nCUBE3 Integrated MPP Node Processor

Bob Duzett, Barry Keane

Naoki Kamiya, Stan Kenoyer, Steve Page, Russell Randolph
John Maneely

I. System/Node Architecture

II. Memory Hierarchy

III. Optimizing Data flow

IV. Optimizing Ctl flow

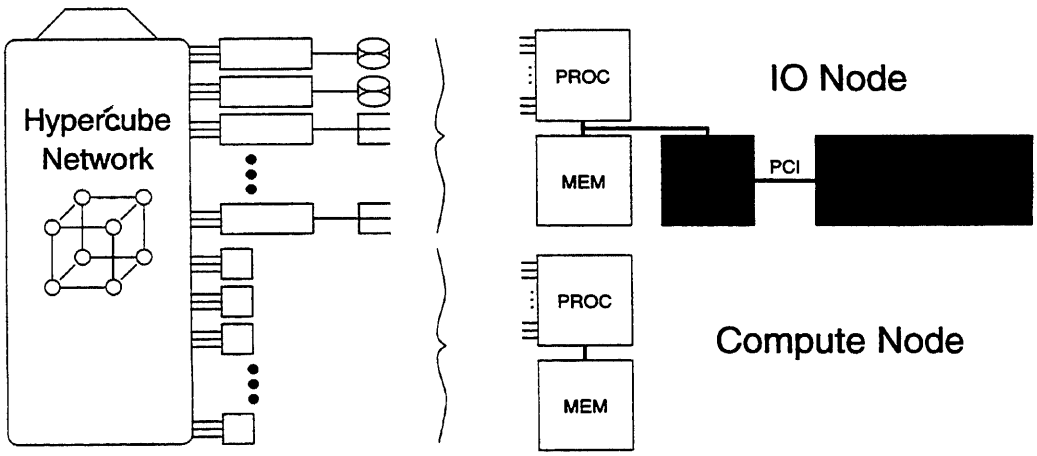
V. Interprocessor Communications

VI. Status/Performance

RD
7/94

nCUBE

nCUBE3 System Architecture



MIMD Hypercube

Adaptive Routing

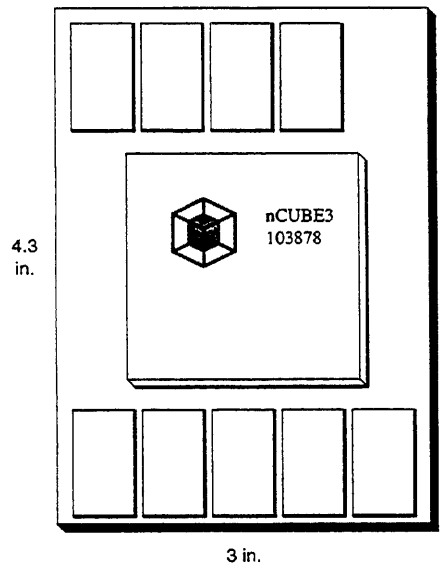
Integrated Node

64-bit CPU

RD
7/94

nCUBE

nCUBE3 Processing Node



Processor

- 345-pin CPGA; 1.85" x 1.85"
- 2.5M transistors
- 0.5 micron CMOS, 3-layer metal
- 3.3 volts
- 6-8 watts

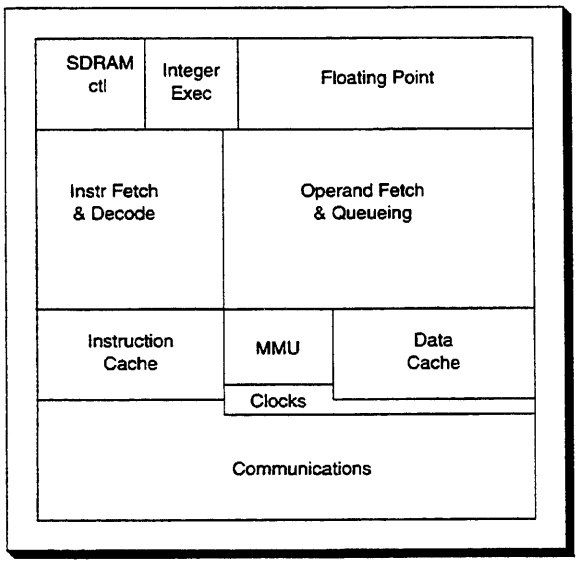
SDRAMs

- 1 or 2 Banks; 9/18/36 SDRAMs
- 16Mbytes - 1Gbyte
- LVTTL - to 100MHz

RD
7/94

nCUBE

nCUBE3 Single-chip Node Computer

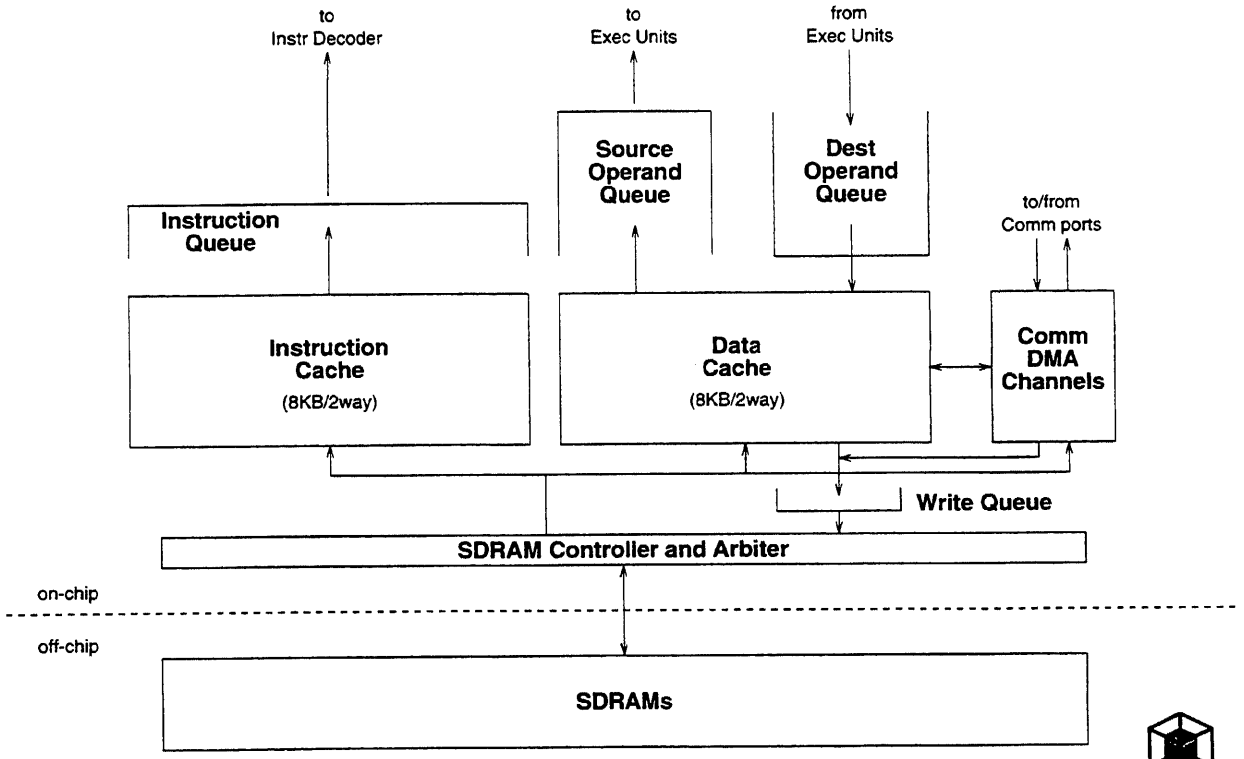


	die area
CPU	**56%**
Instr Fetch & Decode	12
Opr Fetch & Queueing	19
Instr & Data Caches	10
64-bit Integer Execution	3
64-bit FPU	9
64-bit MMU	3
Communications	**20%**
Adaptive Msg Router	2
16 DMA Channels	8
18 Comm Ports	10
SDRAM controller	**5%**
Clock generator	**2%**
Pad Ring	**17%**

RD
7/94

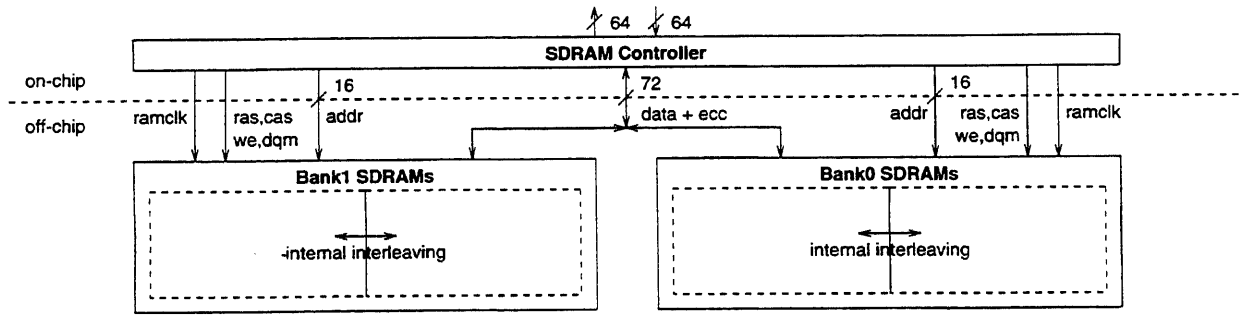
nCUBE

Memory Hierarchy



RD
7/94

nCUBE



SDRAMs

- high bandwidth: 1 data word per cycle (to 100MHz)
- internal interleaving
- burst access w/ line-wrap; page-mode

Integrated SDRAM Controller

- page mode, internal interleaving, line-wrap
- 1 or 2 memory banks; duplicate addr & ctl
- 50MHz to 100MHz: 400 MB/sec to 800 MB/sec

NO Secondary Cache - Cost, Power vs Performance

RD
7/94

nCUBE

Caches

Instruction and Data Caches

- 8K bytes each
- 2-way set associative
- 32-byte line size (4 64-bit words)
- 1-Cycle Access

Instruction Cache

- 3-Line Instruction Queue feeds Decoder
- Prefetch

Data Cache

- Non-blocking, pipelined design; Miss buffer (4-deep)
- Write-back, write-thru, or non-cacheable; Write buffer
- DMA snoop port

RD
7/94

nCUBE

Operand Queues

Source Queue (depth=16)

- decouples instr decoder from mem and exe pipelines
- data fetched from dest_queue, dcache, wr-q, or mem
- exe results forwarded to matching src-q entries
- mem-read data forwarded to matching src-q entries

Destination Queue (depth=16)

- buffers execution results
- writes to data cache or wr-q or both
  - write-back, write-thru, or non-cacheable

nCUBE

RD
7/94

Optimizing Data Flow

Vector_Move (VMOVD/VMOVW)

- moves 32 bytes to/from mem/regs/imm
  - eg, move cache line to/from registers in one cycle
- eliminates cache-line write allocate
- cache override: force either or both operands non-cach.

Prefetch (PREF)

- prefetches memory line into data cache
- bypasses exe pipeline; out-of-order
- overlaps/hides memory latency

Flush-line (FLLN)

- flushes memory line from data cache

Flush-dcache (FLDC)

- flushes indicated cache line

RD
7/94

nCUBE

Vector Load/Store

memcpy():

```
** (r1) = source (in DRAM) **
** (r3) = destination (in DRAM) **
** (sp) = cache temporary **

avec:  vmovd   (r1)+@dram,   0(sp)
       vmovd   (r1)+@dram,   32(sp)
        •
        •
       vmovd   0(sp),        (r3)+@dram
       vmovd   32(sp),       (r3)+@dram
        •
        •
       bg      avec
```

vector load/store cache management:

minimizes DRAM ping-pong (page break penalties)

avoids cache collisions and cache pollution

avoids read-modify-write of result array (write allocate)

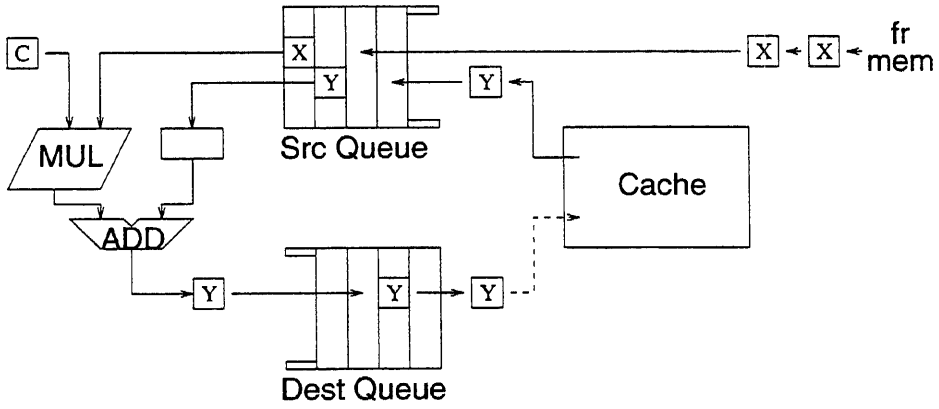
RD
7/94

nCUBE

DAXPY Throughput

[Y] = [Y] + C * [X]

Scalar - 0.67 FLOPs per cycle



C X Y = 64-bit Constant or Vector Element

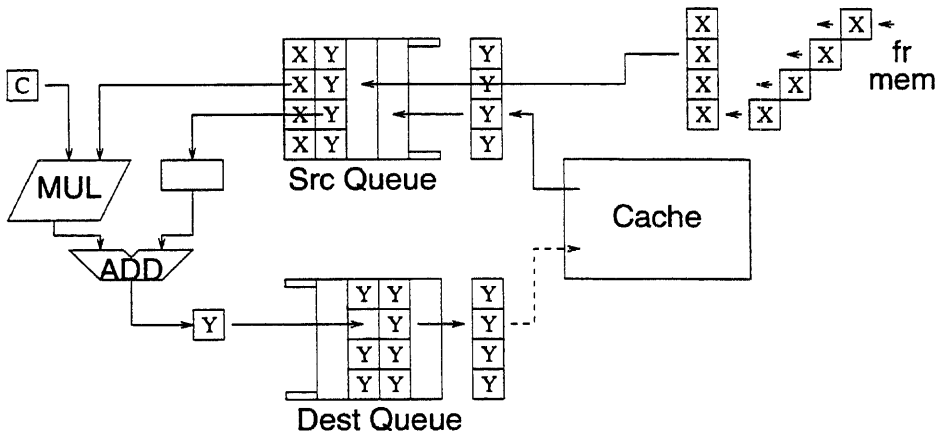
RD
7/94

nCUBE

DAXPY Throughput

[Y] = [Y] + C * [X]

Vector - 2.0 FLOPs per cycle



C X Y = 64-bit Constant or Vector Element

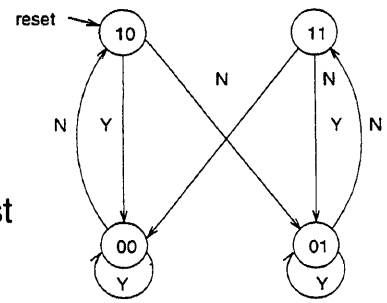
RD
7/94

nCUBE

Optimizing Control Flow

Dynamic Branch Prediction

- 256-entry table, direct map
- 2-bit state



Conditional Move - CMOV <cond>,src,dst

- if <cond> dst = src
- supports speculative execution

Convert_cond_code - CCC <cond>,dst

- dst = <cond> ? 1 : 0
- set boolean result based on flag condition

Boolean_AND - BAND src1,src2,dst

- dst = src1 && src2
- boolean expression evaluation without branches

RD
7/94

nCUBE

Speculative Execution

transform:	into:
if (condition)	block_1
block_1	block_2
else	CMOV
block_2	CMOV

depending on:

- average branch cost
- execution costs of block_1 and block_2
- branch profiling data (if available)

criteria: spec_exec cost < branch cost

- combine two cheap blocks into one
- combine an infrequently-executed, cheap block with an oft-executed, expensive block

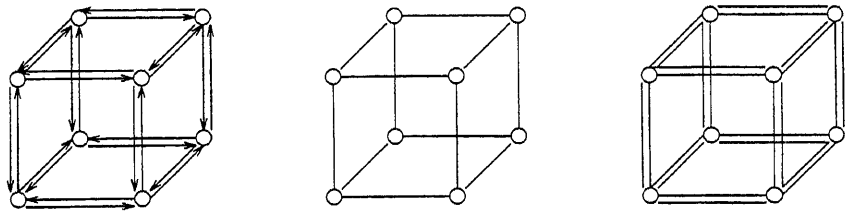
RD
7/94

nCUBE

nCUBE3 Communications

Hypercube Network: 2**n

- 18 full-duplex ports on each processor
- n hypercube connections + m folded connections
- messages cut-through without CPU intervention



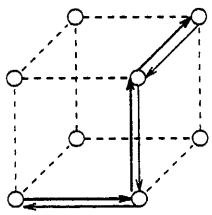
16 DMA channels: 8 Send, 8 Receive

- double-buffered DMA control
- 32-byte data buffer per channel
- scatter-gather support

RD
7/94

nCUBE

End-to-End Reporting



Status packet delivered from target node, back to src

- status = h/w error status or s/w status
- retraces back along xmission path, using same links
- xmission path held until ETE packet unravels it

Benefits of hardware ETE

- Quick and Reliable
- no additional xmissions (no CPU and s/w overhead)

RD
7/94

nCUBE

Routing Methods

Oblivious Wormhole routing

- only 1 fixed path - dimension-order, 'e-route'
- address pkt advances the path until blocked
- data flows directly behind address pkt, wormhole fashion

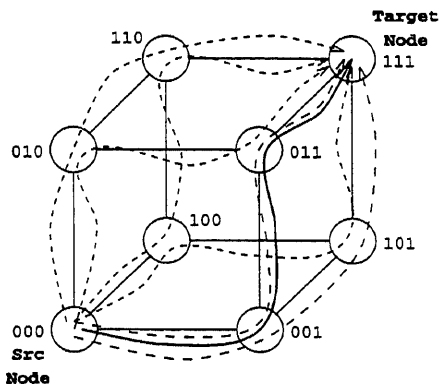
'Maze' adaptive routing

- exhaustive search of min-length paths until free path found
- addr pkt traverses any min-path unallocated link, back-tracking and deallocating when blocked
- uses ETE reporting network to back-track/report
- data xmission begins after path established

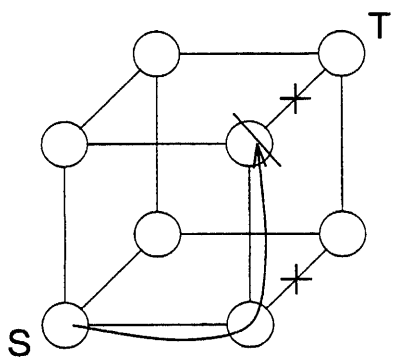
RD
7/94

nCUBE

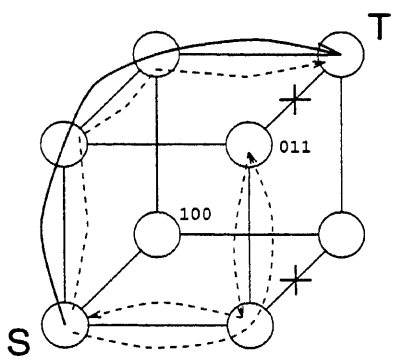
Routing



Oblivious
Wormhole



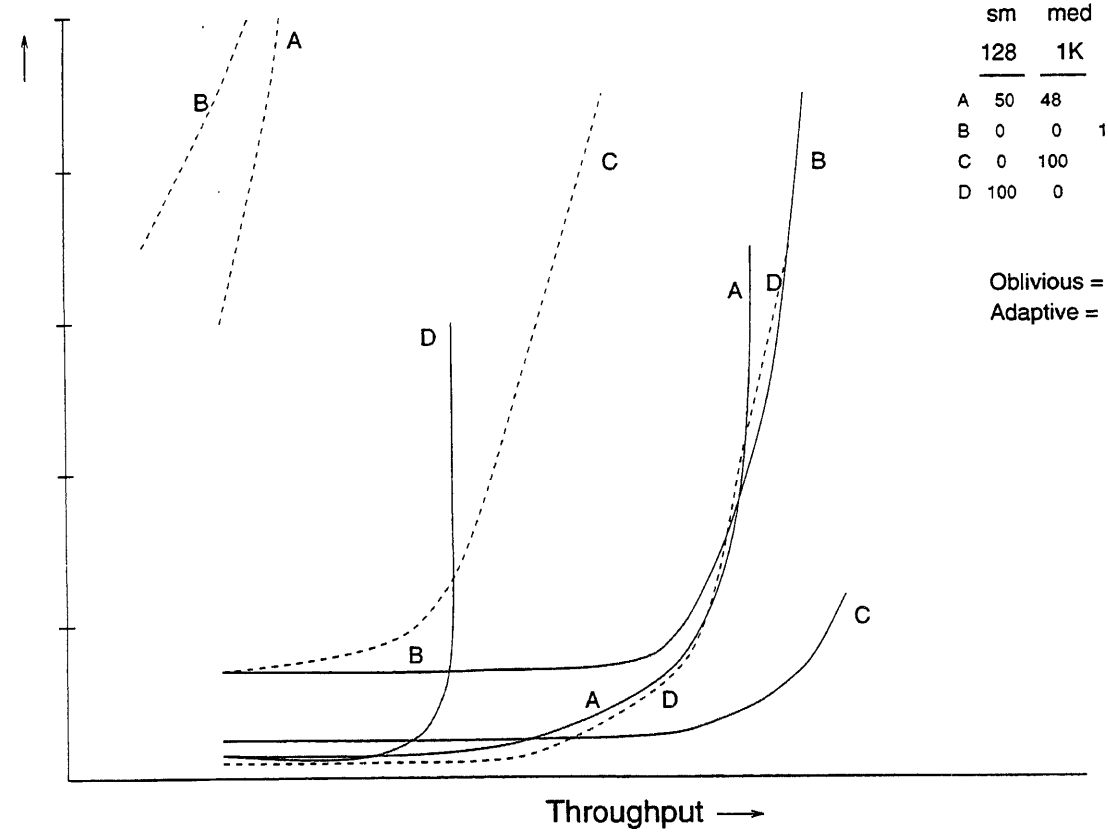
Maze



RD
7/94

nCUBE

Message Latency



Message Mixes

	sm	med	lg	
	128	1K	8K	bytes
A	50	48	2	%
B	0	0	100	
C	0	100	0	
D	100	0	0	

Oblivious = - - - - -
Adaptive = ——

RD
7/94

nCUBE

nCUBE3 Processor Status and Performance

(to be provided at time of presentation)

RD
7/94

nCUBE