

Lecture: Review Session

- Datacenters, energy proportionality, GPUs – watch posted recordings
- Final exam details:
 - Tuesday 12/13, 1pm – 3pm
 - 80%+ on post-midterm material
 - A couple “unseen” problems, a few “short-response” questions
 - 3+3 reference sheets (double sided)
 - Show steps; calculators allowed

Hardware Trends

Why the recent emphasis on accelerators?

- Stagnant single- and multi-thread performance with general-purpose cores
 - Dark silicon (emphasis on power-efficient throughput)
 - End of scaling
 - No low-hanging fruit
- Emergence of deep neural networks

Commercial Hardware

Machine Learning accelerators

Google

Google TPU (inference and training)



Recent NVIDIA chips (Volta, NVDLA)



Microsoft Brainwave, Catapult



Intel Loihi and Nervana



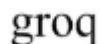
Cambricon

GRAPHCORE

Graphcore (training)



Cerebras (training)

groq

Groq (inference)



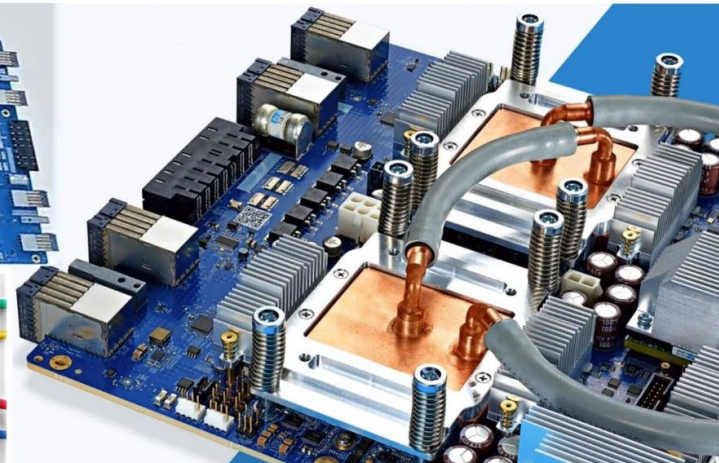
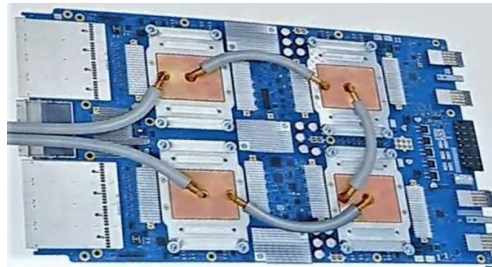
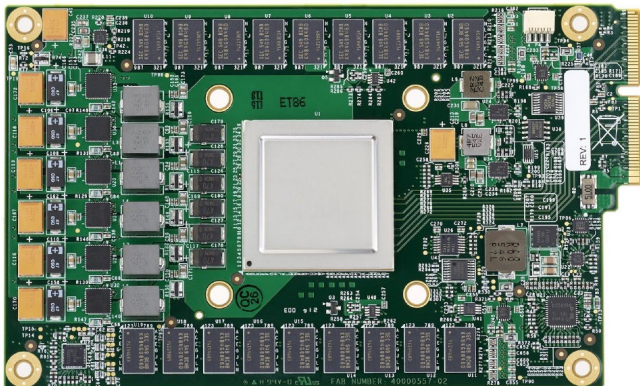
Tesla FSD (inference)

Machine Learning Workloads

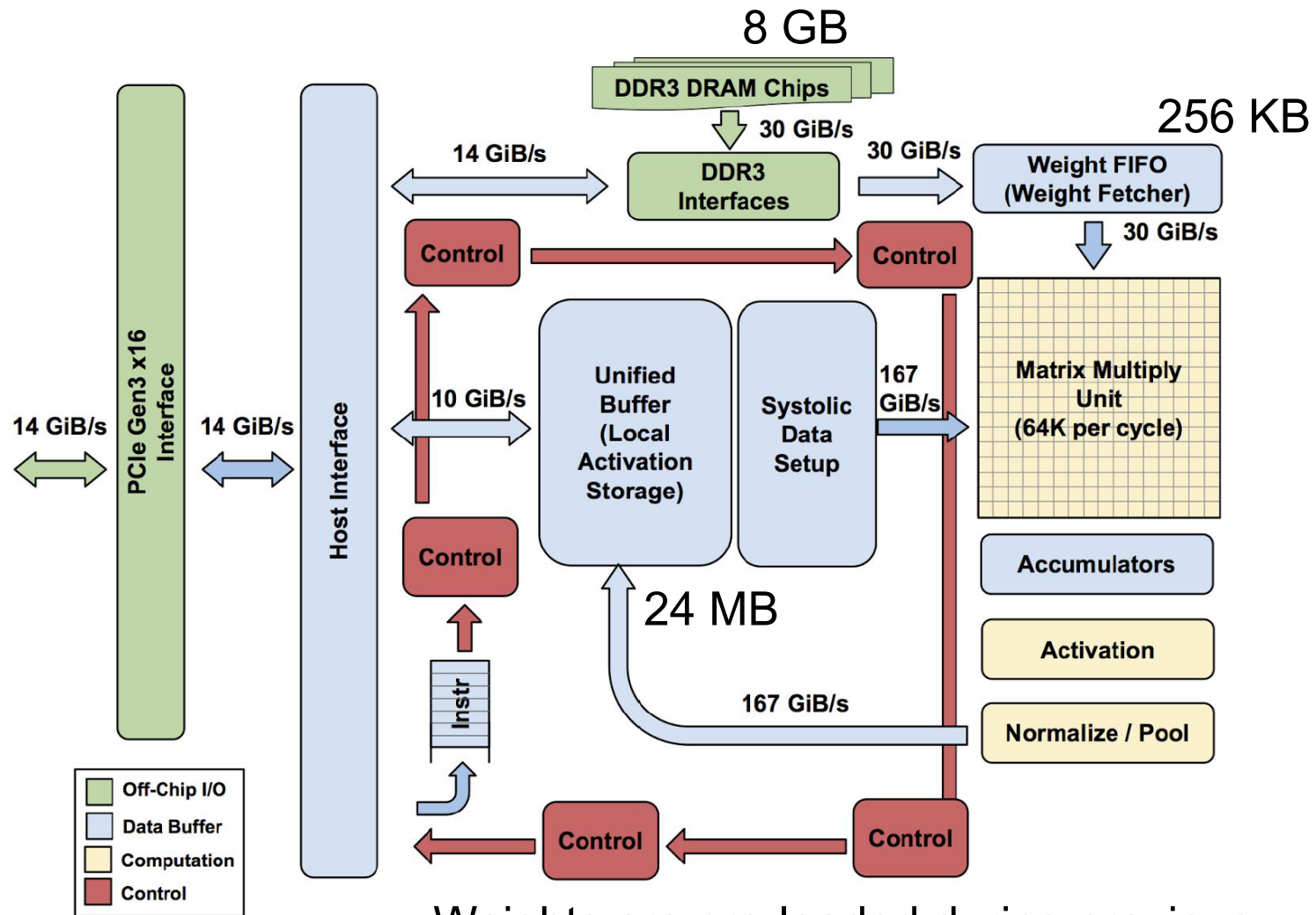
- Dominated by dot-product computations
- Deep neural networks: convolutional and fully-connected layers
- Convolutions exhibit high data reuse
- Fully-connected layers have high memory-to-compute ratio

Google TPU

- Version 1: 15-month effort, basic design, only for inference, 92 TOPs peak, 15x faster than GPU, 40 W 28nm 300 mm² chip
- Version 2: designed for training, a pod is a collection of v2 chips connected with a torus topology
- Version 3: 8x higher throughput, liquid cooled



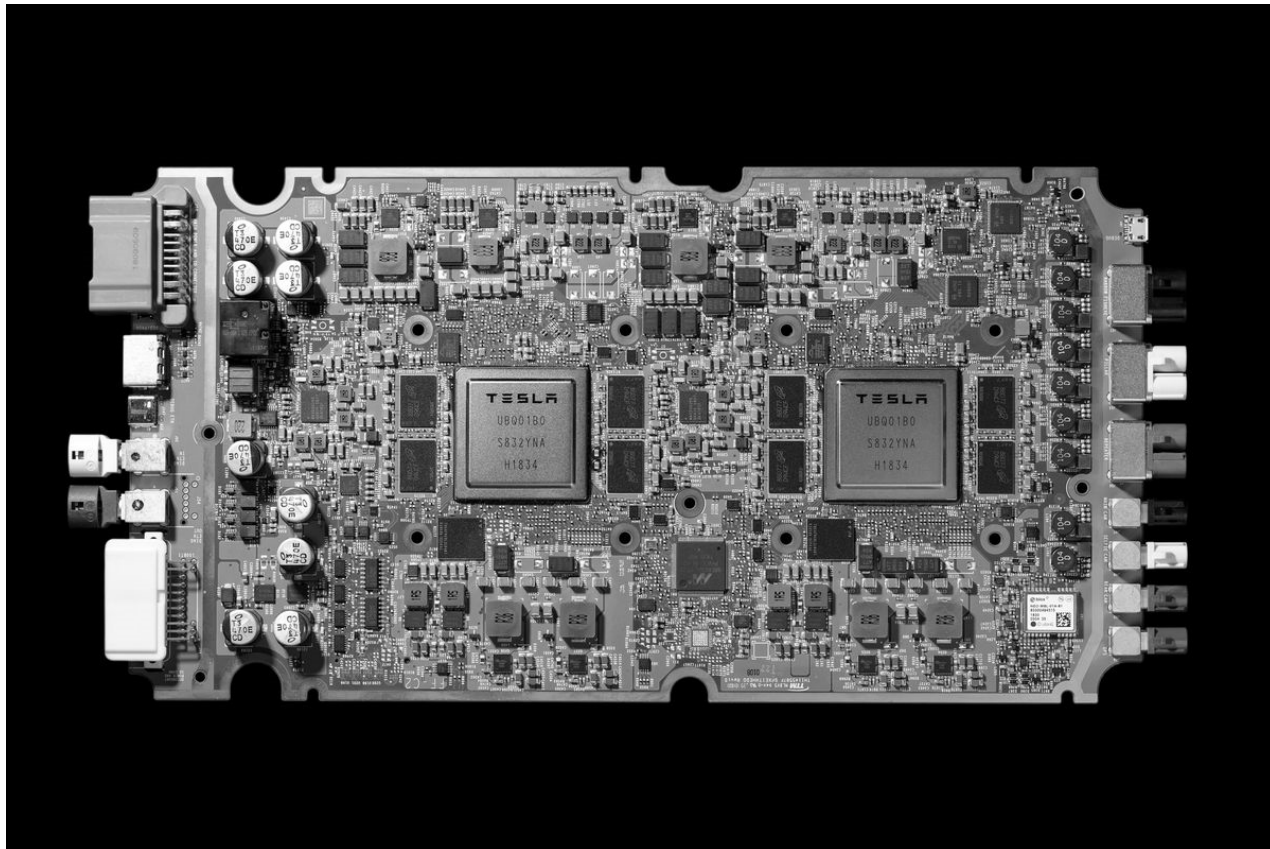
TPU Architecture



Weights are pre-loaded during previous phase and inputs flow left to right.

Tesla FSD

- Tesla's custom accelerator chip, shipping in cars since April 2019
- FSD sits behind the glovebox, consumes 72W
- 18 months for first design, next generation out in 2 years



NN Accelerator Chip (NNA)

- Goals: under 100 W (2% impact on driving range, cooling, etc.), 50 TOPs, batch size of 1 for low latency, GPU support as well, security/safety.
- Security: all code must be attested by Tesla
- Safety: two completely independent systems on the board that verify every output
- The FSD 2.5 design (GPU based) consumes 57 W, the 3.0 design consumes 72 W, but is 21x faster (72 TOPs)
- 20% saving in cost by designing their own chip

OoO Timeline

*1: Fetch width full. Fetched in next cycle.

*2: Issue width full. Issued in next cycle.

*3: Commit width full. Committed in next cycle

*4: Commit delayed in order to commit in order

*5: No free register in Free Register List. Must wait until a physical register frees up.

- InQ - Cycle at which the instruction arrived into the Issue Queue
- Issued - Cycle at which the instruction is issued (leaves Issue Queue)
- Complete - Cycle at which the instruction completes
- Commit - Cycle at which the instruction gets committed

Inst #	Original code	Renamed Code	Entry added to Spec. Reg. Map	InQ	Issued	Complete	Commit	On commit, entry added to	
								Committed Reg. Map	Reg. Free List
1	LD LR1, 0(LR2)	LD PR33, 0(PR2)	LR1->PR33	i	i+1	i+7	i+7	LR1->PR33	PR1
2	DADD LR1, LR1, LR3	DADD PR34, PR33, PR3	LR1->PR34	i	i+3	i+8	i+8	LR1->PR34	PR33
3	ST.D LR1, 0(LR5)	ST.D PR34, 0(PR5)	-	i	i+4	i+10	i+10	-	-
4	DADD LR2, LR2, 8	DADD PR35, PR2, 8	LR2->PR35	i+1 *1	i+2	i+7	i+10 *4	LR2->PR35	PR2
5	DADD LR5, LR5, 8	DADD PR36, PR5, 8	LR5->PR36	i+1	i+2	i+7	i+10 *4	LR5->PR36	PR5
6	BNE LR2, LR4, line1	BNE PR35, PR4, line1	-	i+1	i+3	i+8	i+11 *3 *4-	-	-
7	LD LR1, 0(LR2)	LD PR37, 0(PR35)	LR1->PR37	i+2 *1	i+3	i+9	i+11 *4	LR1->PR37	PR34
8	DADD LR1, LR1, LR3	DADD PR38, PR37, PR3	LR1->PR38	i+2	i+5	i+10	i+11 *4	LR1->PR38	PR37
9	ST.D LR1, 0(LR5)	ST.D PR38, 0(PR36)	-	i+2	i+6	i+12	i+12	-	-
10	DADD LR2, LR2, LR8	DADD PR1, PR35, 8	LR2->PR1	i+8 *5	i+9	i+14	i+14	LR2->PR1	PR35
11	DADD LR5, LR5, LR8	DADD PR33, PR36, 8	LR5->PR33	i+9 *5	i+10	i+15	i+15	LR5->PR33	PR36
12	BNE LR2, LR4, line1	BNE PR1, PR4, line1	-	i+9	i+10	i+15	i+15	-	-

Problem 4

- Consider the following LSQ and when operands are available. Estimate when the address calculation and memory accesses happen for each ld/st. Assume memory dependence prediction.

	Ad. Op	St. Op	Ad.Val	Ad.Cal	Mem.Acc
LD R1 ← [R2]	3		abcd		
LD R3 ← [R4]	6		adde		
ST R5 → [R6]	4	7	abba		
LD R7 ← [R8]	2		abce		
ST R9 → [R10]	8	3	abba		
LD R11 ← [R12]	1		abba		

Problem 1

- Memory access time: Assume a program that has cache access times of 1-cyc (L1), 10-cyc (L2), 30-cyc (L3), and 300-cyc (memory), and MPKIs of 20 (L1), 10 (L2), and 5 (L3). Should you get rid of the L3?

With L3: $1000 + 10 \times 20 + 30 \times 10 + 300 \times 5 = 3000$

Without L3: $1000 + 10 \times 20 + 10 \times 300 = 4200$

Problem 3

- Assume a 2-way set-associative cache with just 2 sets. Assume that block A maps to set 0, B to 1, C to 0, D to 1, E to 0, and so on. For the following access pattern, estimate the hits and misses:

A B B E C C A D B F A E G C G A

M M H M M H M M H M H M M M H M

Problem 5

- 8 KB fully-associative data cache array with 64 byte line sizes, assume a 40-bit address
- How many sets (1) ? How many ways (128) ?
- How many index bits (0), offset bits (6), tag bits (34) ?
- How large is the tag array (544 bytes) ?

Equations:

Data array size (cache size) = #sets x #ways x blocksize

Tag array size = #sets x #ways x tagsize

Index bits = \log_2 (#sets)

Offset bits = \log_2 (blocksize)

Tag bits + index bits + offset bits = address width

Problem 3

- Assume that page size is 16KB and cache block size is 32 B. If I want to implement a virtually indexed physically tagged L1 cache, what is the largest direct-mapped L1 that I can implement? What is the largest 2-way cache that I can implement?

Similar to HW 7, Q1

- Assume a large shared LLC that is tiled and distributed on the chip. Assume that the OS page size is 16KB. The entire LLC has a size of 32 MB, uses 128-byte blocks, and is 32-way set-associative. What is the maximum number of tiles such that the OS has full flexibility in placing a page in a tile of its choosing?

Problem 1

- What is the maximum memory capacity supported by the following server: 2 processor sockets, each socket has 4 memory channels, each channel supports 2 dual-ranked DIMMs, and x4 4Gb DRAM chips?

2 sockets x 4 channels x 2 DIMMs x 2 ranks x
16 chips x 4Gb capacity = 256 GB

What is the memory bandwidth available to the server if each memory channel runs at 800 MHz?

2 sockets x 4 channels x 800M (cycles per second) x
2 (DDR, hence 2 transfers per cycle) x 64 (bits per transfer)
= 102.4 GB/s

Problem 4

For the following access stream, estimate the finish times for each access with the following scheduling policies:

Req	Time of arrival	Open	Closed	Oracular
X	10 ns	50	50	50
X+1	15 ns	70	70	70
Y	100 ns	160	140	140
Y+1	180 ns	200	220	200
X+2	190 ns	260	300	260
Y+2	205 ns	320	240	320

Note that X, X+1, X+2, X+3 map to the same row and Y, Y+1 map to a different row in the same bank. Ignore bus and queuing latencies. The bank is precharged at the start.

** A more sophisticated oracle can do even better.

Problem 5

- Consider a single 4 GB memory rank that has 8 banks. Each row in a bank has a capacity of 8KB. On average, it takes 40ns to refresh one row. Assume that all 8 banks can be refreshed in parallel. For what fraction of time will this rank be unavailable? How many rows are refreshed with every refresh command?

The memory has $4\text{GB}/8\text{KB} = 512\text{K}$ rows

There are 8K refresh operations in one 64ms interval.

Each refresh operation must handle $512\text{K}/8\text{K} = 64$ rows

Each bank must handle 8 rows

One refresh operation is issued every 7.8us and the memory is unavailable for 320ns, i.e., for 4% of time.

Meltdown

Attacker code

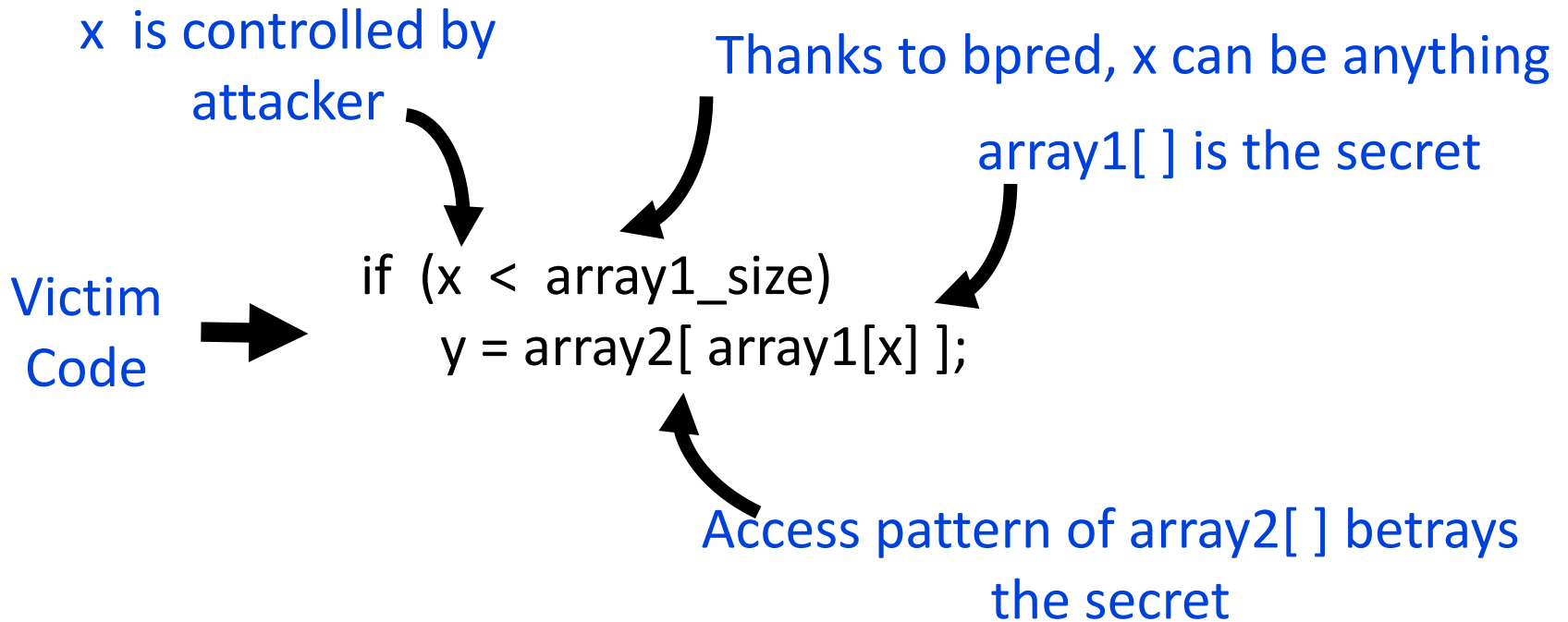
Fill the cache with your own data X

```
lw R1 ← [illegal address]
```

```
lw ... ← [R1]
```

Scan through X and record time per access

Spectre: Variant 1



Spectre: Variant 2

Attacker code

Label0: if (1)

Label1: ...

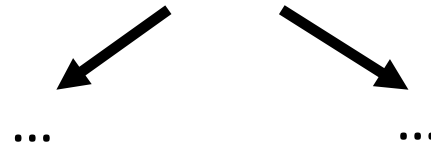


Victim code

R1 ← (from attacker)

R2 ← some secret

Label0: if (...)



Victim code

Label1:

lw [R2]

Snooping Example

Request	Cache Hit/Miss	Request on the bus	Who responds	State in Cache 1	State in Cache 2	State in Cache 3	State in Cache 4
				Inv	Inv	Inv	Inv
P1: Rd X	Miss	Rd X	Memory	S	Inv	Inv	Inv
P2: Rd X	Miss	Rd X	Memory	S	S	Inv	Inv
P2: Wr X	Perms Miss	Upgrade X	No response. Other caches invalidate.	Inv	M	Inv	Inv
P3: Wr X	Write Miss	Wr X	P2 responds	Inv	Inv	M	Inv
P3: Rd X	Read Hit	-	-	Inv	Inv	M	Inv
P4: Rd X	Read Miss	Rd X	P3 responds. Mem wrtbn	Inv	Inv	S	S

Directory Example

Request	Cache Hit/Miss	Messages	Dir State	State in C1	State in C2	State in C3	State in C4
				Inv	Inv	Inv	Inv
P1: Rd X	Miss	Rd-req to Dir. Dir responds.	X: S: 1	S	Inv	Inv	Inv
P2: Rd X	Miss	Rd-req to Dir. Dir responds.	X: S: 1, 2	S	S	Inv	Inv
P2: Wr X	Perms Miss	Upgr-req to Dir. Dir sends INV to P1. P1 sends ACK to Dir. Dir grants perms to P2.	X: M: 2	Inv	M	Inv	Inv
P3: Wr X	Write Miss	Wr-req to Dir. Dir fwds request to P2. P2 sends data to Dir. Dir sends data to P3.	X: M: 3	Inv	Inv	M	Inv
P3: Rd X	Read Hit	-	-	Inv	Inv	M	Inv
P4: Rd X	Read Miss	Rd-req to Dir. Dir fwds request to P3. P3 sends data to Dir. Memory wrtbk. Dir sends data to P4.	X: S: 3, 4	Inv	Inv	S	S

Test-and-Test-and-Set

- lock: test register, location
bnz register, lock
t&s register, location
bnz register, lock
CS
st location, #0

Spin Lock with Low Coherence Traffic

lockit: LL R2, 0(R1) ; load linked, generates no coherence traffic
BNEZ R2, lockit ; not available, keep spinning
DADDUI R2, R0, #1 ; put value 1 in R2
SC R2, 0(R1) ; store-conditional succeeds if no one
; updated the lock since the last LL
BEQZ R2, lockit ; confirm that SC succeeded, else keep trying

- If there are i processes waiting for the lock, how many bus transactions happen?
1 write by the releaser + i (or 1) read-miss requests +
 i (or 1) responses + 1 write by acquirer + 0 ($i-1$ failed SCs) +
 $i-1$ (or 1) read-miss requests + $i-1$ (or 1) responses

(The $i/i-1$ read misses can be reduced to 1)

Example Programs

Initially, $A = B = 0$

P1

$A = 1$

if ($B == 0$)

critical section

P2

$B = 1$

if ($A == 0$)

critical section

Initially, $Head = Data = 0$

P1

$Data = 2000$

$Head = 1$

P2

while ($Head == 0$)

{ }

... = $Data$

Initially, $A = B = 0$

P1

$A = 1$

P2

if ($A == 1$)

$B = 1$

P3

if ($B == 1$)

register = A

Problem 1

- What are possible outputs for the program below?

Assume $x=y=0$ at the start of the program

	Thread 1		Thread 2
A	$x = 10$	a	$y=20$
B	$y = x+y$	b	$x = y+x$
C	Print y		

Possible scenarios: 5 choose 2 = 10

ABCab	ABaCb	ABabC	AaBCb	AaBbC
10	20	20	30	30
AabBC	aABCb	aABbC	aAbBC	abABC
50	30	30	50	30

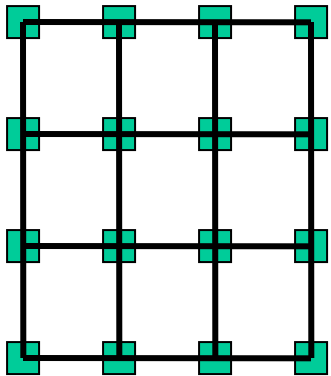
Fences

P1	P2
{ Region of code with no races }	{ Region of code with no races }
Fence Acquire_lock Fence	Fence Acquire_lock Fence
{ Racy code }	{ Racy code }
Fence Release_lock Fence	Fence Release_lock Fence

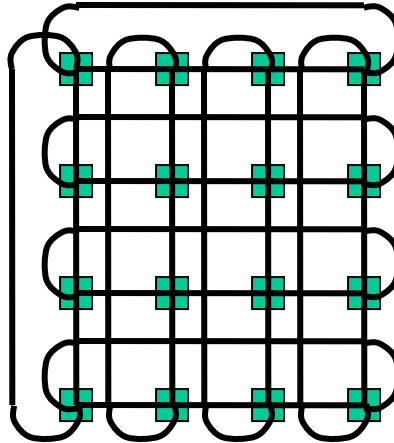
Deadlock

- Deadlock happens when there is a cycle of resource dependencies – a process holds on to a resource (A) and attempts to acquire another resource (B) – A is not relinquished until B is acquired

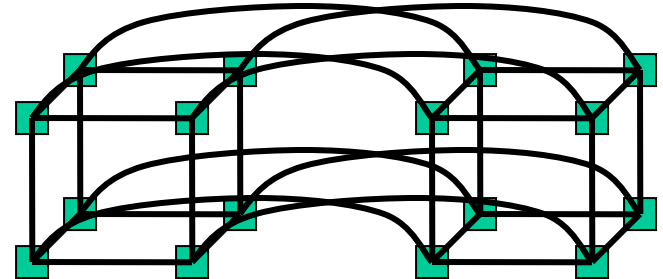
Topology Examples



Grid



Torus



Hypercube

Criteria	Bus	Ring	2Dtorus	Hypercube	Fully connected
64 nodes					
Performance					
Diameter	1	32	8	6	1
Bisection BW	1	2	16	32	1024
Cost					
Ports/switch		3	5	7	64
Total links	1	64	128	192	2016

k-ary d-Cube

- Consider a k-ary d-cube: a d-dimension array with k elements in each dimension, there are links between elements that differ in one dimension by 1 (mod k)

- Number of nodes $N = k^d$

Number of switches :	N	Avg. routing distance:	$d(k-1)/4$
Switch degree :	$2d + 1$	Diameter :	$d(k-1)/2$
Number of links :	Nd	Bisection bandwidth :	$2wk^{d-1}$
Pins per node :	$2wd$	Switch complexity :	$(2d + 1)^2$

The switch degree, num links, pins per node, bisection bw for a hypercube are half of what is listed above (diam and avg routing distance are twice, switch complexity is $(d + 1)^2$) because unlike the other cases, a hypercube does not have right and left neighbors.

32

Should we minimize or maximize dimension?

Problem 1

Assume that a server consumes 100W at peak utilization and 50W at zero utilization. Assume a linear relationship between utilization and power. The server is capable of executing many threads in parallel. Assume that a single thread utilizes 25% of all server resources (functional units, caches, memory capacity, memory bandwidth, etc.). What is the total power dissipation when executing 99 threads on a collection of these servers, such that performance and energy are close to optimal?

For near-optimal performance and energy, use 25 servers. 24 servers at 100% utilization, executing 96 threads, consuming 2400W. The 25th server will run the last 3 threads and consume 87.5~W.

RAID 4 and RAID 5

- Data is block interleaved – this allows us to get all our data from a single disk on a read – in case of a disk error, read all 9 disks
- Block interleaving reduces thruput for a single request (as only a single disk drive is servicing the request), but improves task-level parallelism as other disk drives are free to service other requests
- On a write, we access the disk that stores the data and the parity disk – parity information can be updated simply by checking if the new data differs from the old data

