The Design and Implementation of the TRIPS Prototype Chip

Robert McDonald,
Doug Burger, Steve Keckler

HotChips 17, August 2005

Computer Architecture and Technology Laboratory
Department of Computer Sciences
The University of Texas at Austin

TRIPS Project Goals

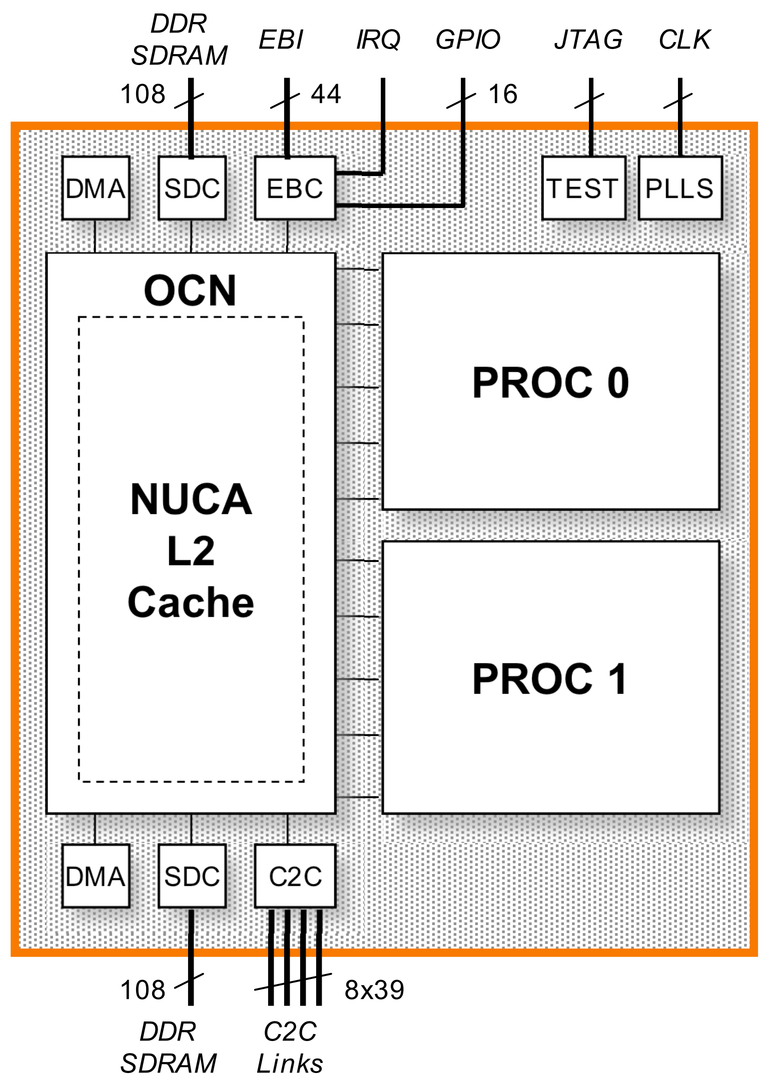
- Technology scalable processor and memory architectures
  - Techniques to scale to 35nm and beyond
  - Enable high clock rates if desired
  - High design productivity through replication
- Good performance across diverse workloads
  - Exploit instruction, thread, and data level parallelism
  - Work with standard programming models
- Power-efficient instruction level parallelism
- Demonstrate via custom hardware prototype
  - Implement with small design team
  - Evaluate, identify bottlenecks, tune microarchitecture

Key Features

- EDGE ISA
  - Block-oriented instruction set architecture
  - Helps reduce bottlenecks and expose ILP
- Tiled Microarchitecture
  - Modular design
  - No global wires
- TRIPS Processor
  - Distributed processor design
  - Dataflow graph execution engine
- NUCA L2 Cache
  - Distributed cache design

TRIPS Chip

- 2 TRIPS Processors
- NUCA L2 Cache
  - 1 MB, 16 banks
- On-Chip Network (OCN)
  - 2D mesh network
  - Replaces on-chip bus
- Misc Controllers
  - 2 DDR SDRAM controllers
  - 2 DMA controllers
  - External bus controller
  - C2C network controller



TRIPS Processor

- Want an aggressive, general-purpose processor
  - Up to 16 instructions per cycle
  - Up to 4 loads and stores per cycle
  - Up to 64 outstanding L1 data cache misses
  - Up to 1024 dynamically executed instructions
  - Up to 4 simultaneous multithreading (SMT) threads
- But existing microarchitectures don’t scale well
  - Structures become large, multi-ported, and slow
  - Lots of overhead to convert from sequential instruction semantics
  - Vulnerable to speculation hazards
- TRIPS introduces a new microarchitecture and ISA

EDGE ISA

- Explicit Data Graph Execution (EDGE)
- Block-Oriented
  - Atomically fetch, execute, and commit whole blocks of instructions
  - Programs are partitioned into blocks
  - Each block holds *dozens* of instructions
  - *Sequential* execution semantics at the block level
  - *Dataflow* execution semantics inside each block
- Direct Target Encoding
  - Encode instructions so that results go directly to the instruction(s) that will consume them
  - No need to go through centralized register file and rename logic

RISC

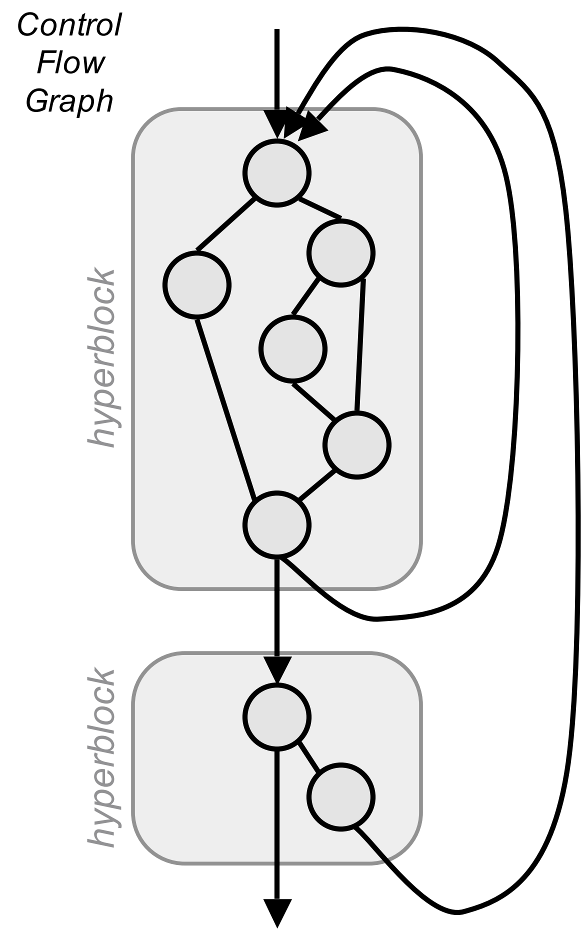
```
LD R1, 8(R0)
ADDI R1, 1
SD R1, 8(R0)
```

EDGE

```
R[0] READ N[0,0] N[2,0]
N[0] LD 8 N[1,0]
N[1] ADDI 1 N[2,1]
N[2] SD 8
```

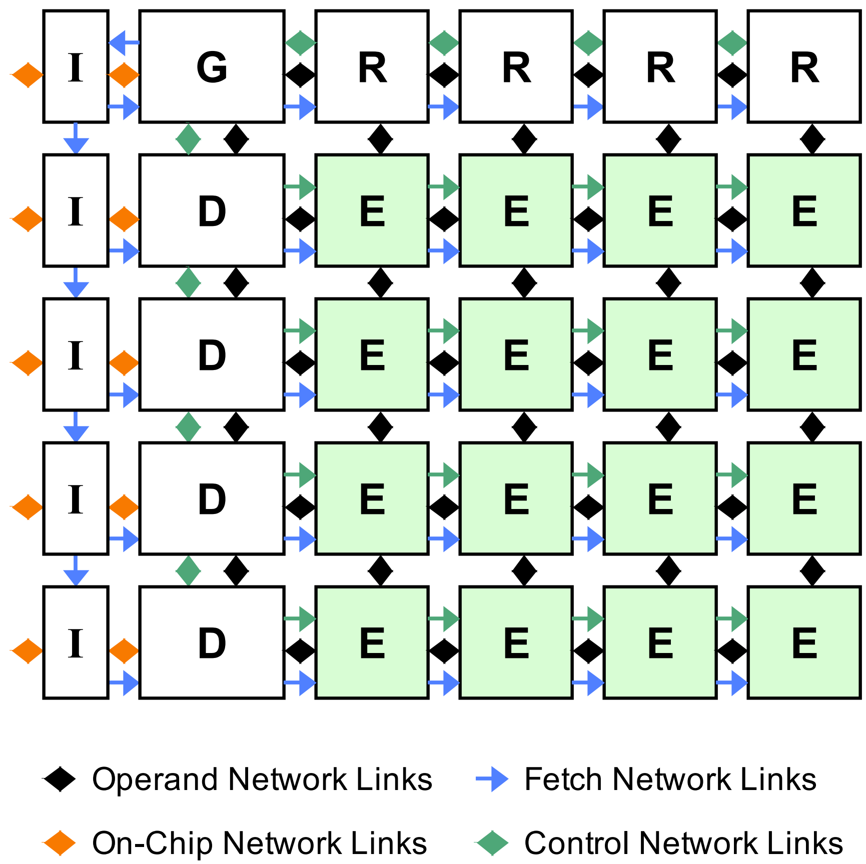
Block Formation

- Basic blocks are often too small (just a few insts)
- Predication allows larger *hyperblocks* to be created
- Loop unrolling and function inlining also help
- TRIPS blocks can hold up to 128 instructions
- Large blocks improve fetch bandwidth and expose ILP
- Hard-to-predict branches can sometimes be hidden inside a hyperblock



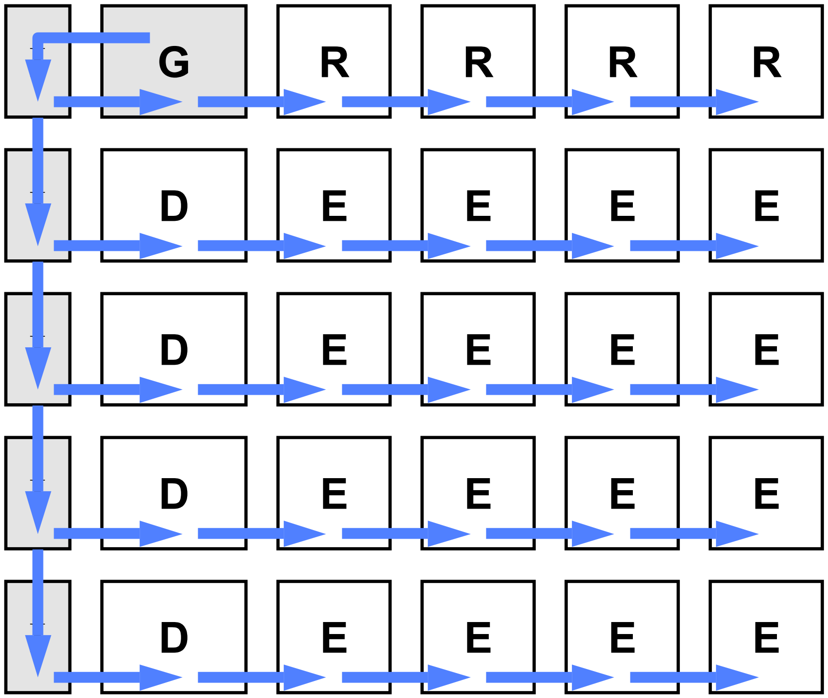
Processor Tiles

- Partition all major structures into banks, distribute, and interconnect
- Execution Tile (E)
  - 64-entry Instruction Queue bank
  - Single-issue execute pipeline
- Register Tile (R)
  - 32-entry Register bank (per thread)
- Data Tile (D)
  - 8KB Data Cache bank
  - LSQ and MHU banks
- Instruction Tile (I)
  - 16KB Instruction Cache bank
- Global Control Tile (G)
  - Tracks up to 8 blocks of insts
  - Branch prediction & resolution logic



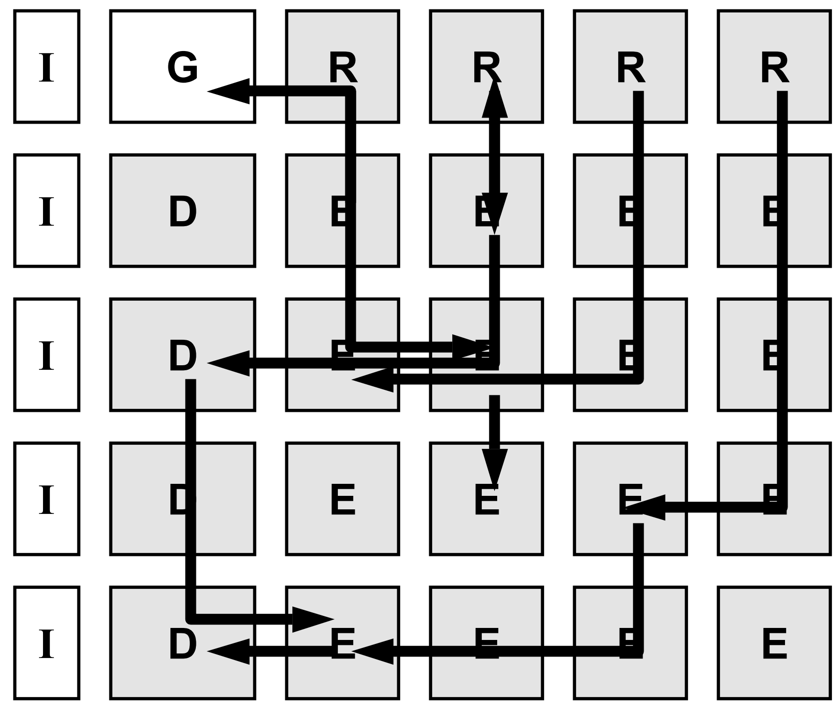
Block Fetch

- Fetch commands sent to each Instruction Cache bank
- The fetch pipeline is from 4 to 11 stages deep
- A new block fetch can be initiated every 8 cycles
- Instructions are fetched into Instruction Queue banks (chosen by the compiler)
- EDGE ISA allows instructions to be fetched out-of-order



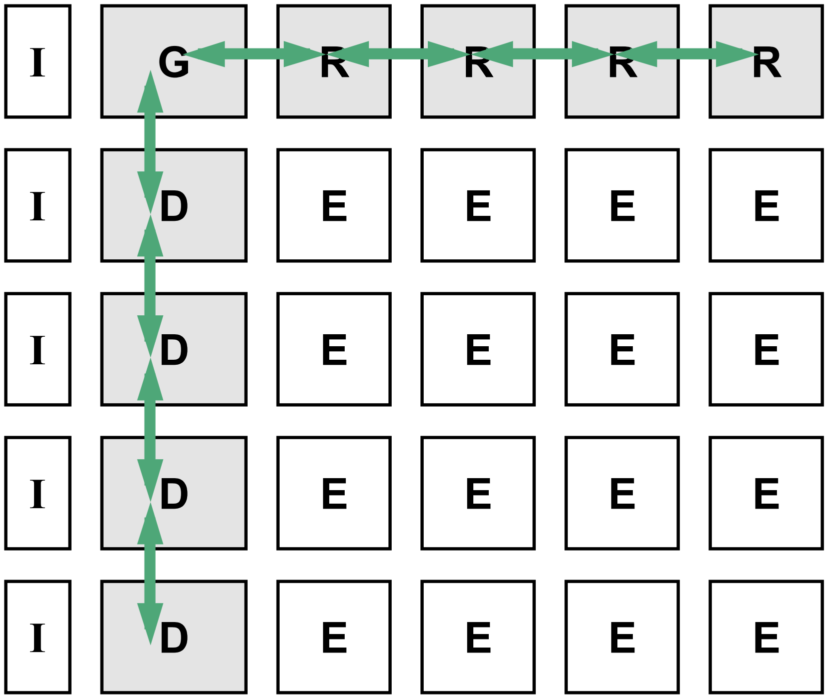
Block Execution

- Instructions execute (out-of-order) when all of their operands arrive
- Intermediate values are sent from instruction to instruction
- Register reads and writes access the register banks
- Loads and stores access the data cache banks
- Branch results go to the global controller
- Up to 8 blocks can execute simultaneously



Block Commit

- Block completion is detected and reported to the global controller
- If no exceptions occurred, the results may be committed
- Writes are committed to Register files
- Stores are committed to cache or memory
- Resources are deallocated after a commit acknowledgement



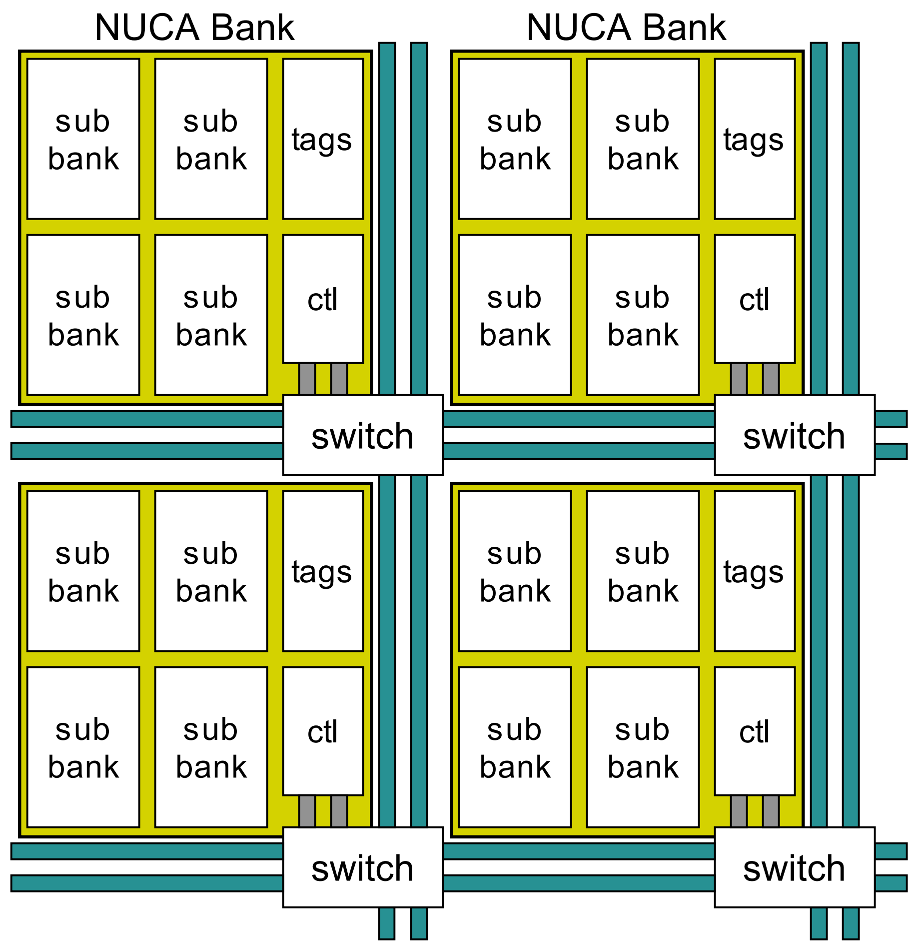
Processor Performance

Name	TRIPS Speedup	Alpha IPC	TRIPS IPC	TRIPS Inst/Block	Description
a2time	5.05	0.81	4.05	77	Control, integer math
bezier	3.30	1.05	3.20	76	Bezier curve, fixed-point math
dct8x8	2.66	1.70	4.70	90	2D discrete cosine transform
matrix	3.30	1.68	4.05	72	Matrix multiply
sha	0.92	2.28	2.10	80	Secure hash (mostly sequential algorithm)
vadd	1.92	3.04	6.51	74	Vector add (limited by load/store bandwidth)

Simulated on TRIPS and Alpha 21264 cycle simulators
Alpha compilation with GEM compiler and maximum opts (O4 and tuned for 21264)
TRIPS compilation with in-development compiler plus some hand-tuning
Speedup measured by comparing Alpha cycles to TRIPS cycles

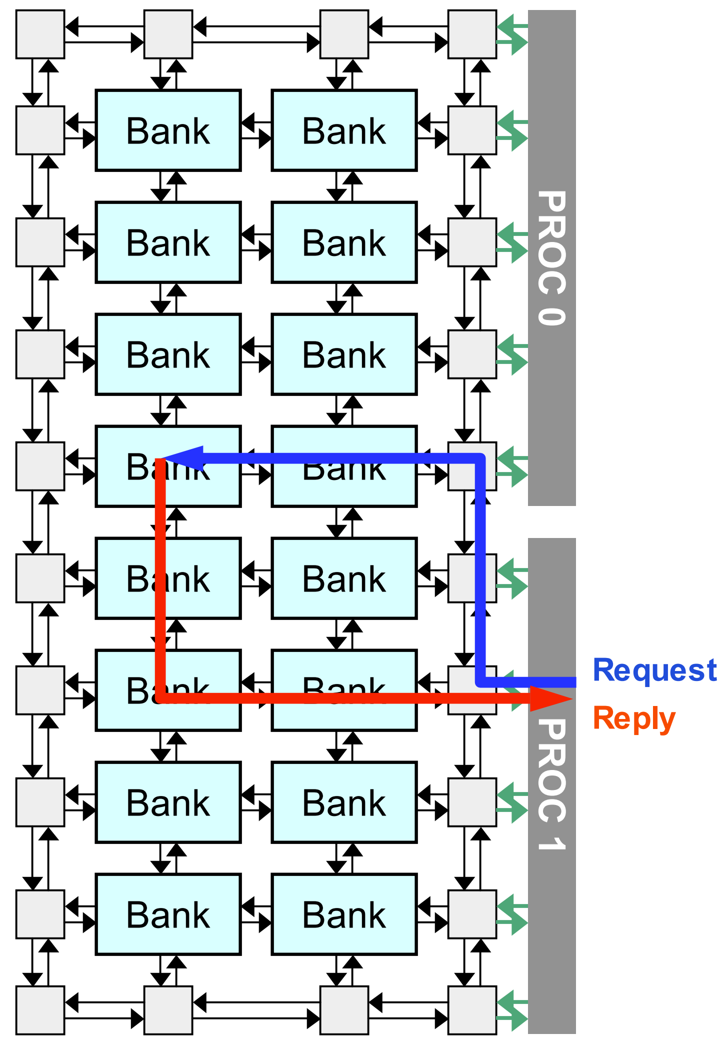
NUCA Concept

- Non-Uniform Cache Architecture (NUCA)
- Divide cache into small, fast banks
- Connect via switch network
- Interleave cache lines across banks
- Allows cache capacity and bandwidth to scale up
- Maintains high frequency and short wires
- Access latency varies



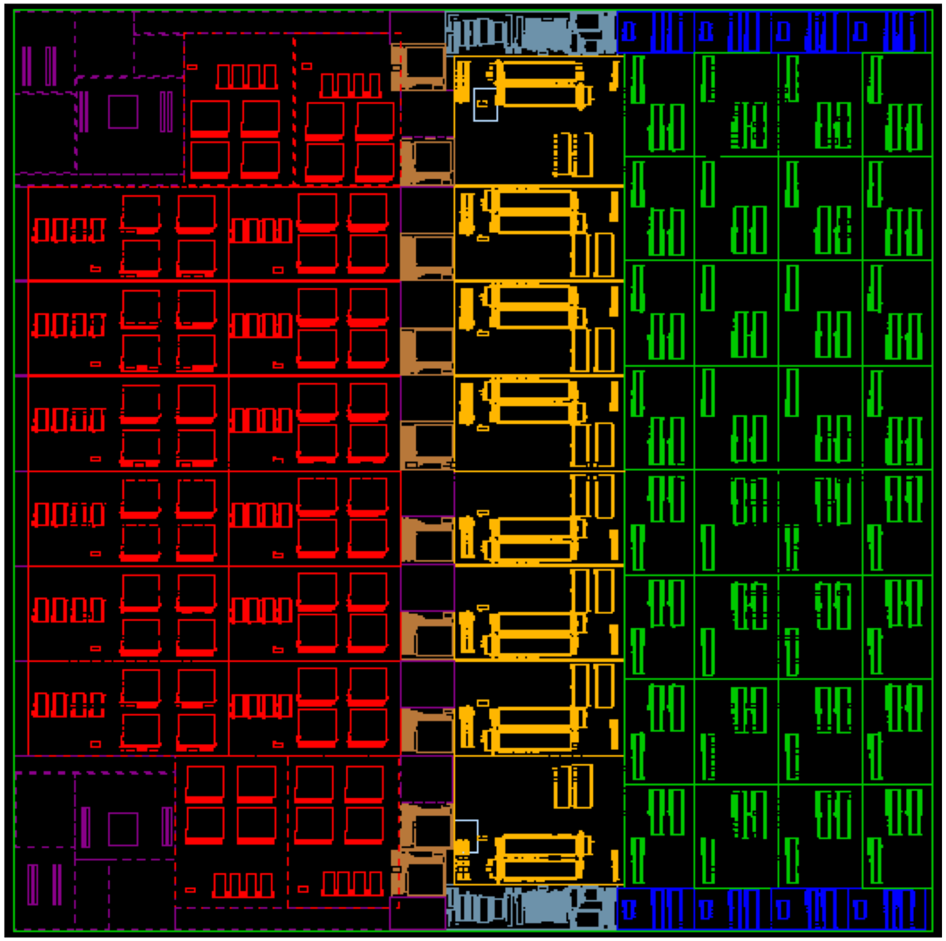
NUCA L2 Cache

- Prototype has 1MB L2 cache divided into sixteen 64KB banks
- 4x10 2D mesh topology
- Links are 128 bits wide
- Each processor can initiate 5 requests per cycle
- Requests and replies are wormhole-routed across the network
- 4 virtual channels prevent deadlocks
- Can sustain over 100 bytes per cycle to the processors

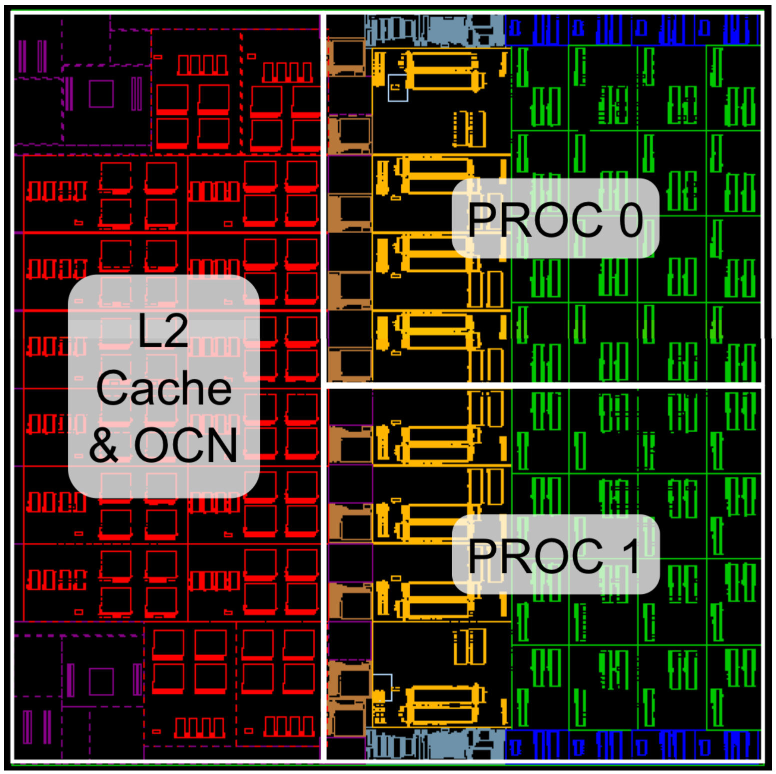


ASIC Implementation

- 130 nm 7LM IBM ASIC process
- 335 mm^2 die
- 47.5 mm x 47.5 mm package
- ~170 million transistors
- ~600 signal I/Os
- ~500 MHz clock freq
- Tape-out : fall 2005
- System bring-up : spring 2006



Functional Area Breakdown

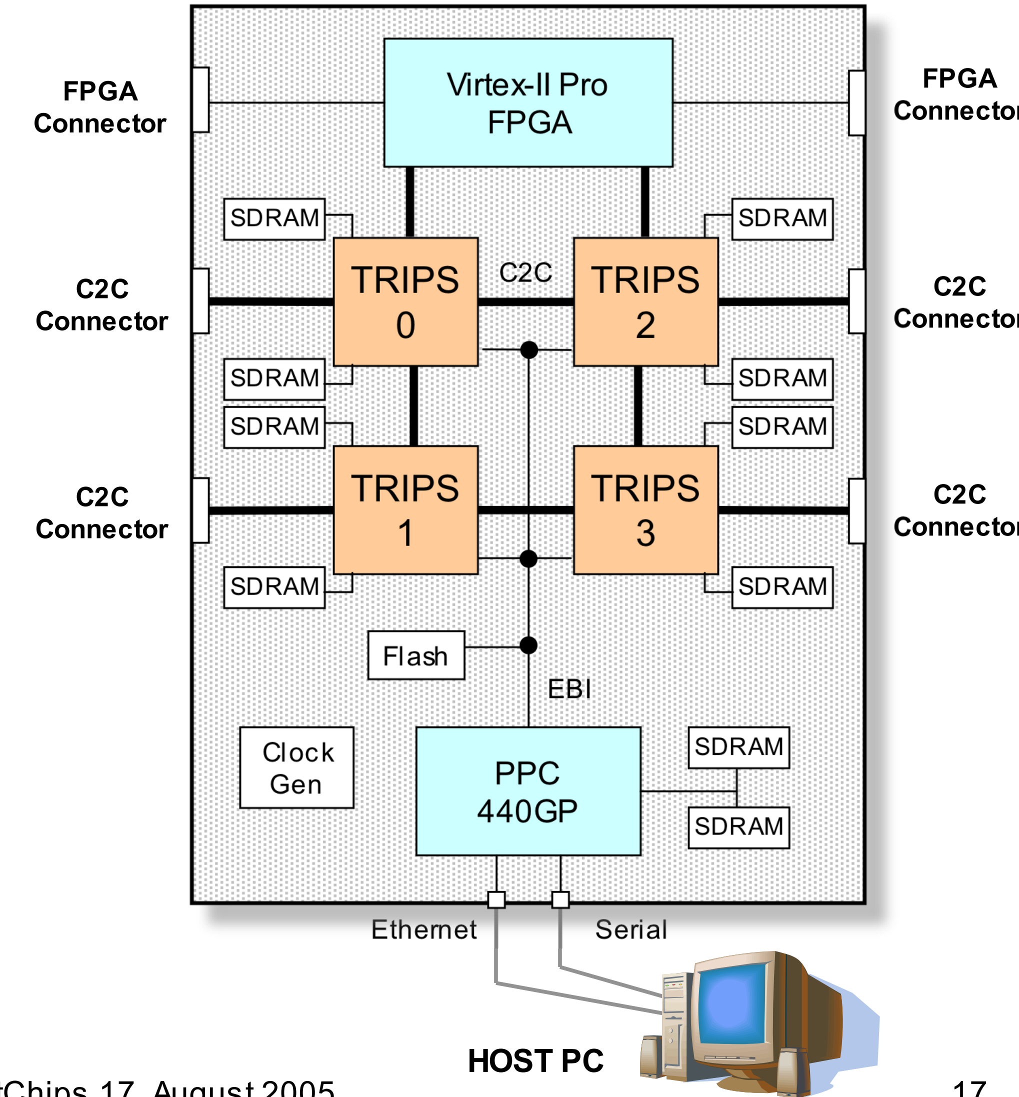


Overall Chip Area:
29% - Processor 0
29% - Processor 1
21% - Level 2 Cache
14% - On-Chip Network
7% - Other

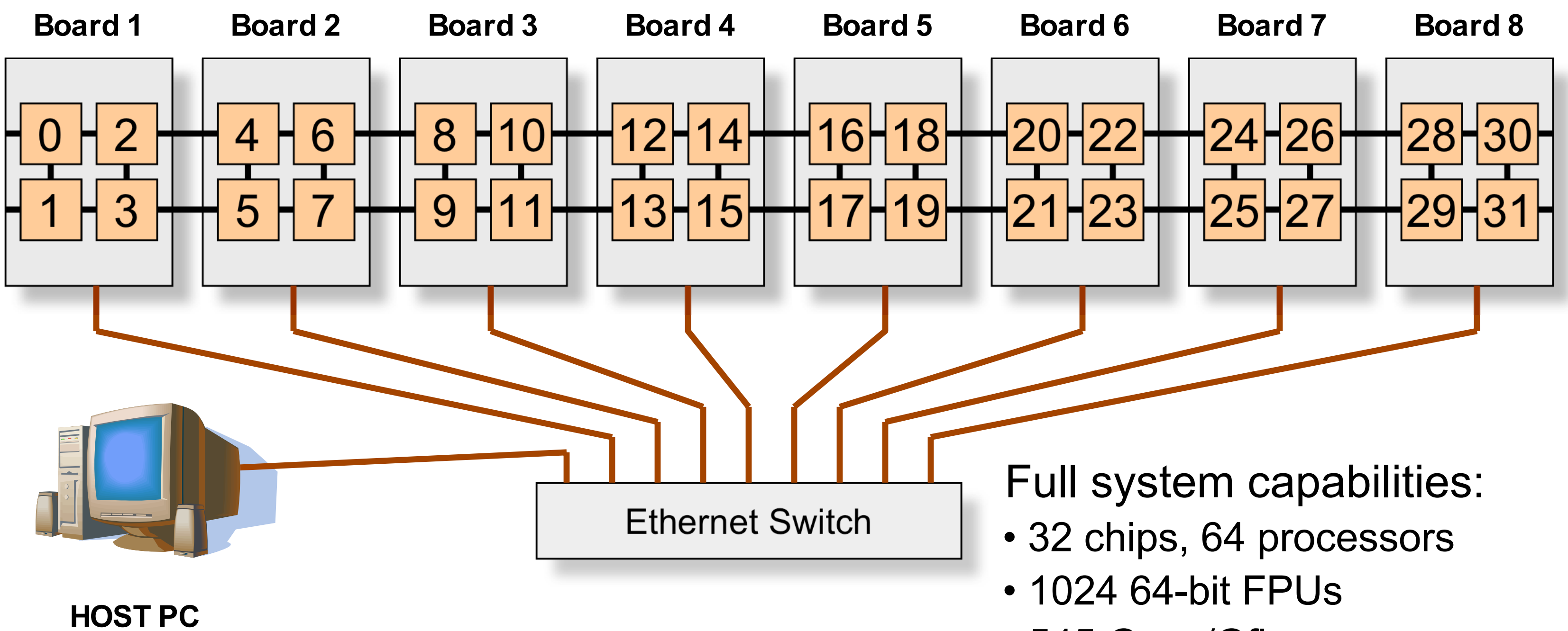
Processor Area:
30% - Functional Units (ALUs)
4% - Register Files & Queues
10% - Level 1 Caches
13% - Instruction Queues
13% - Load & Store Queues
12% - Operand Network
2% - Branch Predictor
16% - Other

TRIPS Board

- Board implements
  - 4 TRIPS chips
  - 8 GB of SDRAM (NUMA)
  - PPC 440GP
  - FPGA
- PowerPC 440GP used as control processor and host interface
- 2D chip-to-chip (C2C) network connects multiple TRIPS chips
- Intended for exploration of parallel processing scenarios, including streaming applications



TRIPS System



Full system capabilities:

- 32 chips, 64 processors
- 1024 64-bit FPUs
- 545 Gops/Gflops
- Up to 256 simultaneous threads
- 64 GB SDRAM
- 100 GB/s aggregate DRAM BW
- 3.4 GB/s C2C bisection BW

TRIPS Summary

- Distributed microarchitecture
  - Acknowledges and tolerates wire delay
  - Scalable protocols tailored for distributed components
- Tiled microarchitecture
  - Simplifies scalability
  - Improves design productivity
- The next step for instruction-level parallelism
  - EDGE ISA enables increased ILP
  - While also exploiting coarser types of parallelism

Q&A

Acknowledgements

- TRIPS Hardware Team
  - Raj Desikan, Saurabh Drolia, Madhu Sibi Govindan, Divya Gulati, Paul Gratz, Heather Hanson, Changkyu Kim, Haiming Liu, Ramdas Nagarajan, Nitya Ranganathan, Karu Sankaralingam, Simha Sethumadhavan, Premkishore Shivakumar
- TRIPS Software Team
  - Kathryn McKinley, Jim Burrill, Xia Chen, Sundeep Kushwaha, Bert Maher, Nick Nethercote, Suriya Narayanan, Sadia Sharif, Aaron Smith, Bill Yoder
- IBM Microelectronics Austin ASIC Group
- TRIPS Sponsors
  - DARPA Polymorphous Computing Architectures
  - Air Force Research Laboratories
  - National Science Foundation
  - IBM, Intel, Sun Microsystems

References

- **"Scaling to the End of Silicon with EDGE Architectures,"** D. Burger, S.W. Keckler, K.S. McKinley, et al. *IEEE Computer*, 37 (7), pp. 44-55, July, 2004.
- **"A Wire-Delay Scalable Microprocessor Architecture for High Performance Systems,"** S.W. Keckler, D. Burger, C.R. Moore, R. Nagarajan, K. Sankaralingam, V. Agarwal, M.S. Hrishikesh, N. Ranganathan, and P. Shivakumar. *International Solid-State Circuits Conference (ISSCC)*, pp. 1068-1069, February, 2003.
- **"A Design Space Evaluation of Grid Processor Architectures,"** R. Nagarajan, K. Sankaralingam, D. Burger, and S.W. Keckler. *34th Annual International Symposium on Microarchitecture (MICRO)*, pp. 40-51, December, 2001.
- **"An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches,"** C. Kim, D. Burger, and S.W. Keckler. *10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 211-222, October, 2002.

TRIPS Instruction Formats

General Instruction Formats

31 – 25	24 23	22 – 18	17 – 9	8 – 0	
OPCODE	PR	XOP	T1	T0	**G**
OPCODE	PR	XOP	IMM	T0	**I**

Load and Store Instruction Formats

31 – 25	24 23	22 – 18	17 – 9	8 – 0	
OPCODE	PR	LSID	IMM	T0	**L**
OPCODE	PR	LSID	IMM	0	**S**

Branch Instruction Format

31 – 25	24 23	22 – 20	19 – 0	
OPCODE	PR	EXIT	OFFSET	**B**

Constant Instruction Format

31 – 25	24 – 9	8 – 0	
OPCODE	CONST	T0	**C**

Read Instruction Format

21	20 – 16	15 – 8	7 – 0	
V	GR	RT0	RT1	**R**

Write Instruction Format

5	4 – 0	
V	GR	**W**

Not shown: M3, M4 formats

INSTRUCTION FIELDS

OPCODE = Primary Opcode

XOP = Extended Opcode

PR = Predicate Field

IMM = Signed Immediate

T0 = Target 0 Specifier

T1 = Target 1 Specifier

LSID = Load/Store ID

EXIT = Exit ID

OFFSET = Branch Offset

CONST = 16-bit Constant

V = Valid Bit

GR = General Register Index

RT0 = Read Target 0 Specifier

RT1 = Read Target 1 Specifier

TRIPS Instruction Set

Categories	Instructions
Reads	READ
Writes	WRITE
Loads	LB, LBS, LH, LHS, LW, LWS, LD
Stores	SB, SH, SW, SD
Integer Arithmetics	ADD, ADDI, SUB, SUBI, MUL, MULI, DIVS, DIVSI, DIVU, DIVUI
Integer Logicals	AND, ANDI, OR, ORI, XOR, XORI
Integer Shifts	SLL, SLLI, SRL, SRLI, SRA, SRAI
Integer Extends	EXTSB, EXTSH, EXTSW, EXTUB, EXTUH, EXTUW
Integer Relationals	TEQ, TEQI, TNE, TNEI, TLE, TLEI, TLT, TLTI, TLEU, TLEUI, TLTU, TLTUI, TGE, TGEI, TGT, TGTI, TGEU, TGEUI, TGTU, TGTUI
Floating-Point Arithmetics	FADD, FSUB, FMUL, FDIV
Floating-Point Conversions	FITOD, FDTOI, FDTOS, FSTOD
Floating-Point Relationals	FEQ, FNE, FLE, FLT, FGE, FGT
Branches	BR, BRO, CALL, CALLO, RET, SCALL
Other	MOV, MOVI, MOV3, MOV4, GENS, GENU, APP, MFPC, NULL, LOCK