
3. Send data to Core B
4. `WAIT`
5. Continue execution

Core B — Worker

1. `HALT`
2. Initialize
3. `WAIT`
4. Do work on data
5. Send result to Core A
6. `HALT`

Building NanoMesh

- Mesh Network
  - Full Custom
  - Heavily Templated
- NanoMIPS
  - Some Full-Custom Design (Memories, Crossbars, etc)
  - Rest is synthesized using Proteus[1] (CSP to Standard Cell)
  - Some CSP-level decomposition for efficiency



[1] Beerel, et al. "Proteus: An ASIC Flow for GHz Asynchronous Designs." IEEE Design & Test of Computers, 2011.

NanoMesh Characterization

- From CSP to P&R in 3–4 man-months
- Early TSMC 28nm HPM simulation results are encouraging
- Further optimization will improve numbers

Simulation Results

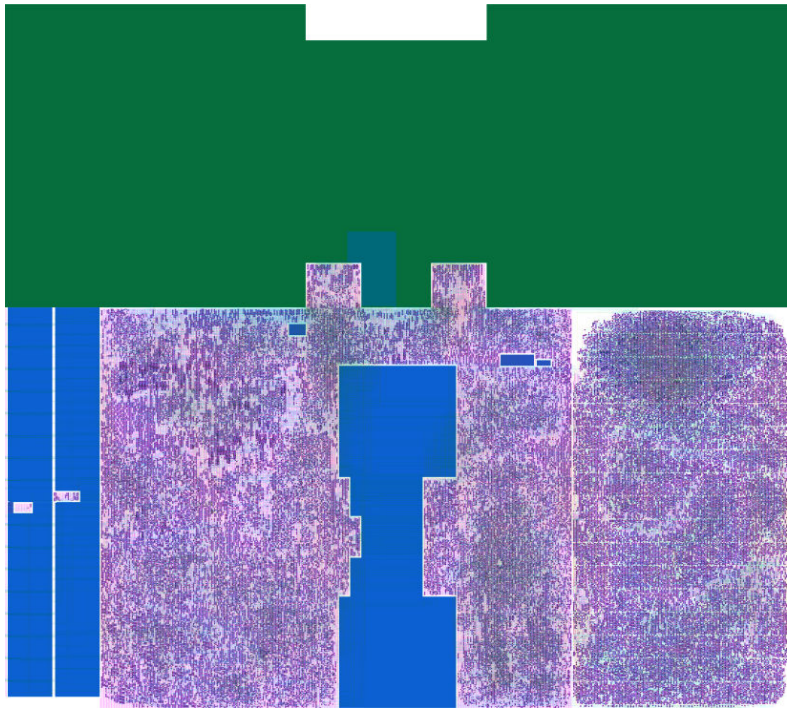
Benchmark	TPC	Score	Perfect R3000
Dhrystone 2.1	45.5	370	1073
Inline Dhrystone 2.1	47.2	482	1317
CoreMark 1.01	45.5	706	2140
`NOP`	26.2		
`ADDU`	26.2		
`SW`	38.9		
`LW`	79.9		

SPICE Results

Unit		PVT		Frequency [MHz]	Energy [pJ]
AL	TT	0.90V	50C	1467	59
AL	SS	0.81V	0C	770	45
Router	TT	0.90V	50C	2010	16
Router	SS	0.81V	0C	1317	12
Regfile	TT	0.90V	50C	2216	31
Regfile	SS	0.81V	0C	1171	24

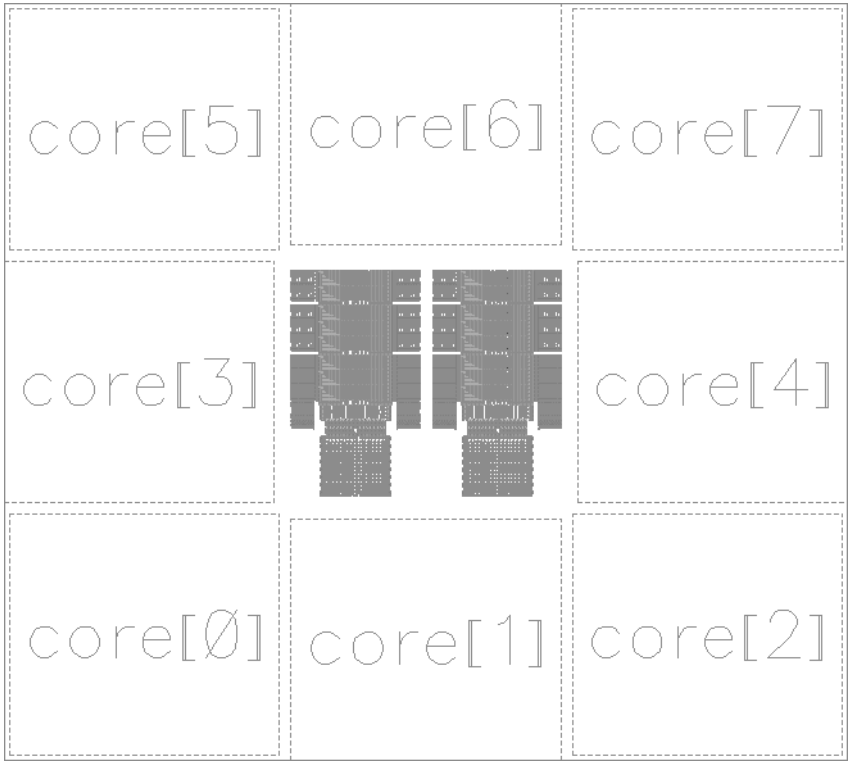
Place and Route

Unit	TPC	Gates	Area [mm^2]
MD	24	30915	0.064
CORE	26	41457	0.108
Total		72372	0.680



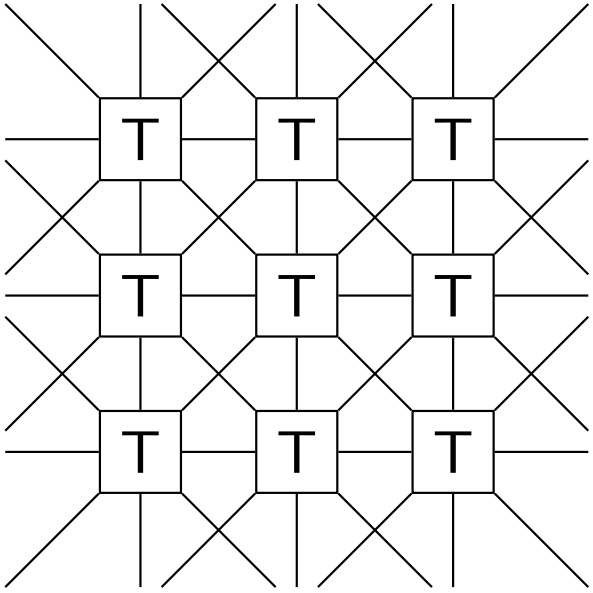
Projected NanoMesh Size

- Tile is 6.8mm^2
- A 64-tile, 512-core system is 435mm^2 + boundary I/O
- 1024-core system is feasible in 22nm



NanoMesh Summary

- Flexible SoC Framework
- Tiles of Lightweight Cores
  - Scalable processing
  - Shared memory
  - Processor-in-memory
- Interconnect
  - Connect what you need
  - Flexible design



NanoMesh:
An Asynchronous Kilo-Core System-on-Chip

Jonathan Tse [1] Andrew Lines [2]

[1]Computer Systems Laboratory, Cornell University

[2]Networking Division, Intel Corporation

May 20, 2013

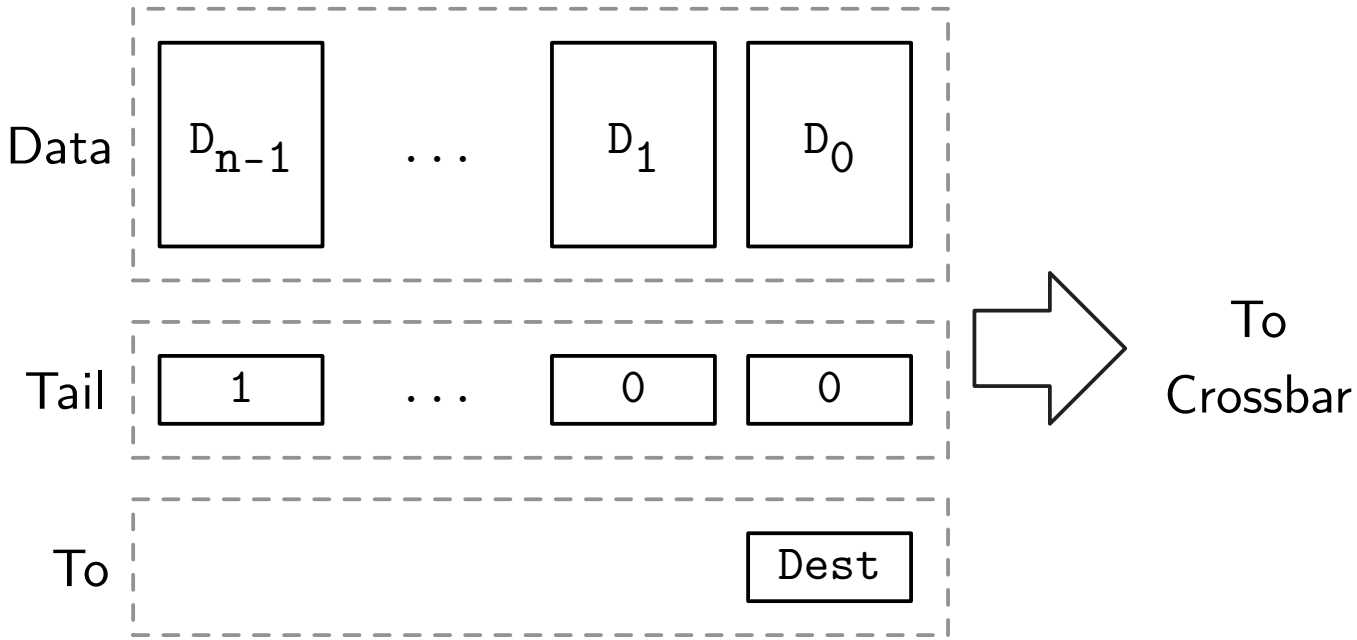
Caveats

- TPC target is 22, but we currently get 26. Earlier runs achieved 18, so this should be fixable after some debugging.
- Post-layout Proteus frequency is slightly lower than Encounter timing. We should be getting $\leq$ 909MHz in slow corner. Will fix with better cap table and calibration.
- Lack of branch prediction causes about 30% of instructions in Dhrystone and CoreMark to be cancelled.
- Conflicts and coupling between code and data memory access. Uncertain impact.
- No latency constraints for synthesis yet, relaxed latency constraints for P&R.

Future Work

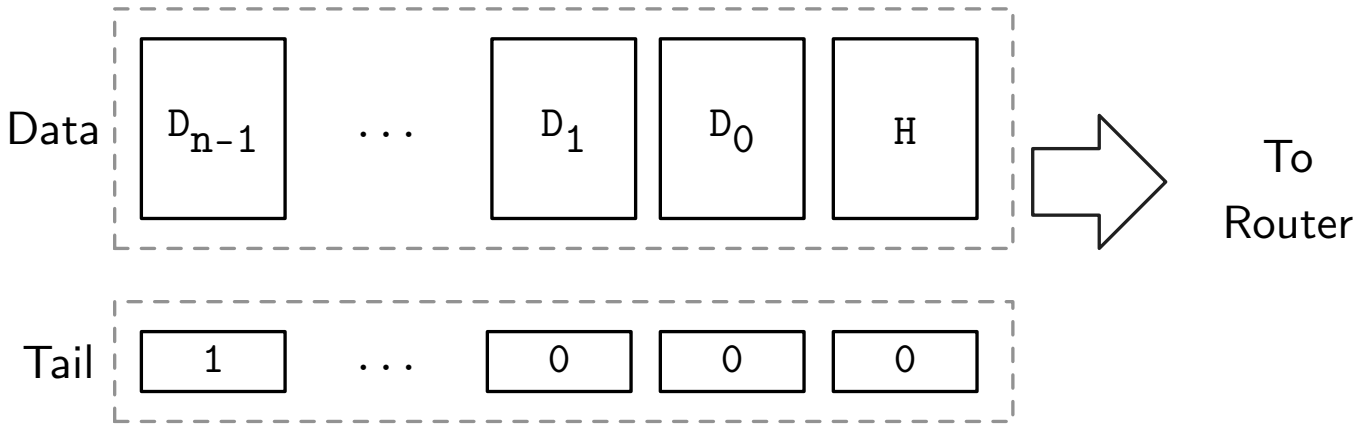
- Implement a branch predictor!
- Split memory into 32KB code and 32KB data for local access, while still 64KB for remote access. This will simplify memory system, avoid bank conflicts and latency interactions between code and data. Less flexibility.
- Simplify hard macros to the minimum essentials. Will replace custom interrupt cells with Proteus. Probably break up REG into a 16-bit BYPASS and core RRW, letting Proteus distribute control signals.
- Finish all Proteus cells, characterize real timing, debug and improve flow, close timing.

Nexus Crossbar



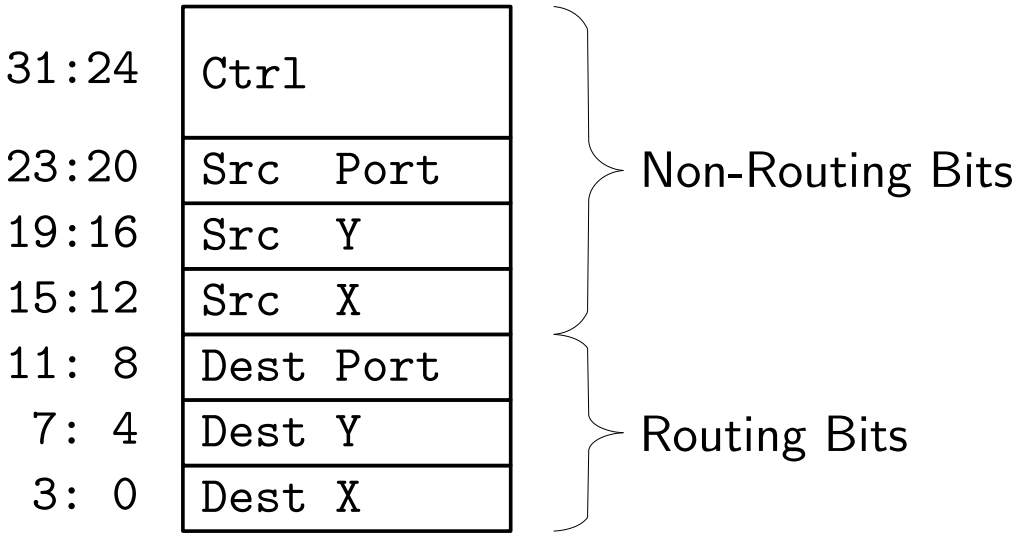
- 16x16 Full-Duplex
- 32-bit Wide Datapath
- Tail-terminated Flits
- Arbitration on output for whole flit

Nexus Mesh Router



- Preface messages with 32-bit `Header` word
- Calculate `To` field from `Header`
- `Header`-only message possible by setting `Tail` to 1

Nexus Mesh Header Word



- 4-bit (X,Y) Tile Coordinates (Dimension Order Routed)
- 4-bit Port Address (4-bit for Boundary Ports)
- 8-bit User-Defined Control Field
- Trivial Return addressing (swap source/dest fields)

NanoMIPS ISA

Unit	R3000	New	
DEC	NOP RFE SYSCALL	HALT WAIT	
AL	ADDU ADDIU SUBU AND ANDI OR ORI NOR XOR XORI LUI SLL SLLV SRA SRL SRLV SLT SLTI SLTU SLTIU		
MD	MULT MULTU DIV DIVU MFHI MFLO MTHI MTLO		
BJ	BEQ BGEZ BGTZ BLEZ BLTZ BNE BGEZAL BLTZAL J JAL JALR JR MFC0 MTC0		
LS	LB LBU LH LHU LW SB SH SW	CF4 CT4	CF16 CT16
Omitted	ADD ADDI SUB LWL LWR SWL SWR		

New NanoMIPS Instructions

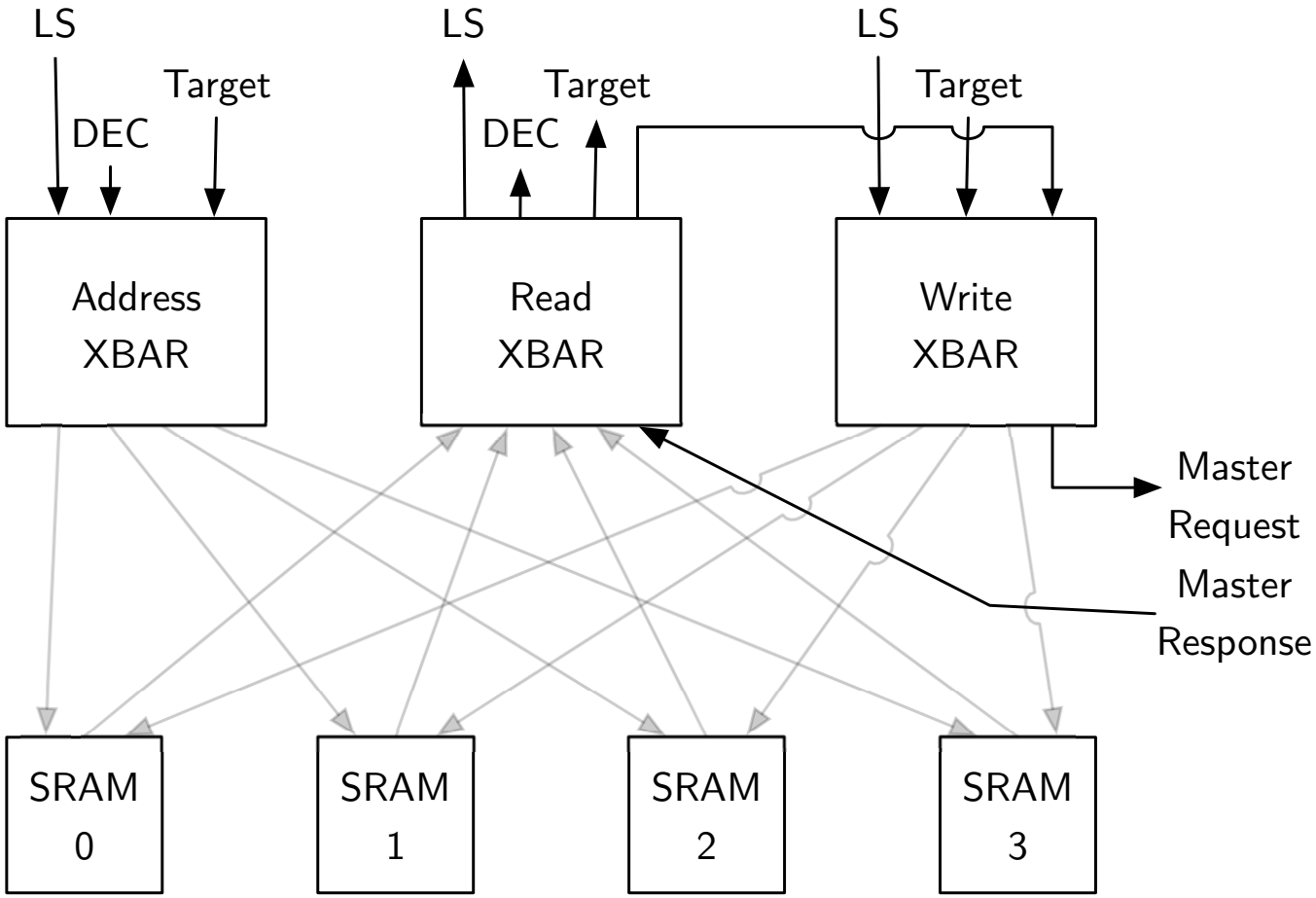
- `WAIT` — Stalls fetch until dirty memory flag set
  - Prevents busy-waiting
  - Allows for rudimentary message-passing constructs
- `HALT` — Stalls instruction fetch until `Interrupt`
  - NanoMIPS cores bootstrap into `HALT` state
  - Exposes clock-gating like behavior directly to the programmer
- Memcopy Instructions
  - `CF4, CF16` — Copies 4 or 16 words from local memory to remote memory address.
  - `CT4, CT16` — Copies 4 or 16 words from remote memory to local memory.
  - Addresses must be block-size aligned.
  - Supports all addressing modes
  - Supports dirty memory flag setting

NanoMIPS Addressing Modes

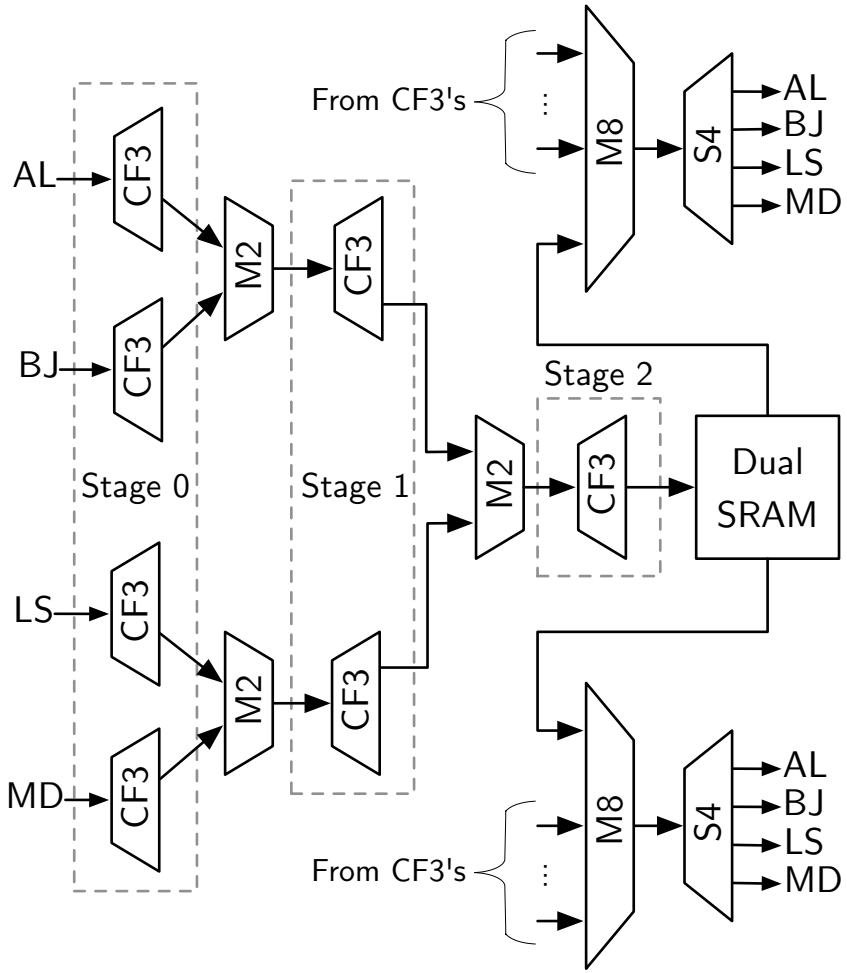
- Addressing mode selected with higher-order bit
- Mesh network routing bits pulled from appropriate memory address

Mode	Addressable Memory	Striping
Local	64KB (Local)	
Remote	64KB (Remote)	
Tile	512KB/Tile	64B within Tile
Global	128MB (2048 cores)	64B across Tiles

NanoMIPS Memory Datapath



NanoMIPS Register File Bypass



Toolchain

- Supports C programs
- `gcc` targeted for MIPS
- New instructions supported by `gcc` inline assembly calls
- Custom memory-mapped `printf()` implementation
- Custom memory-mapped `time()` implementation

Implementation Overview

- Full-Custom Template Library
  - Oft-used/regular cells are templated
  - Sized using estimated diffcap and wirecap
- Proteus Flow
  - CSP to standard cell flow
  - Targets CSP to set of asynchronous standard cells
  - Places and routes cells
  - Typically used for control and sequencing circuits

Full-Custom Design

- Mesh Network
  - Nexus Crossbar
  - Routing logic
  - Mesh interfaces in Master/Target
- NanoMIPS
  - Register file
  - Register file bypass
  - Memory system datapath/crossbars
  - Interrupt channel probe
  - Dual arbiter for Master/Target arbitration