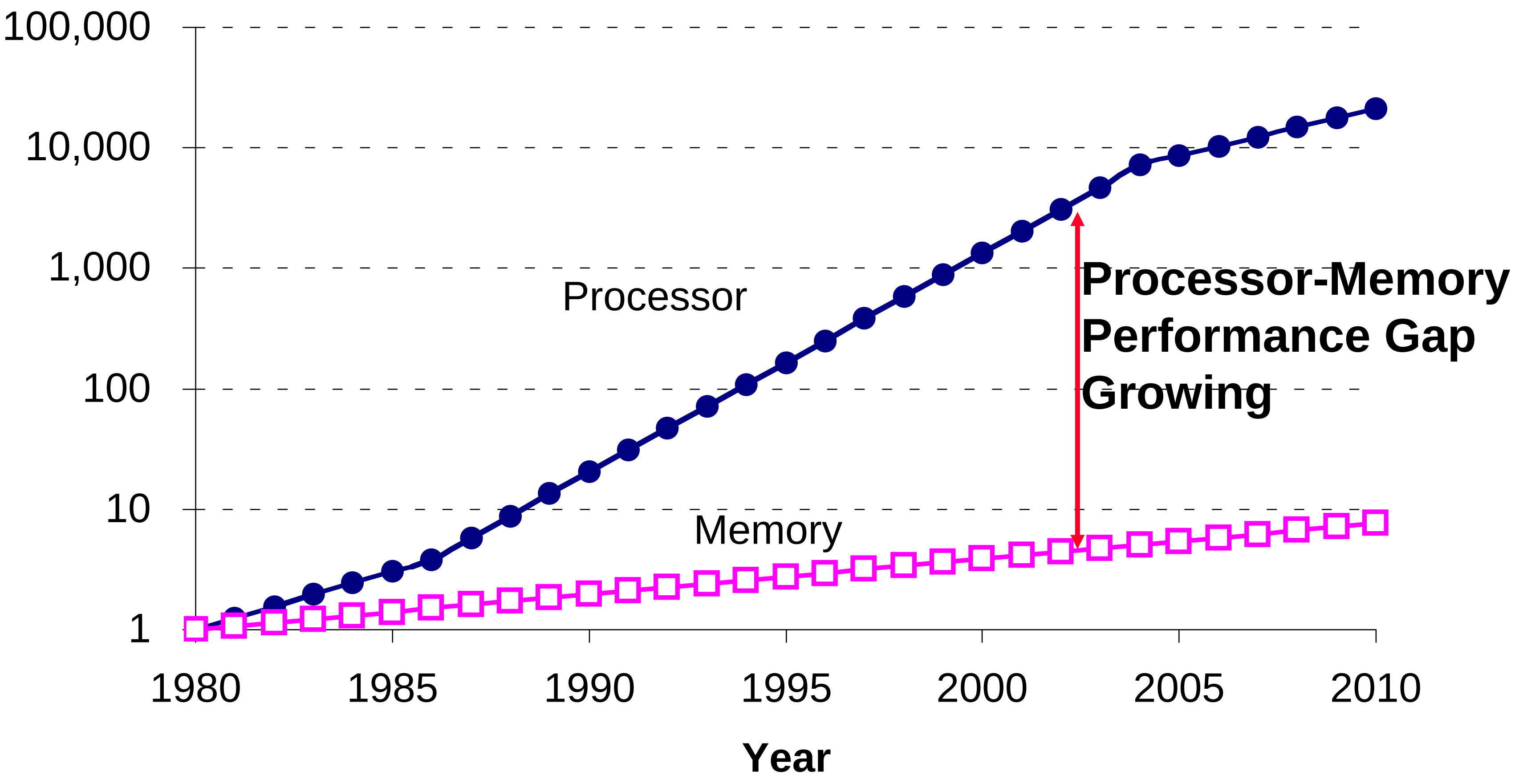
西安交通大學
XI'AN JIAOTONG UNIVERSITY

Modern Computer Architecture

Lecture4 Memory Hierarchy Design

Hongbin Sun
国家集成电路人才培养基地
Xi'an Jiaotong University

The Memory Wall

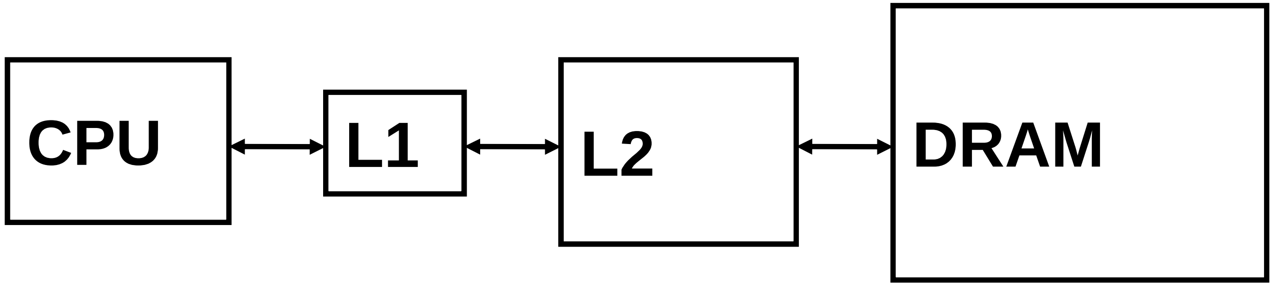


Main Memory Background

- **Performance of Main Memory:**
  - **Latency: Cache Miss Penalty**
    - *Access Time*: time between request and word arrives
    - *Cycle Time*: time between requests
  - **Bandwidth: I/O & Large Block Miss Penalty (L2)**
- **Main Memory is *DRAM*: Dynamic Random Access Memory**
  - **Dynamic since needs to be refreshed periodically (8 ms, 1% time)**
  - **Addresses divided into 2 halves (Memory as a 2D matrix):**
    - *RAS* or *Row Access Strobe*
    - *CAS* or *Column Access Strobe*
- **Cache uses *SRAM*: Static Random Access Memory**
  - **No refresh (6 transistors/bit vs. 1 transistor**
    ***Size*: DRAM/SRAM *4-8*,**
    Cost/Cycle time*: SRAM/DRAM *8-16

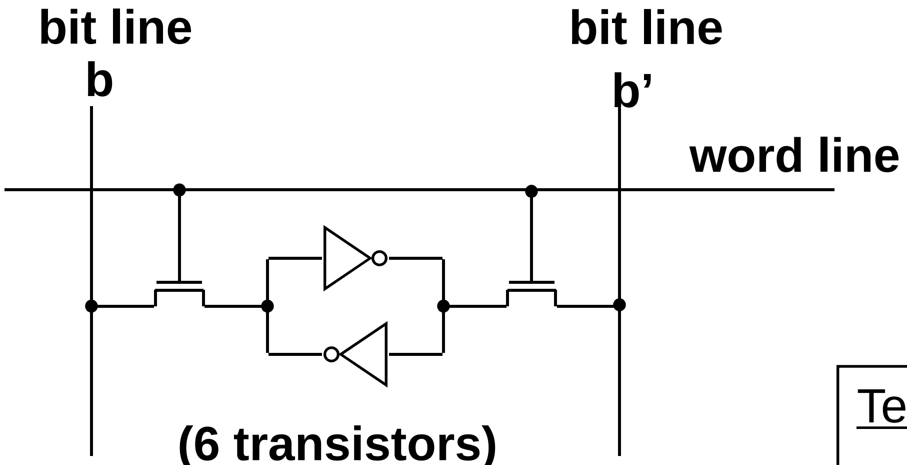
Multilevel Caches

- **A memory cannot be large and fast**
- **Increasing sizes of cache at each level**



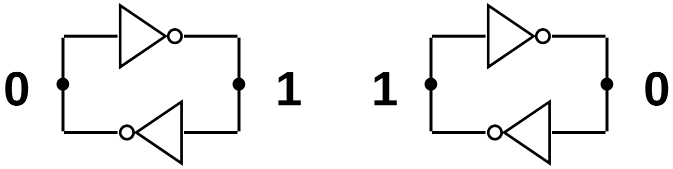
Local miss rate = misses in cache / accesses to cache
Global miss rate = misses in cache / CPU memory accesses
Misses per instruction = misses in cache / number of instructions

Anatomy of an SRAM Cell



(6 transistors)

Stable Configurations



Terminology:

bit line: carries data

word line: used for addressing

Write:

1. set bit lines to new data value
   - **b’** is set to the opposite of **b**
2. raise word line to “high”

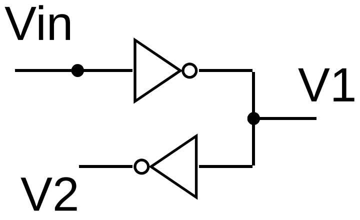
⇒ sets cell to new state (may involve flipping relative to old state)

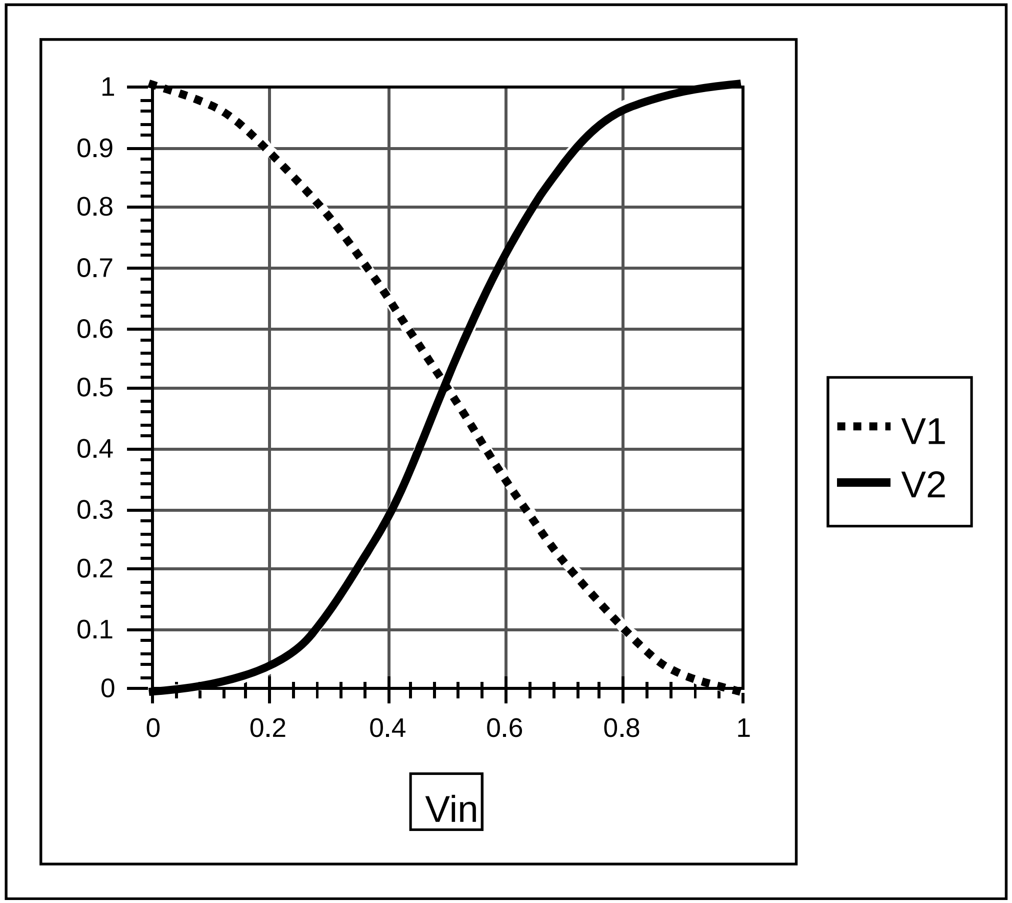
Read:

1. set bit lines high
2. set word line high
3. see which bit line goes low

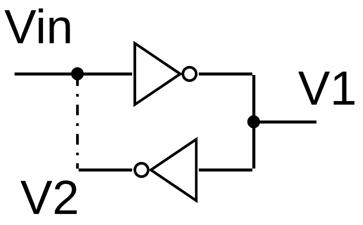
SRAM Cell Principle

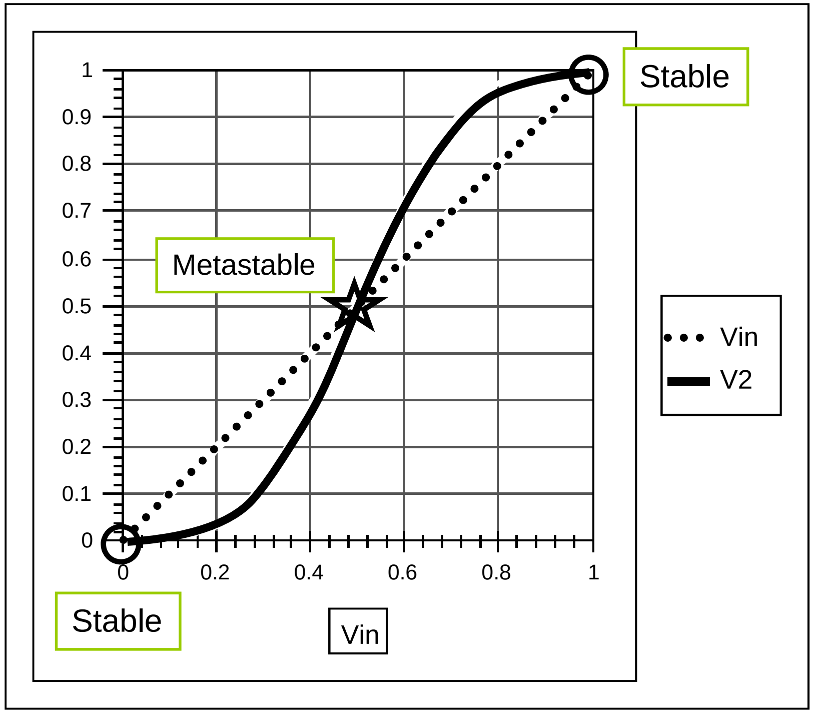
- **Inverter Amplifies**
  - **Negative gain**
  - **Slope < –1 in middle**
  - **Saturates at ends**
- **Inverter Pair Amplifies**
  - **Positive gain**
  - **Slope > 1 in middle**
  - **Saturates at ends**



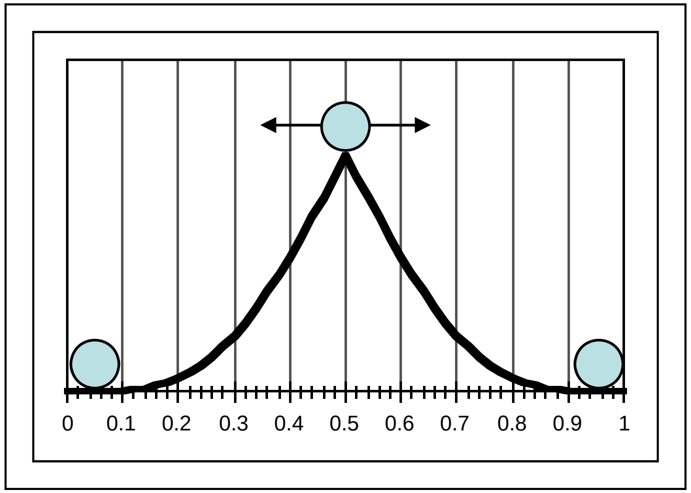


Bistable Element

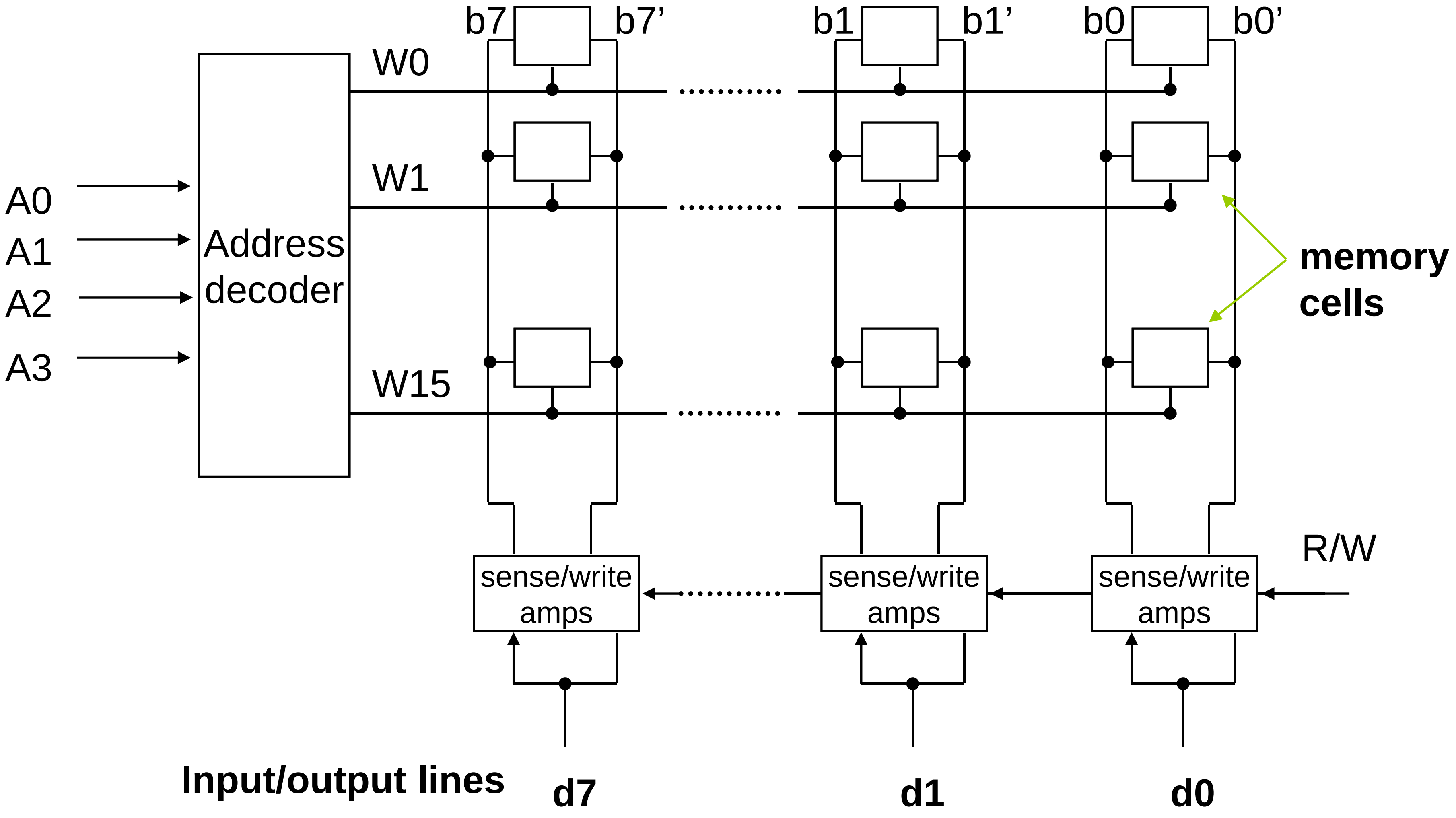




- **Stability**
  - **Require Vin = V2**
  - **Stable at endpoints**
    - recover from pertubation
  - **Metastable in middle**
    - Fall out when perturbed
- **Ball on Ramp Analogy**



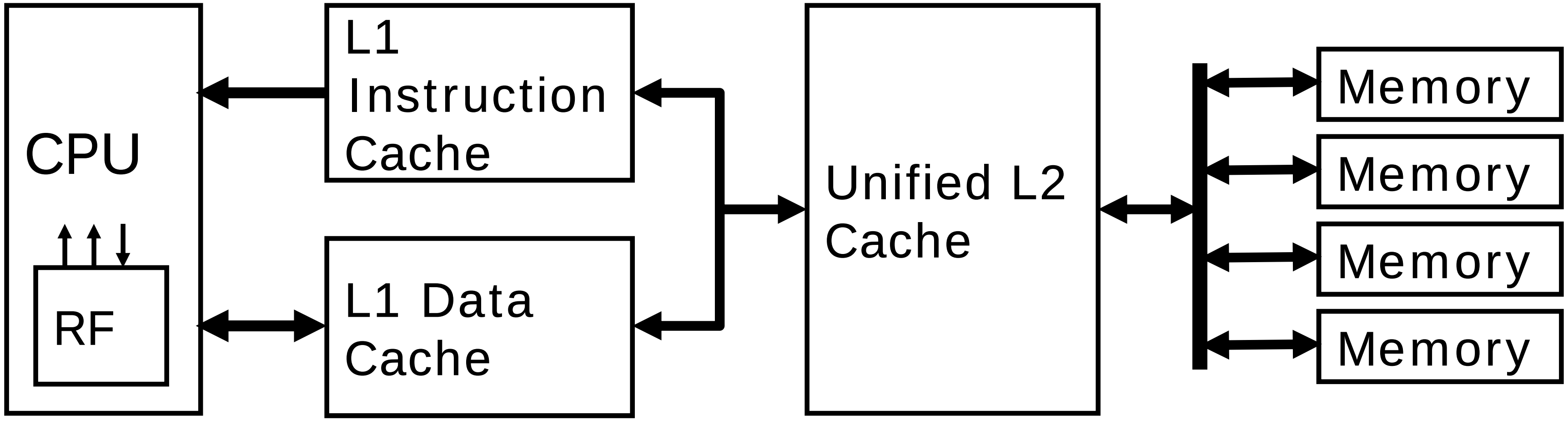
Example SRAM Configuration (16 x 8)



A Typical Memory Hierarchy c.2007

Split instruction & data primary caches (on-chip SRAM)

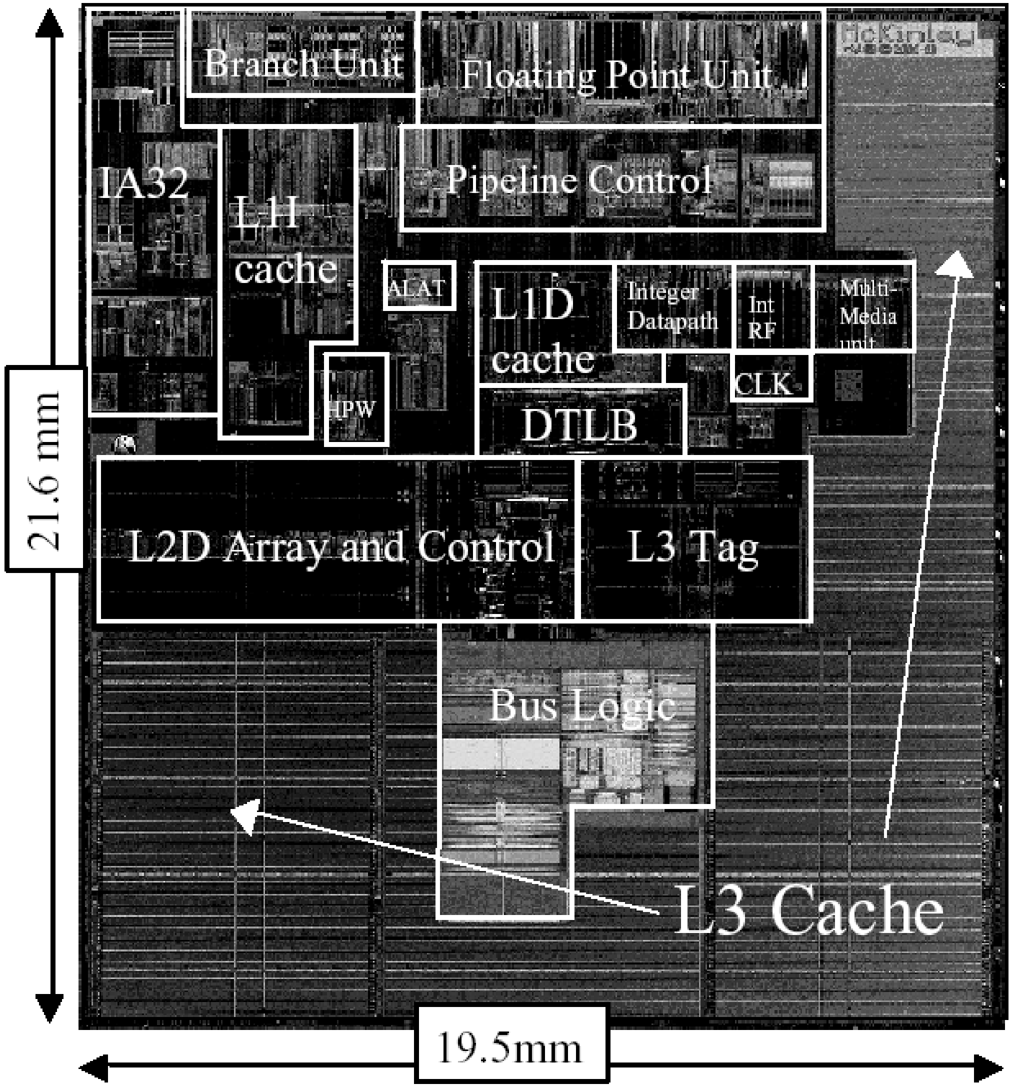
Multiple interleaved memory banks (off-chip DRAM)



Multiported register file (part of CPU)

Large unified secondary cache (on-chip SRAM)

Itanium-2 On-Chip Caches
(Intel/HP, 2002)



Level 1, 16KB, 4-way s.a., 64B line, quad-port (2 load+2 store), single cycle latency

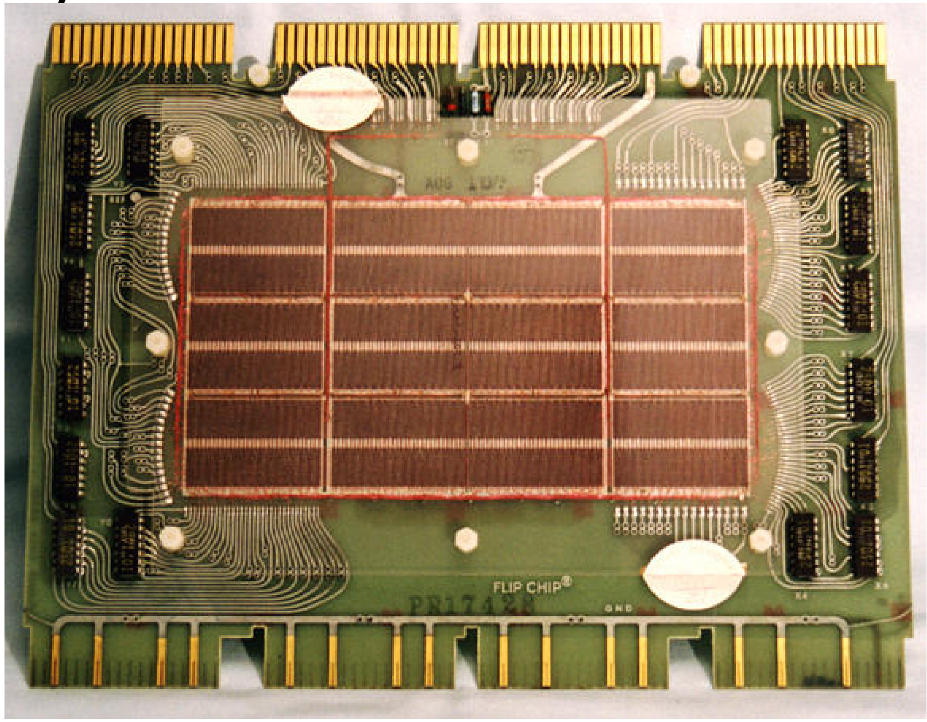
Level 2, 256KB, 4-way s.a, 128B line, quad-port (4 load or 4 store), five cycle latency

Level 3, 3MB, 12-way s.a., 128B line, single 32B port, twelve cycle latency

Core Memory

- **Core memory was first large-scale reliable main memory**
  - **invented by Forrester in late 40s/early 50s at MIT for Whirlwind project**
- **Bits stored as magnetization polarity on small ferrite cores threaded onto 2-dimensional grid of wires**
- **Coincident current pulses on X and Y wires would write cell and also sense original state (destructive reads)**

- Robust, non-volatile storage
- Used on space shuttle computers until recently
- Cores threaded onto wires by hand (25 billion a year at peak production)
- Core access time ~ 1μs



DEC PDP-8/E Board,
4K words x 12 bits,
(1968)

Semiconductor Memory, DRAM

- **Semiconductor memory began to be competitive in early 1970s**
  - **Intel formed to exploit market for semiconductor memory**

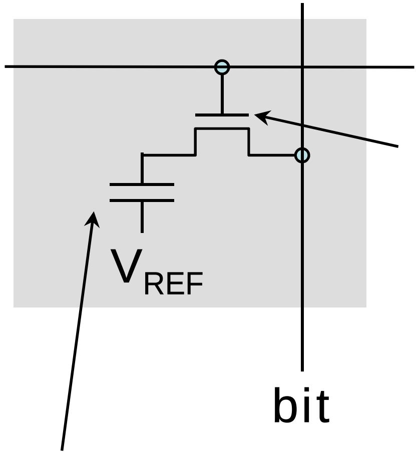
- **First commercial DRAM was Intel 1103**
  - **1Kbit of storage on single chip**
  - **charge on a capacitor used to hold value**

- **Semiconductor memory quickly replaced core in 1970s**

- **Today (September 2007), 1GB DRAM < $30**
  - **Individuals can easily afford to fill a 32-bit address space with DRAM (4GB)**

One Transistor Dynamic RAM

1-T DRAM Cell



word

access transistor

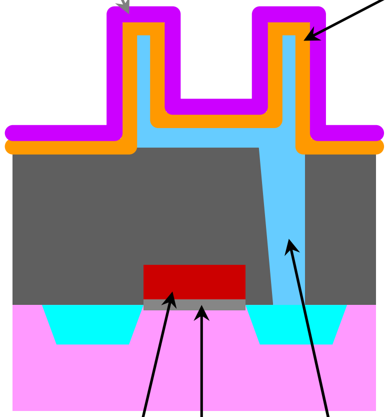
V_{REF}

bit

Storage capacitor (FET gate, trench, stack)

TiN top electrode (V_{REF})

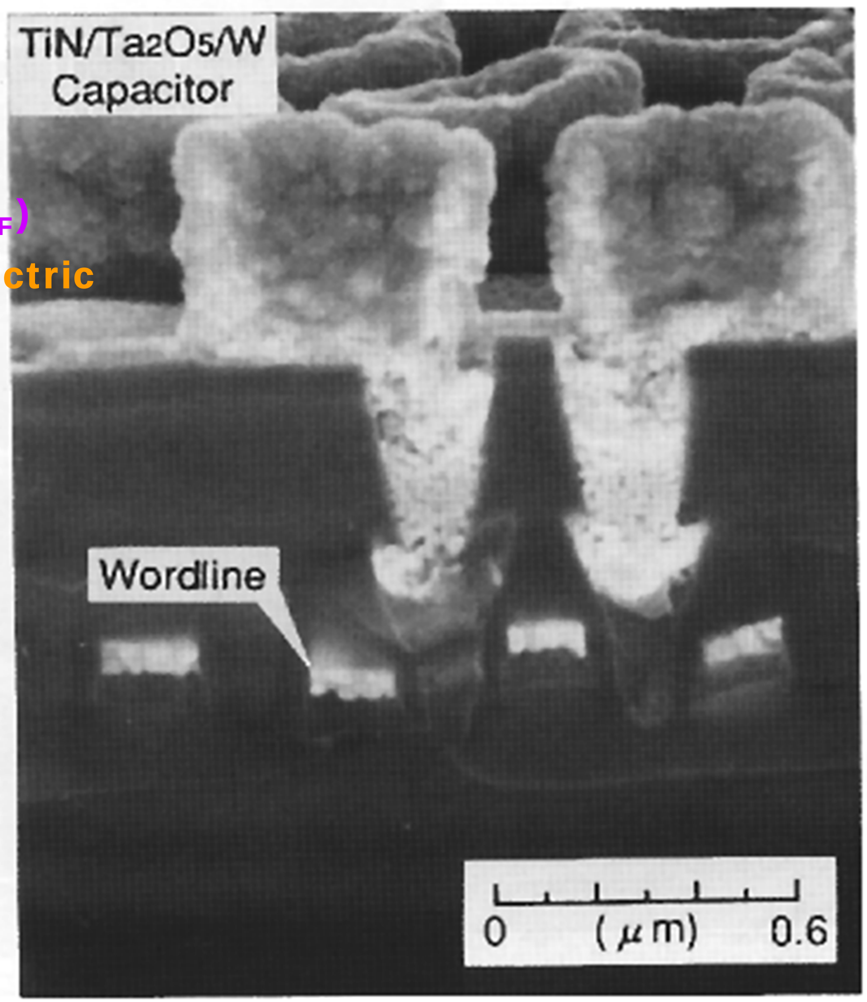
Ta_2O_5 dielectric



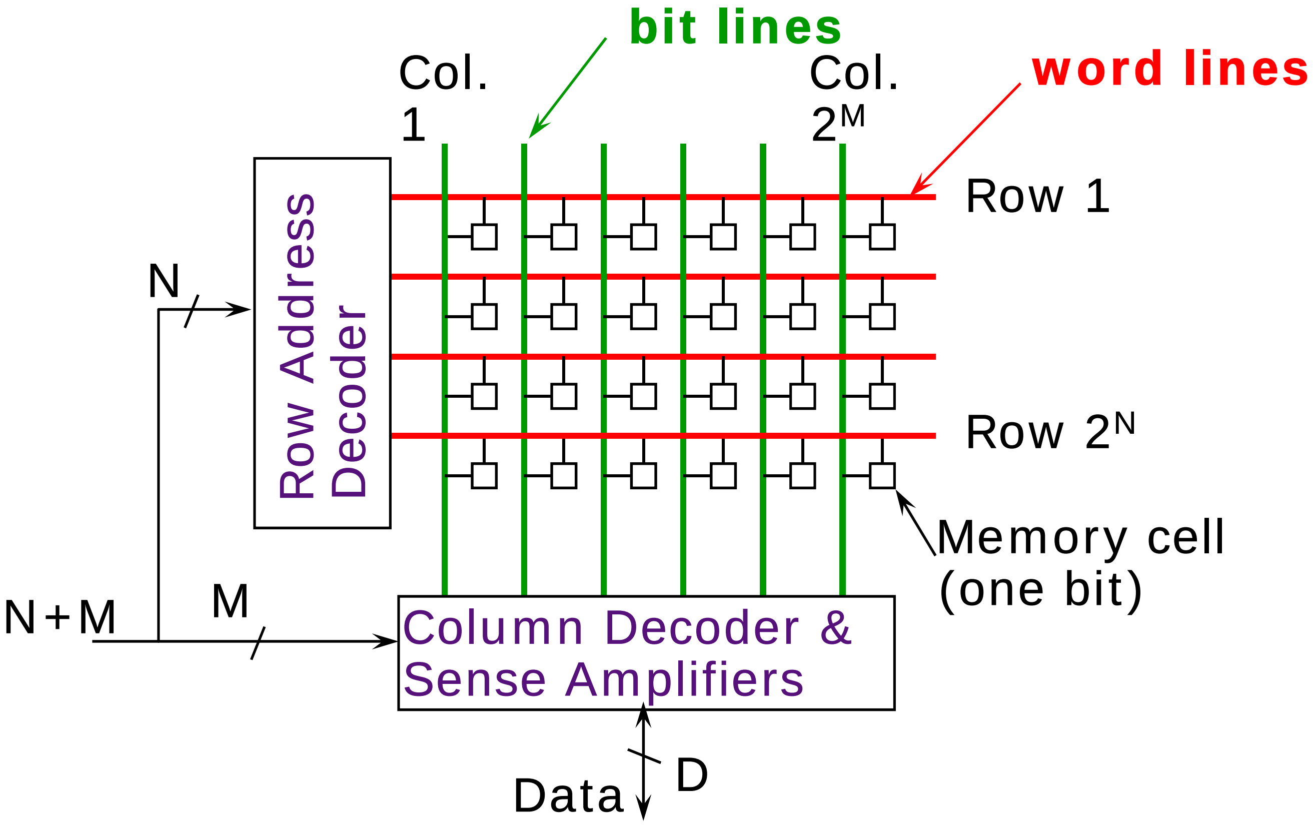
poly word line

access transistor

W bottom electrode



DRAM Architecture



- Bits stored in 2-dimensional arrays on chip
- Modern chips have around 4 logical banks on each chip
  - each logical bank physically implemented as many smaller arrays

DRAM Operation

Three steps in read/write access to a given bank

- **Row access (RAS)**
  - decode row address, enable addressed row (often multiple Kb in row)
  - bitlines share charge with storage cell
  - small change in voltage detected by sense amplifiers which latch whole row of bits
  - sense amplifiers drive bitlines full rail to recharge storage cells
- **Column access (CAS)**
  - decode column address to select small number of sense amplifier latches (4, 8, 16, or 32 bits depending on DRAM package)
  - on read, send latched bits out to chip pins
  - on write, change sense amplifier latches. which then charge storage cells to required value
  - can perform multiple column accesses on same row without another row access (burst mode)
- **Precharge**
  - charges bit lines to known value, required before next row access

- **Each step has a latency of around 10-20ns in modern DRAMs.**
- **Various DRAM standards (DDR, RDRAM) have different ways of encoding the signals for transmission to the DRAM, but all share same core architecture**
- **Can overlap RAS/CAS/Precharge in different banks to increase bandwidth**

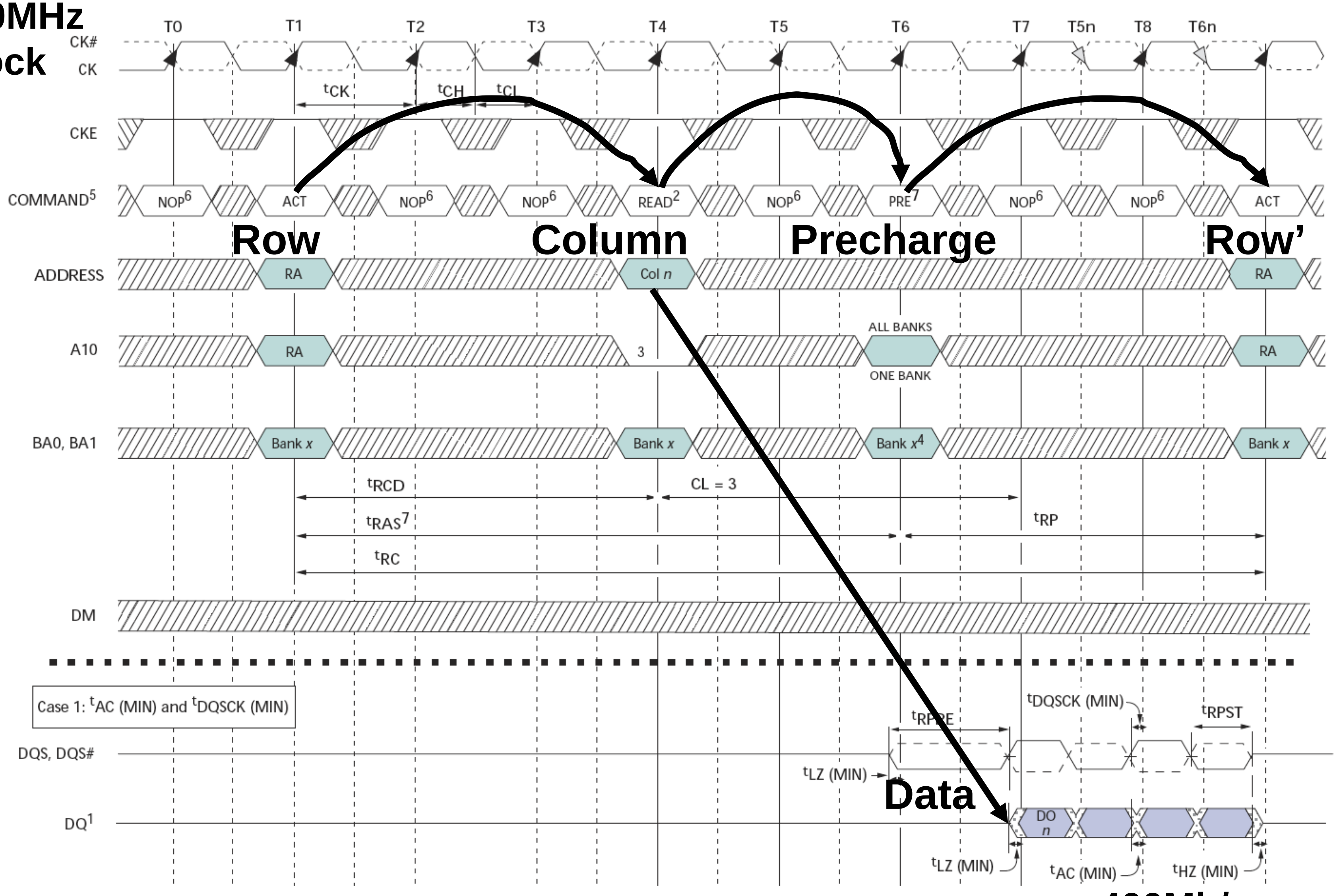
Quest for DRAM Performance

1. **Fast Page mode**
   - Add timing signals that allow repeated accesses to row buffer without another row access time
   - Such a buffer comes naturally, as each array will buffer 1024 to 2048 bits for each access
2. **Synchronous DRAM (SDRAM)**
   - Add a clock signal to DRAM interface, so that the repeated transfers would not bear overhead to synchronize with DRAM controller
3. **Double Data Rate (DDR SDRAM)**
   - Transfer data on both the rising edge and falling edge of the DRAM clock signal ⇒ doubling the peak data rate
   - DDR2 lowers power by dropping the voltage from 2.5 to 1.8 volts + offers higher clock rates: up to 533 MHz
   - DDR3 drops to 1.5 volts + higher clock rates: up to 800 MHz

- **Improved Bandwidth, not Latency**

Double-Data Rate (DDR2) DRAM

200MHz
Clock



Row

Column

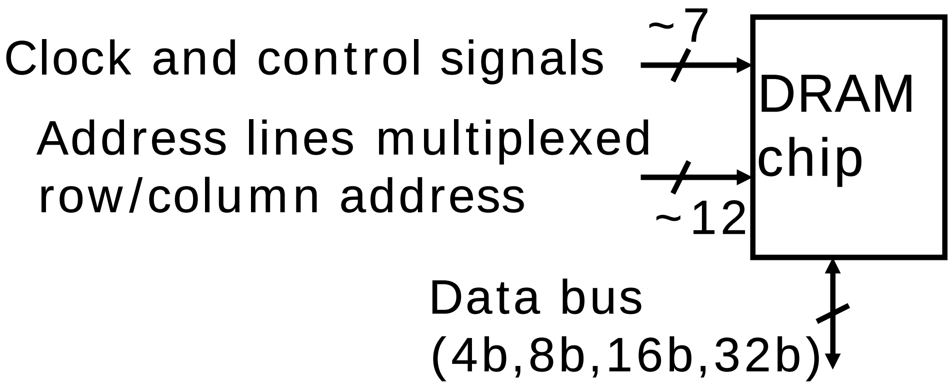
Precharge

Row’

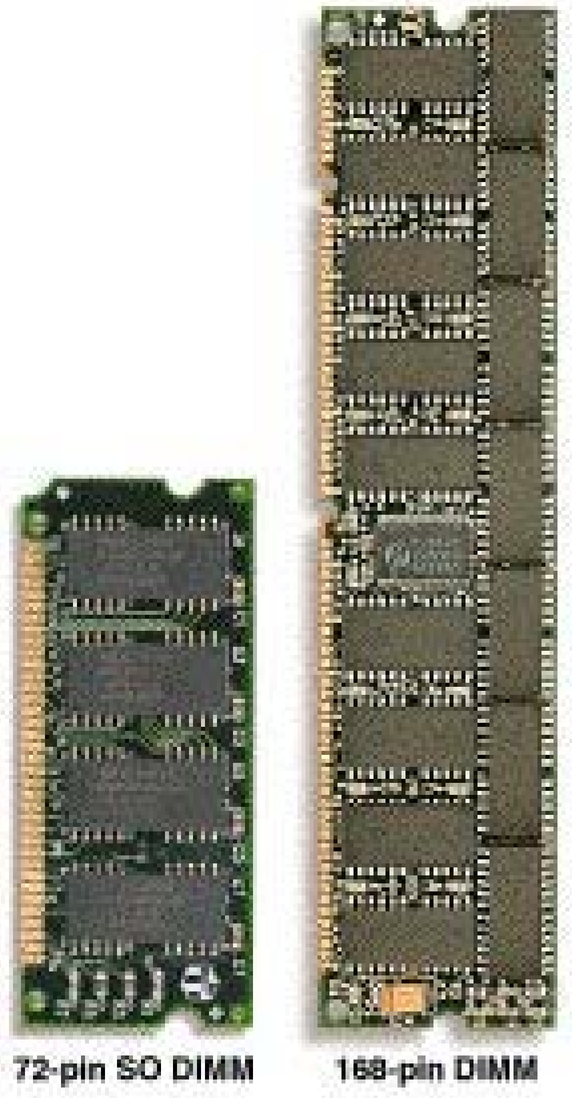
Data

400Mb/s
Data Rate

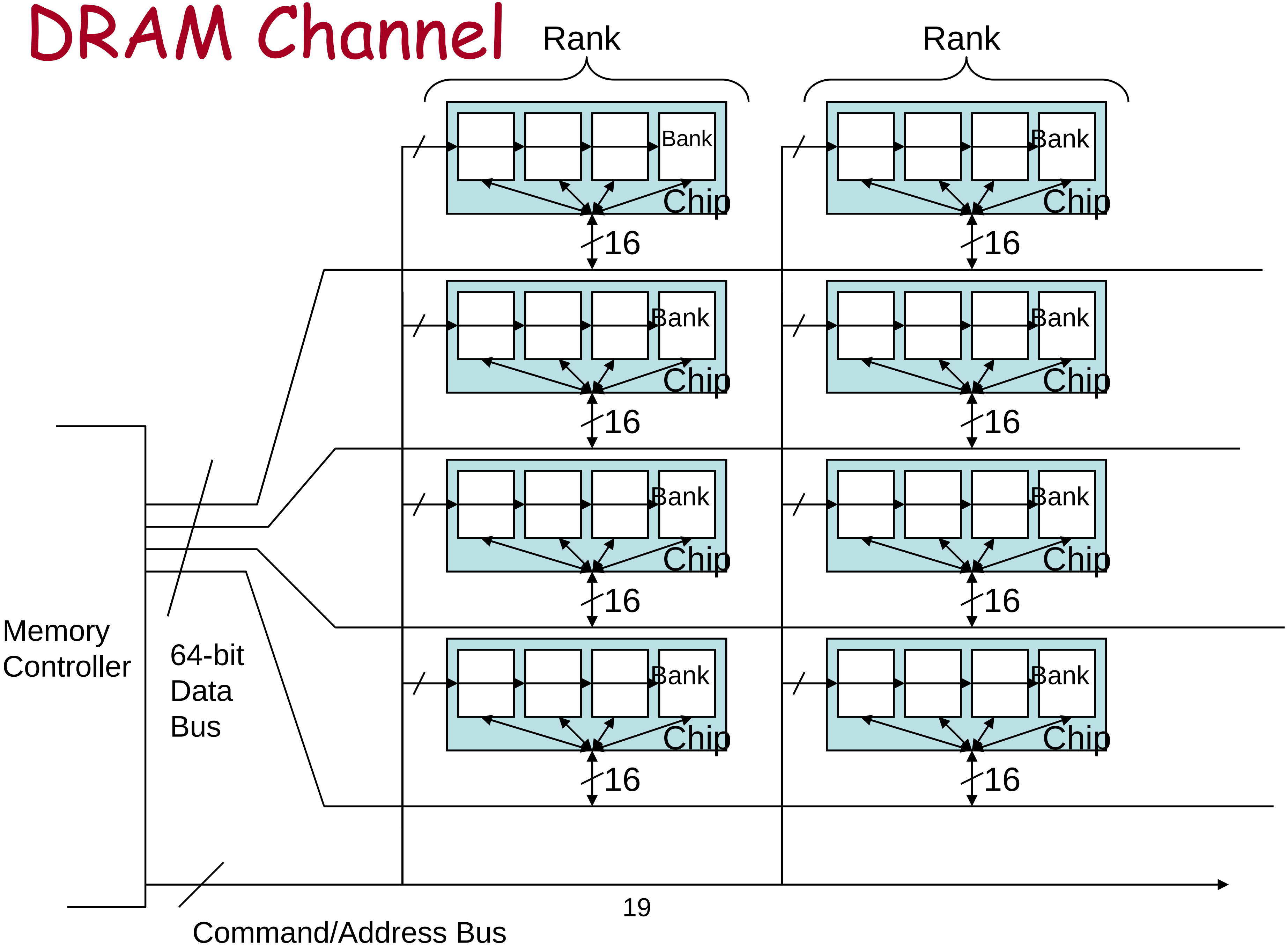
DRAM Packaging


Clock and control signals
~7
Address lines multiplexed row/column address
~12
DRAM chip
Data bus (4b,8b,16b,32b)

- DIMM (Dual Inline Memory Module) contains multiple chips arranged in “ranks”
- Each rank has clock/control/address signals connected in parallel (sometimes need buffers to drive signals to all chips), and data pins work together to return wide word
  - e.g., a rank could implement a 64-bit data bus using 16x4-bit chips, or a 64-bit data bus using 8x8-bit chips.
- A modern DIMM usually has one or two ranks (occasionally 4 if high capacity)
  - A rank will contain the same number of banks as each constituent chip (e.g., 4-8)



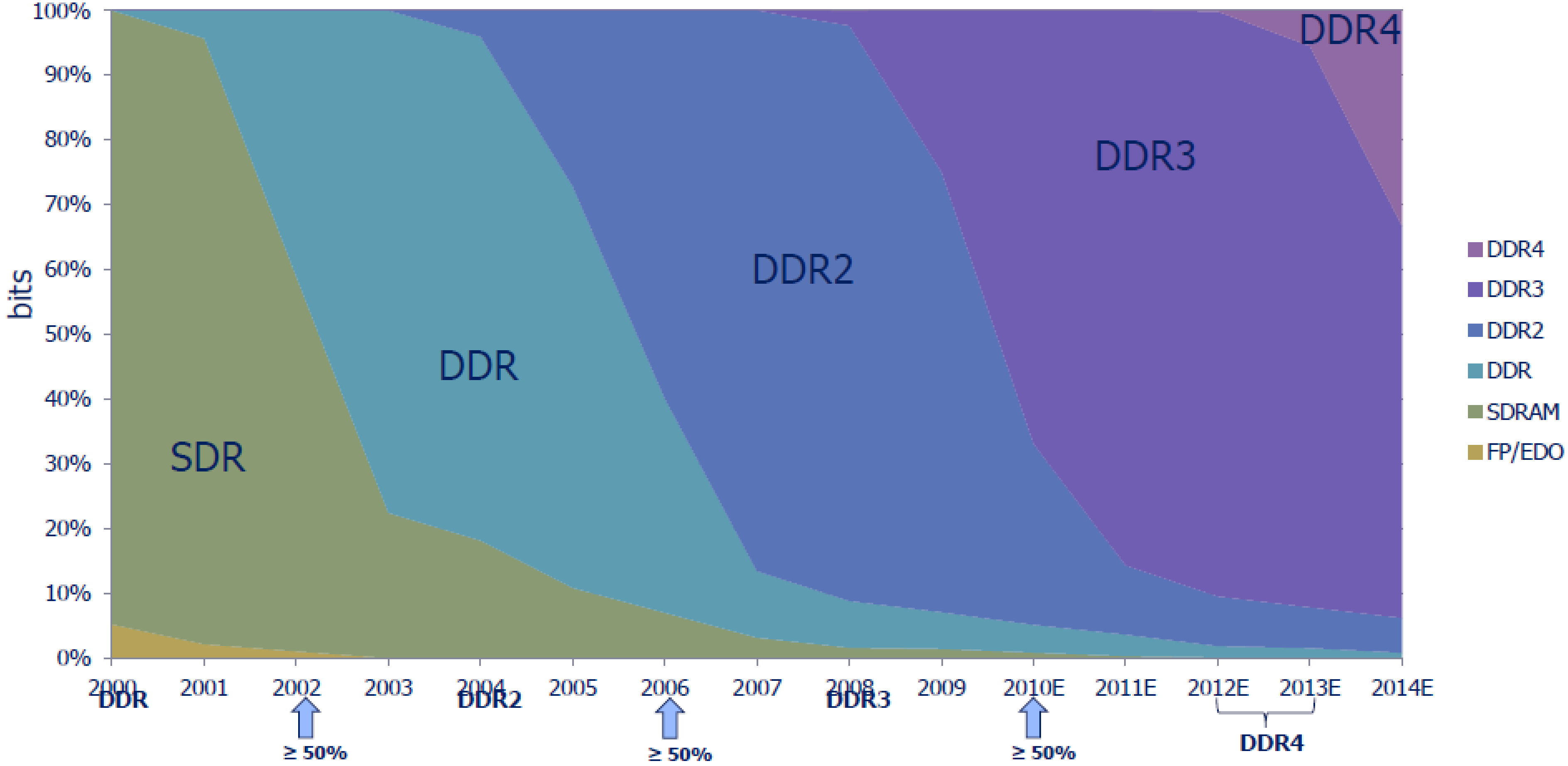
DRAM Channel



Need for Error Correction!

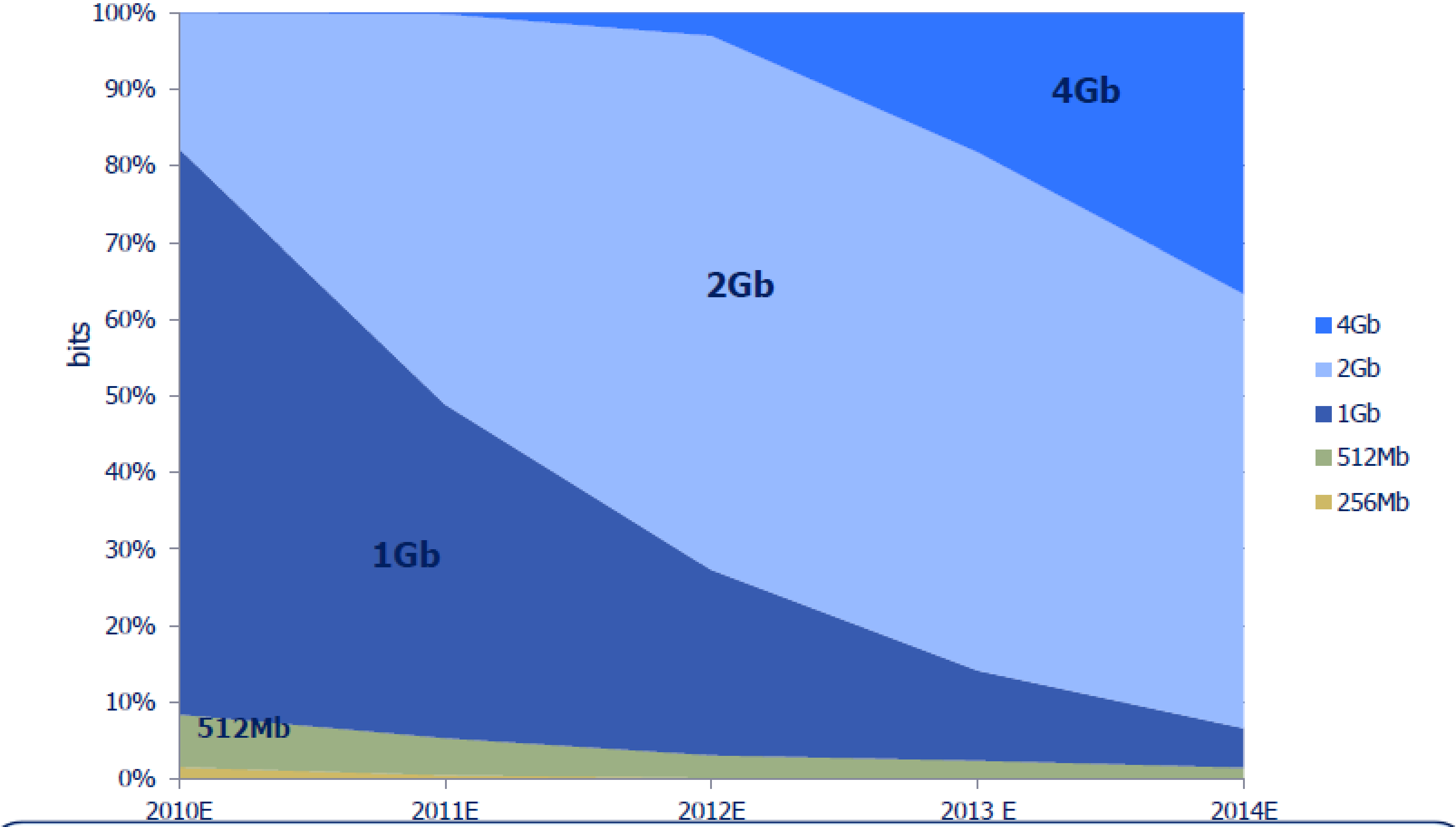
- **Motivation:**
  - **Failures/time *proportional* to number of bits!**
  - **As DRAM cells shrink, more vulnerable**
- **Went through period in which failure rate was low enough without error correction that people didn't do correction**
  - **DRAM banks too large now**
  - **Servers always corrected memory systems**
- **Basic idea: add redundancy through parity bits**
  - **Common configuration: Random error correction**
    - SEC-DED (single error correct, double error detect)
    - One example: 64 data bits + 8 parity bits (11% overhead)
  - **Really want to handle failures of physical components as well**
    - Organization is multiple DRAMs/DIMM, multiple DIMMs
    - Want to recover from failed DRAM and failed DIMM!
    - "Chip kill" handle failures width of single DRAM chip

DRAM Technology Trend



- ➢ Technology cadence is roughly 4 years
- ➢ Technology becomes mainstream (≥ 50% bits) roughly 2 years after production start

DRAM Density Trend



- 1Gb & 2Gb will be dominant densities in 2011 due to popularity of 1GB/2GB/4GB modules
- 1Gb & 2Gb driven by DDR2 & DDR3
- 4Gb driven by DDR3 & DDR4

Speed-Density-Power Comparison

	Freq. Range (MHz)	Bus Width (per device)	Max. Bandwidth (burst rate)	Transfer rate per pin	Density	Row Cycle Time (tRC)	Max Power
SDRAM	100-200	x4, x8, x16, x32	400 MB/s	100-200Mb/s	64Mb - 512Mb	66ns	1W
DDR1	100-200	x4, x8, x16	800 MB/s	200-400Mb/s	128Mb-1Gb	60ns	1W
DDR2	200-400	x4, x8, x16	1.6 GB/s	400-800Mb/s	256Mb-2Gb	55ns	700mW
DDR3	400-1066	x4, x8, x16	3.2 GB/s	800-1600Mb/s	1Gb, 2Gb	48ns	500mW
DDR3L	400-800	x4, x8, x16	3.2 GB/s	800-1600Mb/s	1Gb, 2Gb	48ns	440mW
DDR4	667-1600	x4, x8, x16, x32	12.8 GB/s	1333-3200Mb/s	4-8Gb	TBD<45 ns	TBD-330mW
SDR LPDRAM	100-167	X16, x32	400-667 MB/s	200-333Mb/s	128-512Mb	45-50ns	150-230mW
DDR LPDRAM	100-167	X16, x32	400-667 MB/s	200-333Mb/s	128-512Mb	45-50ns	150-230mW
DDR2 LPDRAM	333-400	X16, x32	667-800	667-800Mb/s	2-8Gb	55ns	200mW

- LPDRAM offers better power consideration but requires a price trade-off consideration
- Technology migration (SDR→DDR→DDR2→DDR3→DDR4) can improve on power/ performance/ density

DRAM Technology Comparison

DDR4 combines features from DDR3 & GDDR5 ➔ "Best of both Worlds"

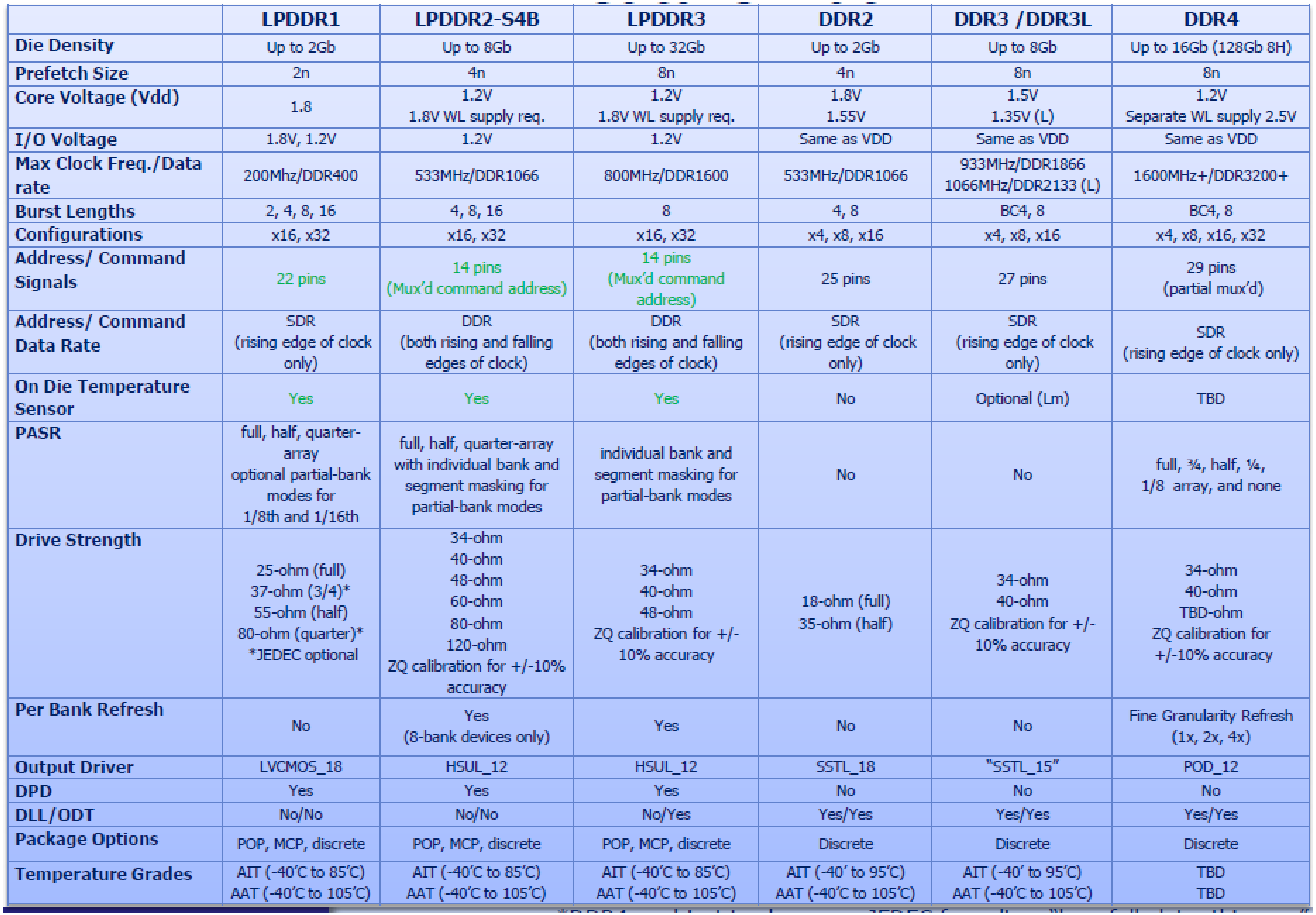
	DDR3	DDR4	GDDR5
Voltage	1.5V/1.35V	1.2V	1.5V/1.35V
Strobe	Bi-directional Differential	Bi-directional Differential	Free Running Differential WRITE Clock
Strobe Config	Per Byte	Per Byte	Per Word
READ Data Capture	Strobe based	Strobe based	Clock Data Recovery
Data Termination	VddQ/2	VddQ	VddQ
Add/Cmd Termination	VddQ/2	VddQ/2	VddQ
Burst Length	BC4, 8	BC4, 8	8
Bank Grouping	No	4 - Bank Groups	4 - Bank Groups
On Chip Error Detection	No	Command/Address Parity	
		CRC for Data bus	CRC for Data bus
Configuration	x4, x8, x16	x4, x8, x16	x16 / x32
Package	78 ball / 96 ball FBGA	78 ball / 96 ball FBGA	170 Ball FBGA
Data Rate (Mbps/pin)	800 – 2133	1600 – 3200+	4000 – 7000
Component Density	1Gb - 8Gb	2Gb - 16Gb	512Mb-2Gb
Stacking Options	DDP, QDP	up to 8H (128Gb stack); single load	No

Power

Performance

Cost

DRAM Feature Matrix



	LPDDR1	LPDDR2-S4B	LPDDR3	DDR2	DDR3 /DDR3L	DDR4
Die Density	Up to 2Gb	Up to 8Gb	Up to 32Gb	Up to 2Gb	Up to 8Gb	Up to 16Gb (128Gb 8H)
Prefetch Size	2n	4n	8n	4n	8n	8n
Core Voltage (Vdd)	1.8	1.2V 1.8V WL supply req.	1.2V 1.8V WL supply req.	1.8V 1.55V	1.5V 1.35V (L)	1.2V Separate WL supply 2.5V
I/O Voltage	1.8V, 1.2V	1.2V	1.2V	Same as VDD	Same as VDD	Same as VDD
Max Clock Freq./Data rate	200Mhz/DDR400	533MHz/DDR1066	800MHz/DDR1600	533MHz/DDR1066	933MHz/DDR1866 1066MHz/DDR2133 (L)	1600MHz+/DDR3200+
Burst Lengths	2, 4, 8, 16	4, 8, 16	8	4, 8	BC4, 8	BC4, 8
Configurations	x16, x32	x16, x32	x16, x32	x4, x8, x16	x4, x8, x16	x4, x8, x16, x32
Address/ Command Signals	22 pins	14 pins (Mux'd command address)	14 pins (Mux'd command address)	25 pins	27 pins	29 pins (partial mux'd)
Address/ Command Data Rate	SDR (rising edge of clock only)	DDR (both rising and falling edges of clock)	DDR (both rising and falling edges of clock)	SDR (rising edge of clock only)	SDR (rising edge of clock only)	SDR (rising edge of clock only)
On Die Temperature Sensor	Yes	Yes	Yes	No	Optional (Lm)	TBD
PASR	full, half, quarter-array optional partial-bank modes for 1/8th and 1/16th	full, half, quarter-array with individual bank and segment masking for partial-bank modes	individual bank and segment masking for partial-bank modes	No	No	full, ¾, half, ¼, 1/8 array, and none
Drive Strength	25-ohm (full) 37-ohm (3/4)* 55-ohm (half) 80-ohm (quarter)* *JEDEC optional	34-ohm 40-ohm 48-ohm 60-ohm 80-ohm 120-ohm ZQ calibration for +/-10% accuracy	34-ohm 40-ohm 48-ohm ZQ calibration for +/-10% accuracy	18-ohm (full) 35-ohm (half)	34-ohm 40-ohm ZQ calibration for +/-10% accuracy	34-ohm 40-ohm TBD-ohm ZQ calibration for +/-10% accuracy
Per Bank Refresh	No	Yes (8-bank devices only)	Yes	No	No	Fine Granularity Refresh (1x, 2x, 4x)
Output Driver	LVCMOS_18	HSUL_12	HSUL_12	SSTL_18	"SSTL_15"	POD_12
DPD	Yes	Yes	Yes	No	No	No
DLL/ODT	No/No	No/No	No/Yes	Yes/Yes	Yes/Yes	Yes/Yes
Package Options	POP, MCP, discrete	POP, MCP, discrete	POP, MCP, discrete	Discrete	Discrete	Discrete
Temperature Grades	AIT (-40'C to 85'C) AAT (-40'C to 105'C)	AIT (-40'C to 85'C) AAT (-40'C to 105'C)	AIT (-40'C to 85'C) AAT (-40'C to 105'C)	AIT (-40' to 95'C) AAT (-40'C to 105'C)	AIT (-40' to 95'C) AAT (-40'C to 105'C)	TBD TBD

11 Advanced Cache Optimizations

- Reducing hit time

1. Small and simple caches
2. Way prediction
3. Trace caches

- Increasing cache bandwidth

4. Pipelined caches
5. Multibanked caches
6. Nonblocking caches

- Reducing Miss Penalty

7. Critical word first
8. Merging write buffers

- Reducing Miss Rate

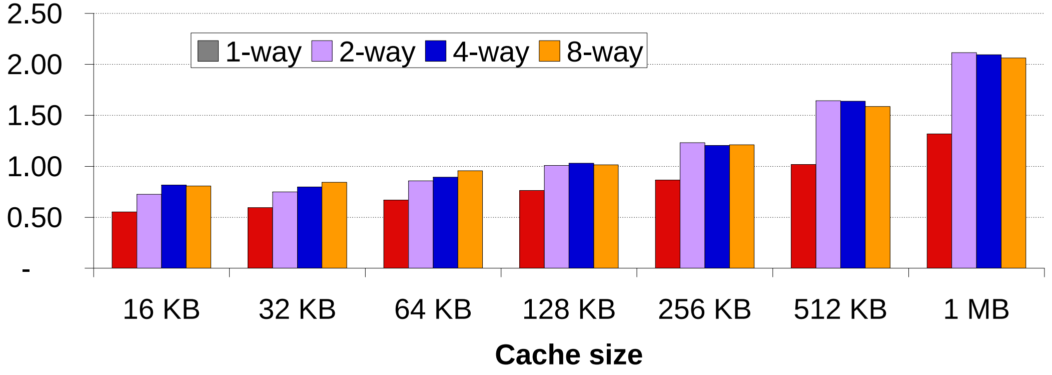
9. Compiler optimizations

- Reducing miss penalty or miss rate via parallelism

10. Hardware prefetching
11. Compiler prefetching

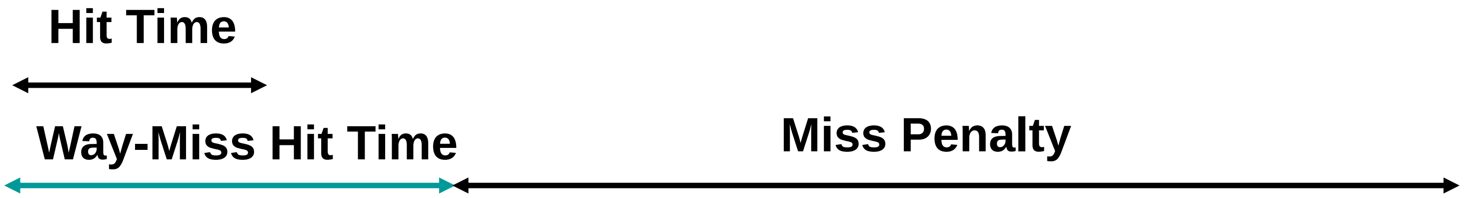
1. Fast Hit times via Small and Simple Caches

- Index tag memory and then compare takes time
- ⇒ Small cache can help hit time since smaller memory takes less time to index
  - E.g., L1 caches same size for 3 generations of AMD microprocessors: K6, Athlon, and Opteron
  - Also L2 cache small enough to fit on chip with the processor avoids time penalty of going off chip
- Simple ⇒ direct mapping
  - Can overlap tag check with data transmission since no choice
- Access time estimate for 90 nm using CACTI model 4.0
  - Median ratios of access time relative to the direct-mapped caches are 1.32, 1.39, and 1.43 for 2-way, 4-way, and 8-way caches



2. Fast Hit times via Way Prediction

- How to combine fast hit time of Direct Mapped and have the lower conflict misses of 2-way SA cache?
- Way prediction: keep extra bits in cache to predict the "way," or block within the set, of next cache access.
  - Multiplexor is set early to select desired block, only 1 tag comparison performed that clock cycle in parallel with reading the cache data
  - Miss ⇒ 1st check other blocks for matches in next clock cycle



Hit Time

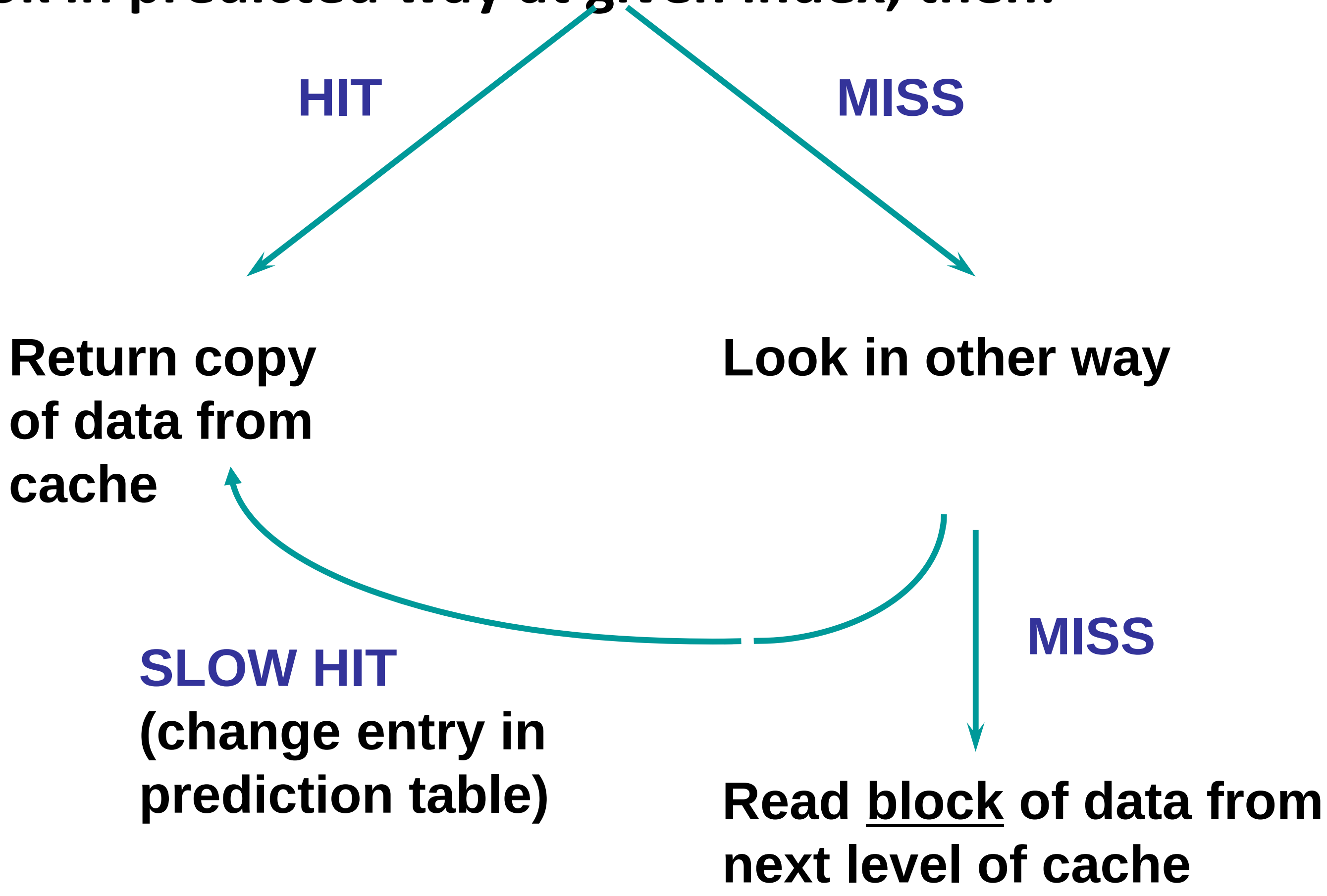
Way-Miss Hit Time

Miss Penalty

- Accuracy ≈ 85%
- Drawback: CPU pipeline is hard if hit takes 1 or 2 cycles
  - Used for instruction caches vs. L1 data caches
  - Also used on MIPS R10K for off-chip L2 unified cache, way-prediction table on-chip

Way Predicting Caches
(MIPS R10000 L2 cache)

- **Use processor address to index into way prediction table**
- **Look in predicted way at given index, then:**



HIT

Return copy of data from cache

MISS

Look in other way

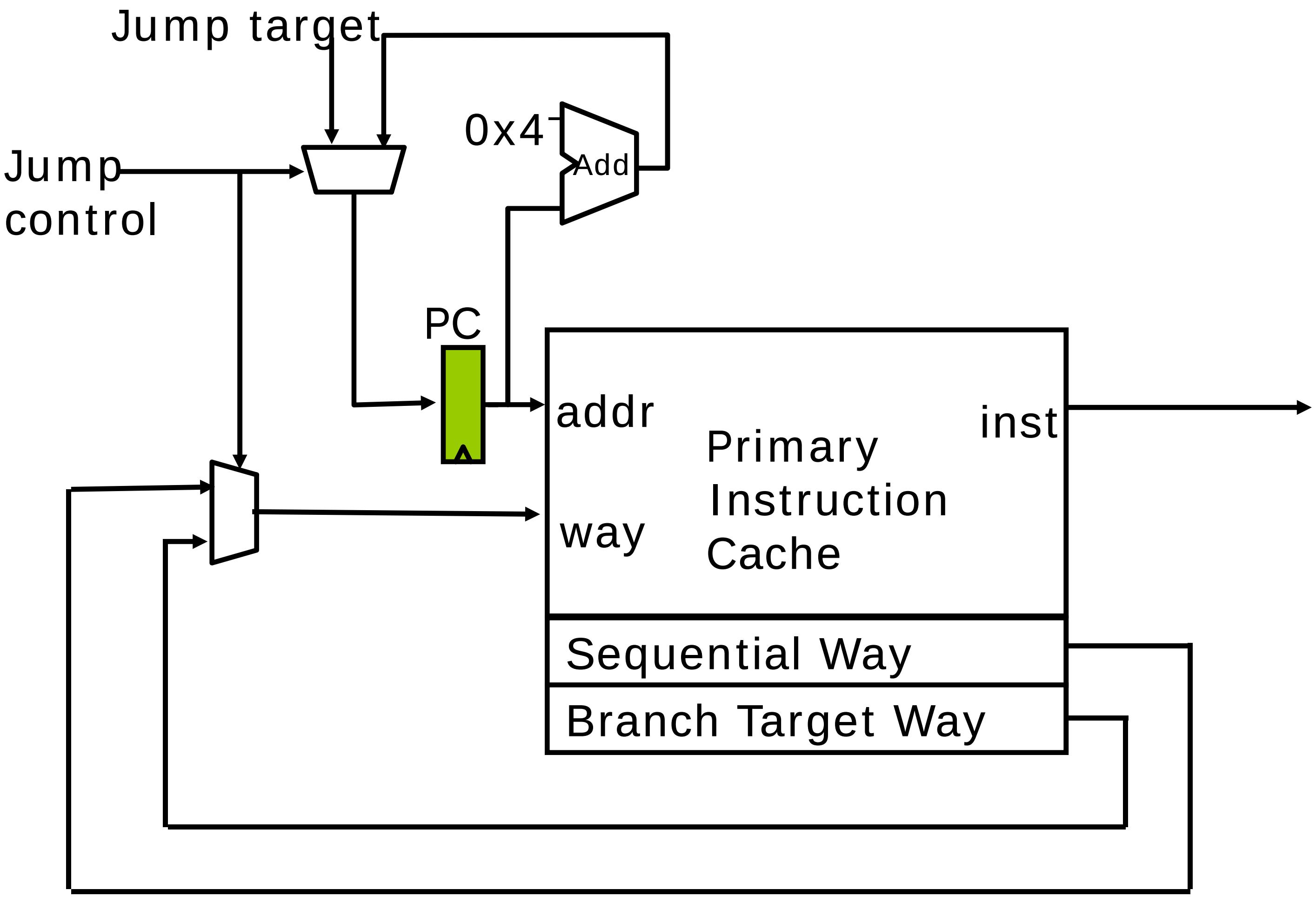
SLOW HIT
(change entry in prediction table)

MISS

Read <u>block</u> of data from next level of cache

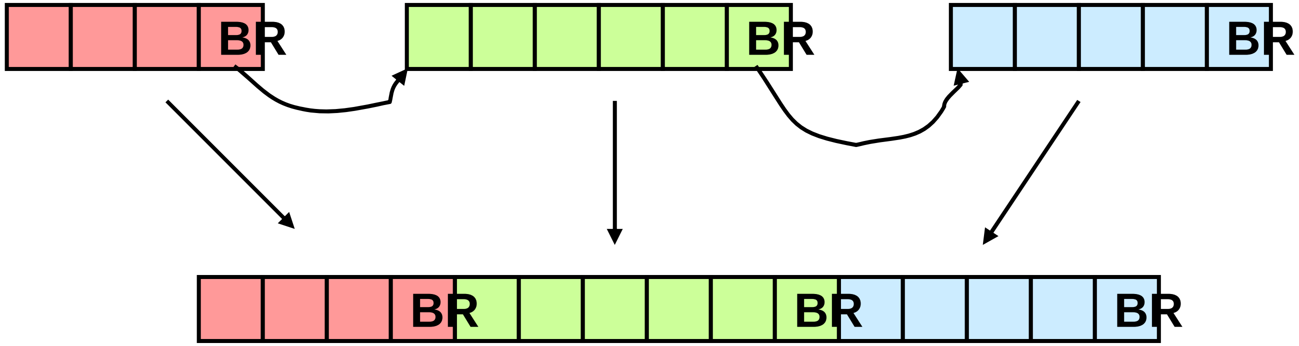
Way Predicting Instruction Cache

(Alpha 21264-like)



3. Fast (Instruction Cache) Hit times via Trace Cache

Key Idea: Pack multiple non-contiguous basic blocks into one contiguous trace cache line



- **Single fetch brings in multiple basic blocks**
- **Trace cache indexed by start address *and* next *n* branch predictions**

3. Fast Hit times via Trace Cache (Pentium 4 only; and last time?)

- Find more instruction level parallelism?
  How avoid translation from x86 to microops?
- Trace cache in Pentium 4

1. Dynamic traces of the executed instructions vs. static sequences of instructions as determined by layout in memory
   – Built-in branch predictor
2. Cache the micro-ops vs. x86 instructions
   – Decode/translate from x86 to micro-ops on trace cache miss

+ ⇒ better utilize long blocks (don't exit in middle of block, don't enter at label in middle of block)

- ⇒ complicated address mapping since addresses no longer aligned to power-of-2 multiples of word size

- ⇒ instructions may appear multiple times in multiple dynamic traces due to different branch outcomes

4: Increasing Cache Bandwidth by Pipelining

- Pipeline cache access to maintain bandwidth, but higher latency
- Instruction cache access pipeline stages:

  1: Pentium

  2: Pentium Pro through Pentium III

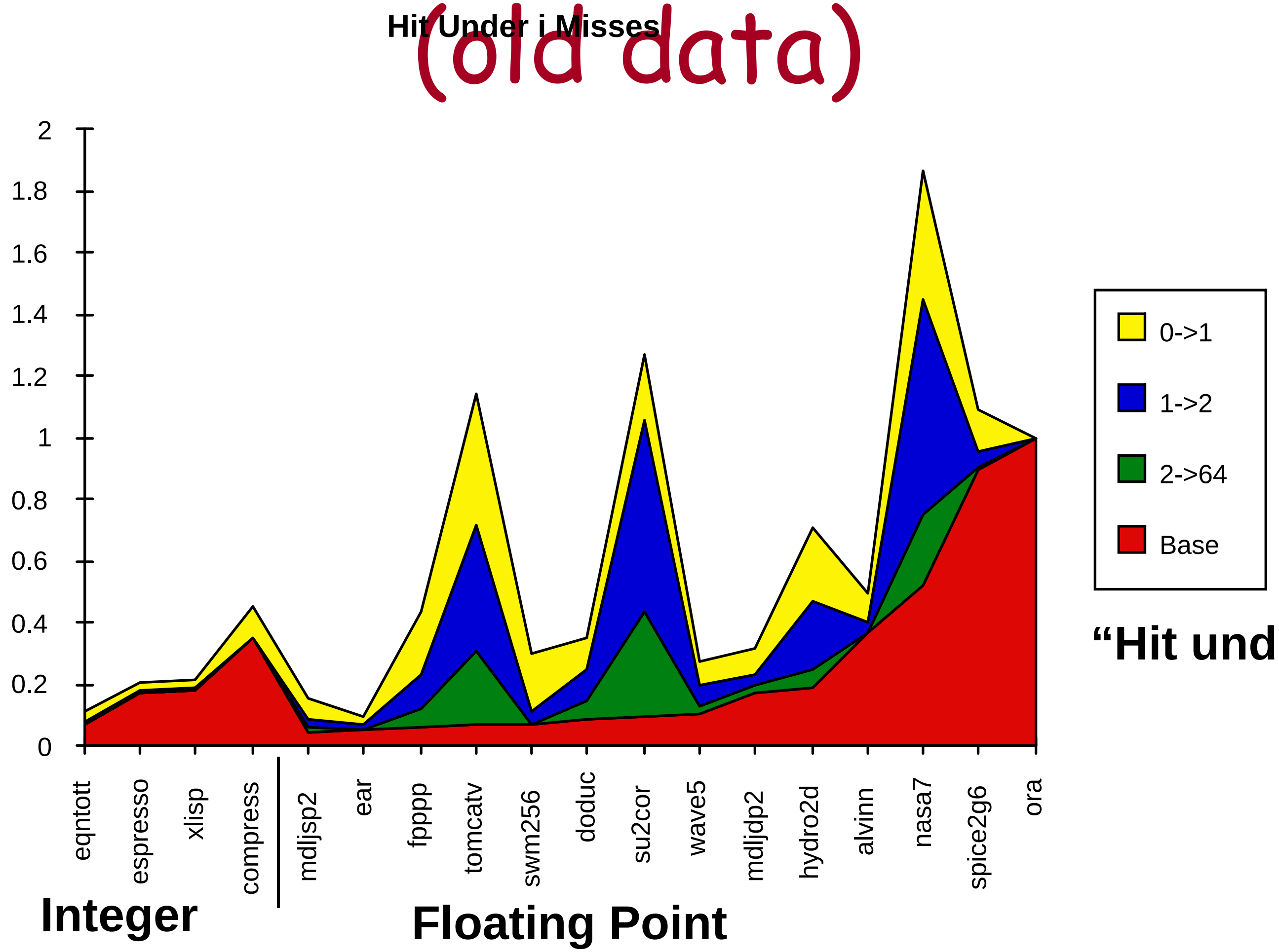
  4: Pentium 4

- ⇒ greater penalty on mispredicted branches
- ⇒ more clock cycles between the issue of the load and the use of the data

5. Increasing Cache Bandwidth: Non-Blocking Caches

- *Non-blocking cache* or *lockup-free cache* allow data cache to continue to supply cache hits during a miss
  - requires Full/Empty bits on registers or out-of-order execution
  - requires multi-bank or pipelined memories
- "*hit under miss*" reduces the effective miss penalty by working during miss vs. ignoring CPU requests
- "*hit under multiple miss*" or "*miss under miss*" may further lower the effective miss penalty by overlapping multiple misses
  - Significantly increases the complexity of the cache controller as there can be multiple outstanding memory accesses
  - Requires pipelined or banked memory system (otherwise cannot support)
  - Pentium Pro allows 4 outstanding memory misses
  - (Cray X1E vector supercomputer allows 2,048 outstanding memory misses)

Value of Hit Under Miss for SPEC (old data)



0->1
1->2
2->64
Base

“Hit under n Misses”

- **FP programs on average: AMAT= 0.68 -> 0.52 -> 0.34 -> 0.26**
- **Int programs on average: AMAT= 0.24 -> 0.20 -> 0.19 -> 0.19**
- **8 KB Data Cache, Direct Mapped, 32B block, 16 cycle miss, SPEC 92**

6: Increasing Cache Bandwidth via Multiple Banks

- **Rather than treat the cache as a single monolithic block, divide into independent banks that can support simultaneous accesses**
  - **E.g.,T1 ("Niagara") L2 has 4 banks**
- **Banking works best when accesses naturally spread themselves across banks ⇒ mapping of addresses to banks affects behavior of memory system**
- **Simple mapping that works well is "sequential interleaving"**
  - **Spread block addresses sequentially across banks**
  - **E,g, if there 4 banks, Bank 0 has all blocks whose address modulo 4 is 0; bank 1 has all blocks whose address modulo 4 is 1; …**

7. Reduce Miss Penalty: Early Restart and Critical Word First

- Don’t wait for full block before restarting CPU
- *Early restart*—As soon as the requested word of the block arrives, send it to the CPU and let the CPU continue execution
  - Spatial locality ⇒ tend to want next sequential word, so not clear size of benefit of just early restart
- *Critical Word First*—Request the missed word first from memory and send it to the CPU as soon as it arrives; let the CPU continue execution while filling the rest of the words in the block
  - Long blocks more popular today ⇒ Critical Word 1st Widely used

block

8. Merging Write Buffer to Reduce Miss Penalty

- **Write buffer to allow processor to continue while waiting to write to memory**
- **If buffer contains modified blocks, the addresses can be checked to see if address of new data matches the address of a valid write buffer entry**
- **If so, new data are combined with that entry**
- **Increases block size of write for write-through cache of writes to sequential words, bytes since multiword writes more efficient to memory**
- **The Sun T1 (Niagara) processor, among many others, uses write merging**

9. Reducing Misses by Compiler Optimizations

- **McFarling [1989] reduced caches misses by 75% on 8KB direct mapped cache, 4 byte blocks in software**
- **Instructions**
  - **Reorder procedures in memory so as to reduce conflict misses**
  - **Profiling to look at conflicts (using tools they developed)**
- **Data**
  - ***Merging Arrays*: improve spatial locality by single array of compound elements vs. 2 arrays**
  - ***Loop Interchange*: change nesting of loops to access data in order stored in memory**
  - ***Loop Fusion*: Combine 2 independent loops that have same looping and some variables overlap**
  - ***Blocking*: Improve temporal locality by accessing “blocks” of data repeatedly vs. going down whole columns or rows**

Merging Arrays Example

```
/* Before: 2 sequential arrays */
int val[SIZE];
int key[SIZE];

/* After: 1 array of stuctures */
struct merge {
   int val;
   int key;
};
struct merge merged_array[SIZE];
```

Reducing conflicts between val & key;
improve spatial locality

Loop Interchange Example

```
/* Before */
for (k = 0; k < 100; k = k+1)
   for (j = 0; j < 100; j = j+1)
        for (i = 0; i < 5000; i = i+1)
                x[i][j] = 2 * x[i][j];
/* After */
for (k = 0; k < 100; k = k+1)
   for (i = 0; i < 5000; i = i+1)
        for (j = 0; j < 100; j = j+1)
                x[i][j] = 2 * x[i][j];
```

Sequential accesses instead of striding through memory every 100 words; improved spatial locality

Loop Fusion Example

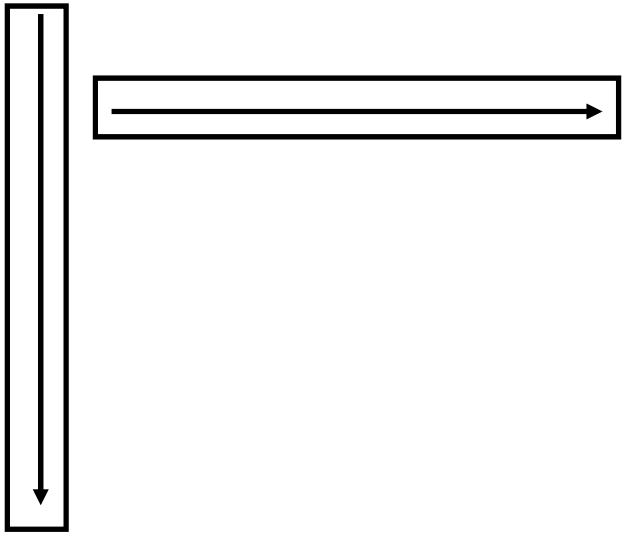
```
/* Before */
for (i = 0; i < N; i = i+1)
   for (j = 0; j < N; j = j+1)
        a[i][j] = 1/b[i][j] * c[i][j];
for (i = 0; i < N; i = i+1)
   for (j = 0; j < N; j = j+1)
        d[i][j] = a[i][j] + c[i][j];

/* After */
for (i = 0; i < N; i = i+1)
   for (j = 0; j < N; j = j+1)
   {    a[i][j] = 1/b[i][j] * c[i][j];
        d[i][j] = a[i][j] + c[i][j];}
```

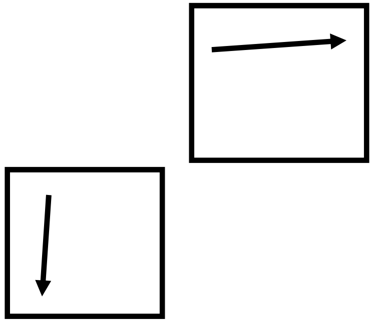
2 misses per access to a & c vs. one miss per access; improve spatial locality

Blocking Example

```
/* Before */
for (i = 0; i < N; i = i+1)
   for (j = 0; j < N; j = j+1)
      {r = 0;
       for (k = 0; k < N; k = k+1){
          r = r + y[i][k]*z[k][j];};
       x[i][j] = r;
      };
```



- **Two Inner Loops:**
  - **Read all NxN elements of z[]**
  - **Read N elements of 1 row of y[] repeatedly**
  - **Write N elements of 1 row of x[]**
- **Capacity Misses a function of N & Cache Size:**
  - **$2N^3 + N^2$ => (assuming no conflict; otherwise …)**
- **Idea: compute on BxB submatrix that fits in cache**

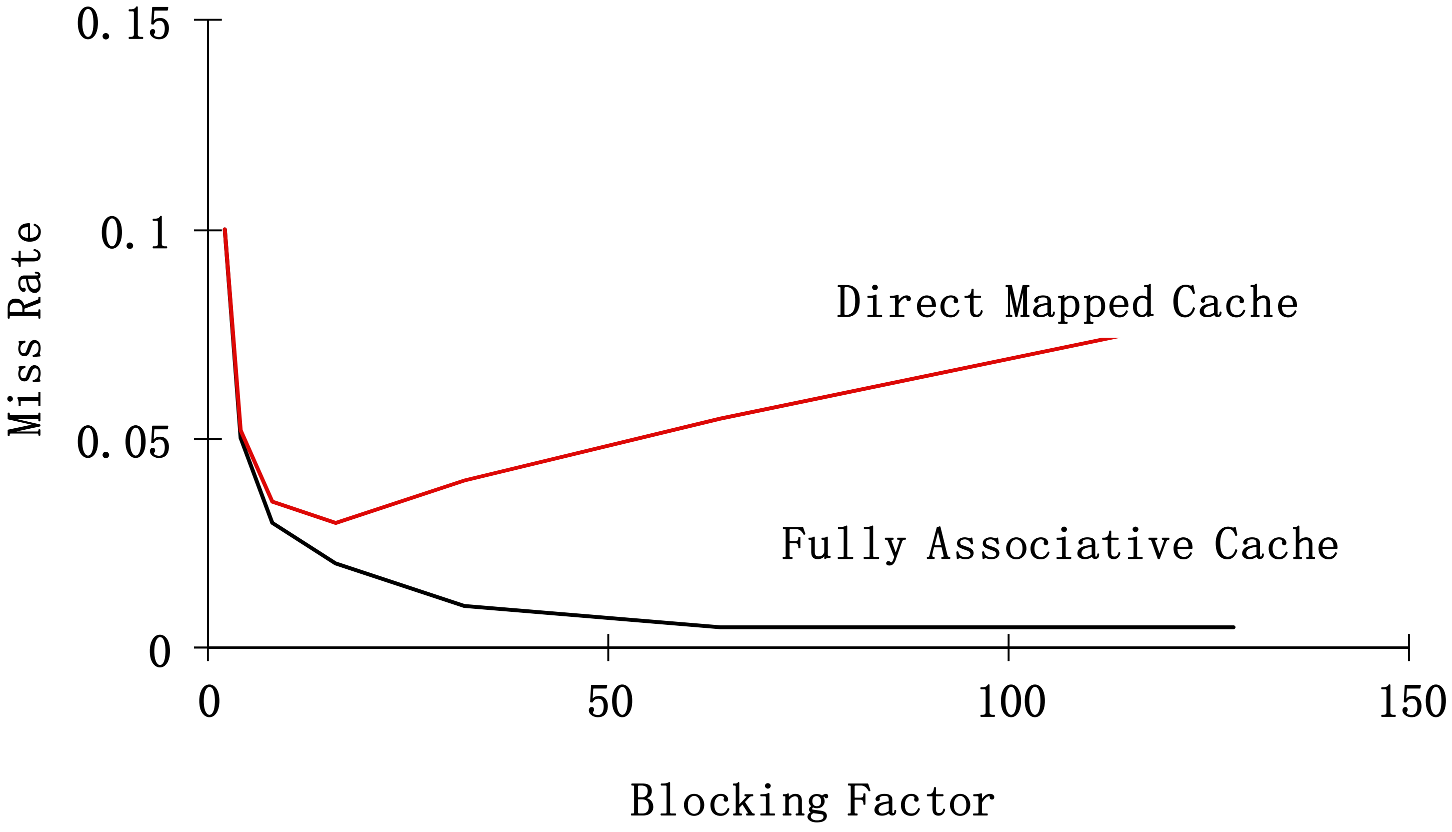


Blocking Example

```
/* After */
for (jj = 0; jj < N; jj = jj+B)
for (kk = 0; kk < N; kk = kk+B)
for (i = 0; i < N; i = i+1)
    for (j = jj; j < min(jj+B-1,N); j = j+1)
     {r = 0;
      for (k = kk; k < min(kk+B-1,N); k = k+1) {
        r = r + y[i][k]*z[k][j];};
      x[i][j] = x[i][j] + r;
     };
```

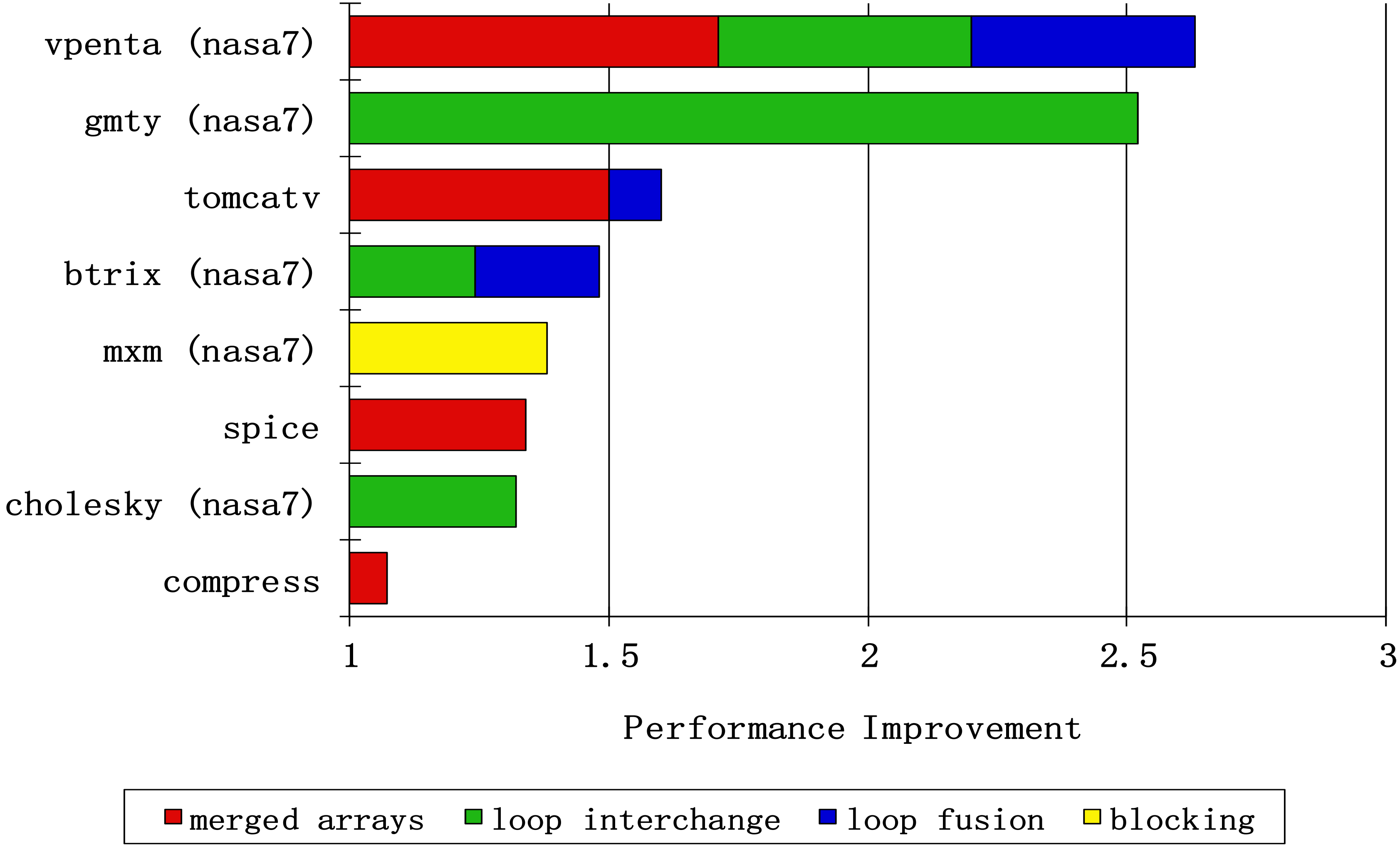
- **B called *Blocking Factor***
- **Capacity Misses from $2N^3 + N^2$ to $2N^3/B + N^2$**
- **Conflict Misses Too?**

Reducing Conflict Misses by Blocking



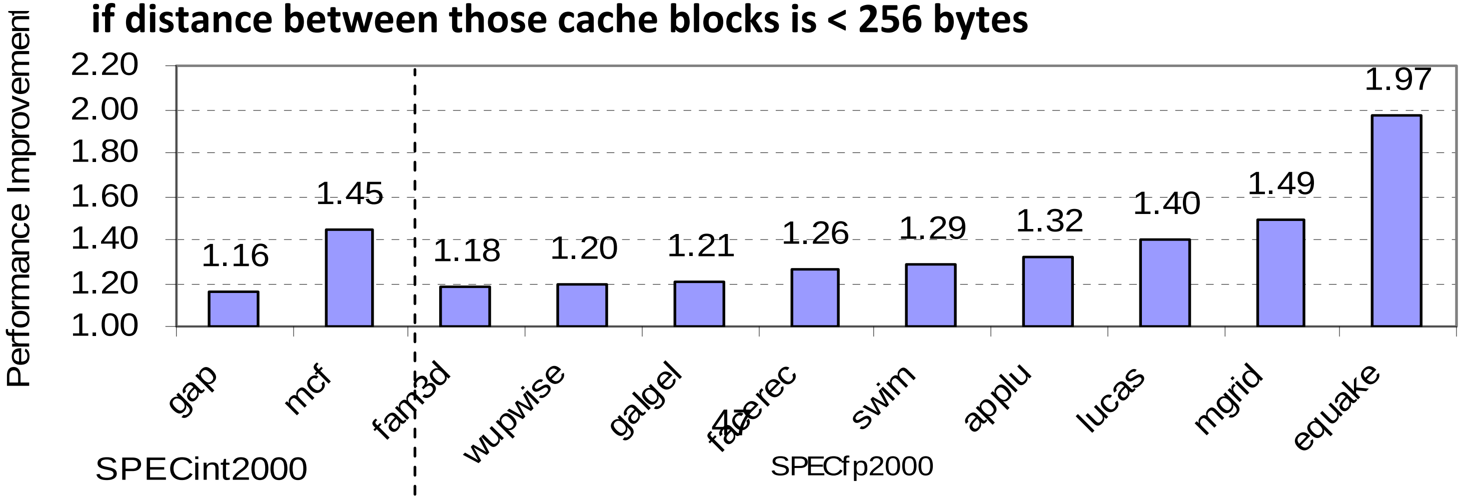
- **Conflict misses in caches not FA vs. Blocking size**
  - **Lam et al [1991] a blocking factor of 24 had a fifth the misses vs. 48 despite both fit in cache**

Summary of Compiler Optimizations to Reduce Cache Misses (by hand)



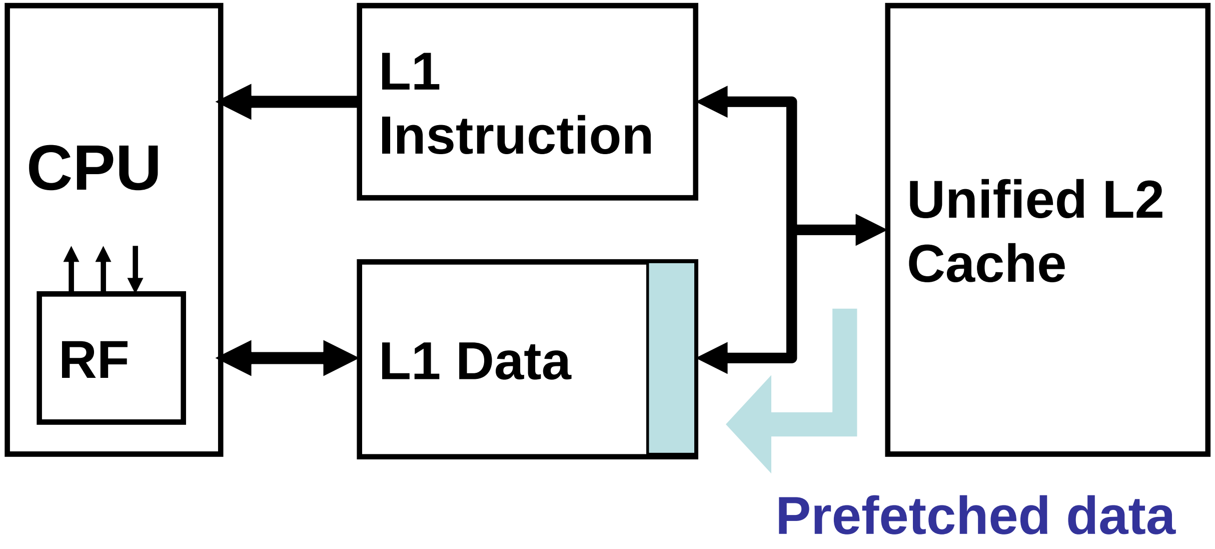
10. Reducing Misses by Hardware Prefetching of Instructions & Data

- Prefetching relies on having extra memory bandwidth that can be used without penalty
- Instruction Prefetching
  - Typically, CPU fetches 2 blocks on a miss: the requested block and the next consecutive block.
  - Requested block is placed in instruction cache when it returns, and prefetched block is placed into instruction stream buffer
- Data Prefetching
  - Pentium 4 can prefetch data into L2 cache from up to 8 streams from 8 different 4 KB pages
  - Prefetching invoked if 2 successive L2 cache misses to a page, if distance between those cache blocks is < 256 bytes



Issues in Prefetching

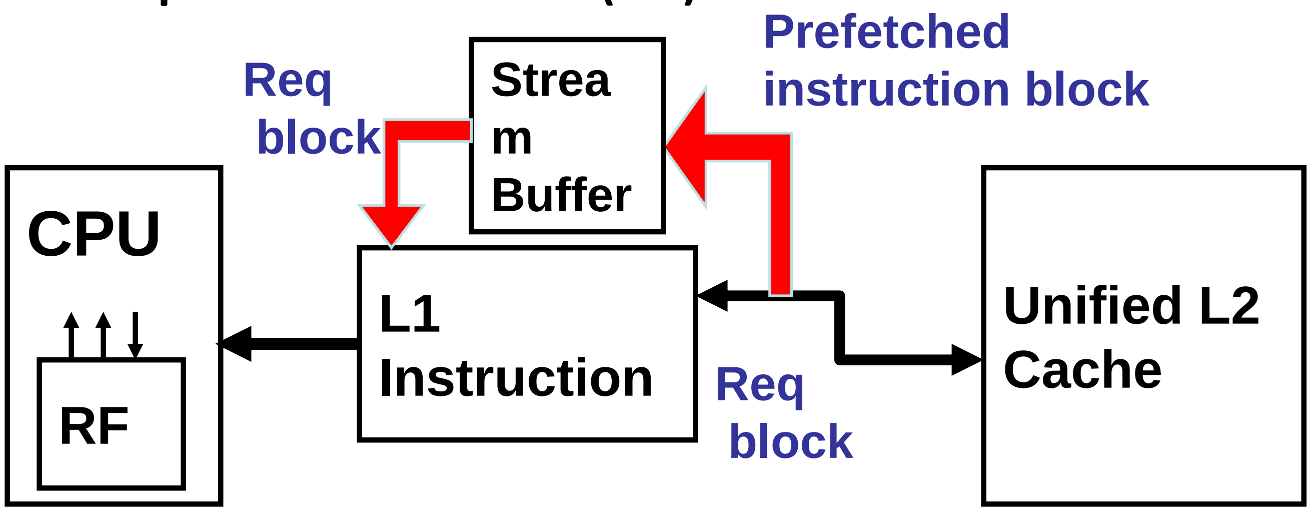
- Usefulness – should produce hits
- Timeliness – not late and not too early
- Cache and bandwidth pollution



Hardware Instruction Prefetching

Instruction prefetch in Alpha AXP 21064

- Fetch two blocks on a miss; the requested block (i) and the next consecutive block (i+1)
- Requested block placed in cache, and next block in instruction stream buffer
- If miss in cache but hit in stream buffer, move stream buffer block into cache and prefetch next block (i+2)



Hardware Data Prefetching

- **Prefetch-on-miss:**
  - Prefetch b + 1 upon miss on b
- **One Block Lookahead (OBL) scheme**
  - Initiate prefetch for block b + 1 when block b is accessed
  - *Why is this different from doubling block size?*
  - Can extend to N block lookahead
- **Strided prefetch**
  - If observe sequence of accesses to block b, b+N, b+2N, then prefetch b+3N etc.

Example: IBM Power 5 [2003] supports eight independent streams of strided prefetch per processor, prefetching 12 lines ahead of current access

11. Reducing Misses by Software Prefetching Data

- Data Prefetch
  - Load data into register (HP PA-RISC loads)
  - Cache Prefetch: load into cache
    (MIPS IV, PowerPC, SPARC v. 9)
  - Special prefetching instructions cannot cause faults;
    a form of speculative execution
- Issuing Prefetch Instructions takes time
  - Is cost of prefetch issues < savings in reduced misses?
  - Wider superscalar reduces difficulty of issue bandwidth

Technique	Hit Time	Bandwidth	Miss penalty	Miss rate	HW cost/ complexity	Comment
Small and simple caches	+			–	0	Trivial; widely used
Way-predicting caches	+				1	Used in Pentium 4
Trace caches	+				3	Used in Pentium 4
Pipelined cache access	–	+			1	Widely used
Nonblocking caches		+	+		3	Widely used
Banked caches		+			1	Used in L2 of Opteron and Niagara
Critical word first and early restart			+		2	Widely used
Merging write buffer			+		1	Widely used with write through
Compiler techniques to reduce cache misses				+	0	Software is a challenge; some computers have compiler option
Hardware prefetching of instructions and data			+	+	2 instr., 3 data	Many prefetch instructions; AMD Opteron prefetches data
Compiler-controlled prefetching			+	+	3	Needs nonblocking cache; in many CPUs

Next Time

- **Address translation and protection for virtual memory systems**