CS250
VLSI Systems Design

Lecture 7: Memory Technology and Patterns

Spring 2016

John Wawrzynek
with
Chris Yarp (GSI)

Thanks to John Lazzaro for the slides

Memory: Technology and Patterns

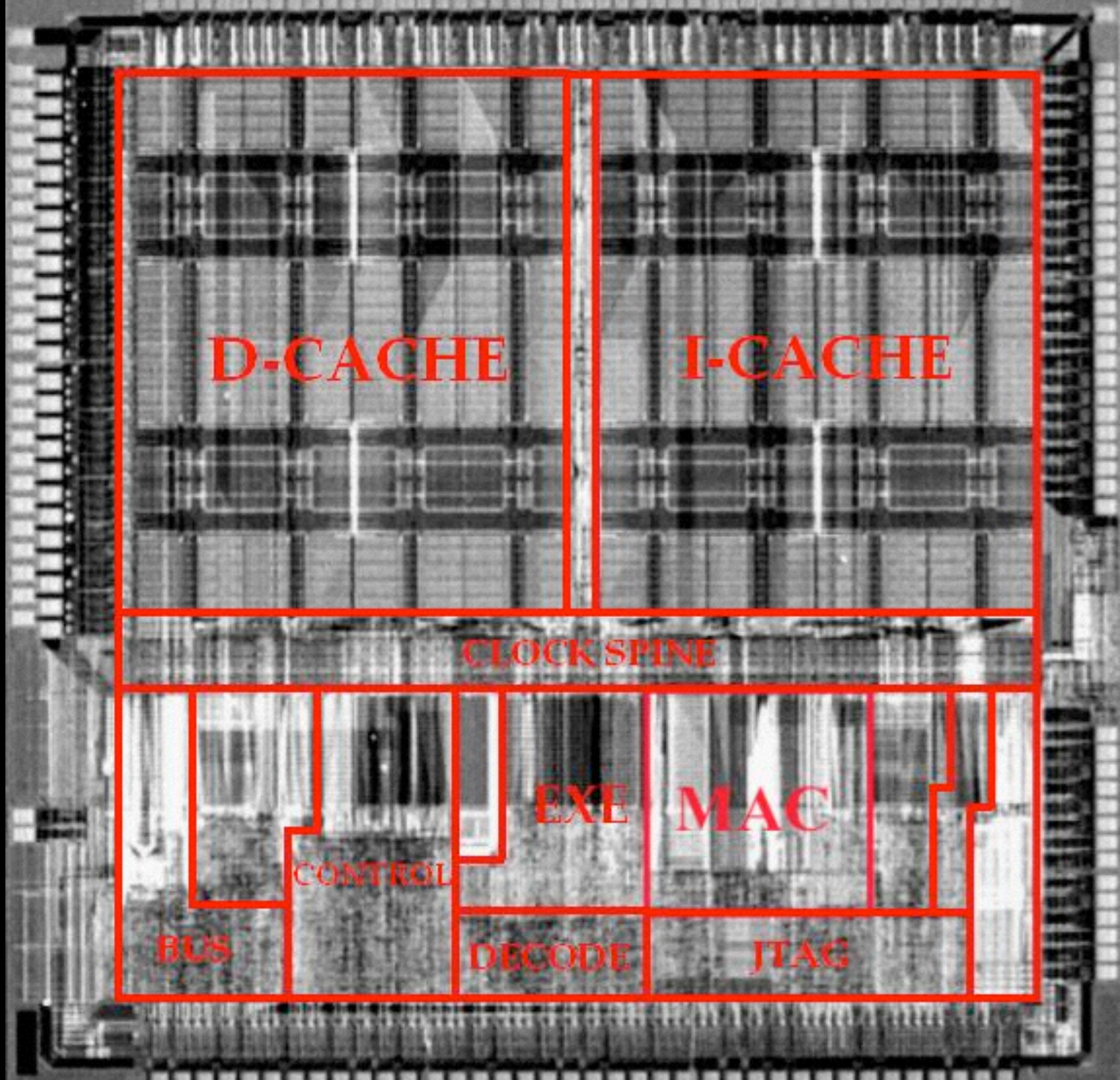
- **Memory, the 10,000 ft view.** Latency, from Steve Wozniak to the power wall.
- **How DRAM works.** Memory design when low cost per bit is the priority.

Break

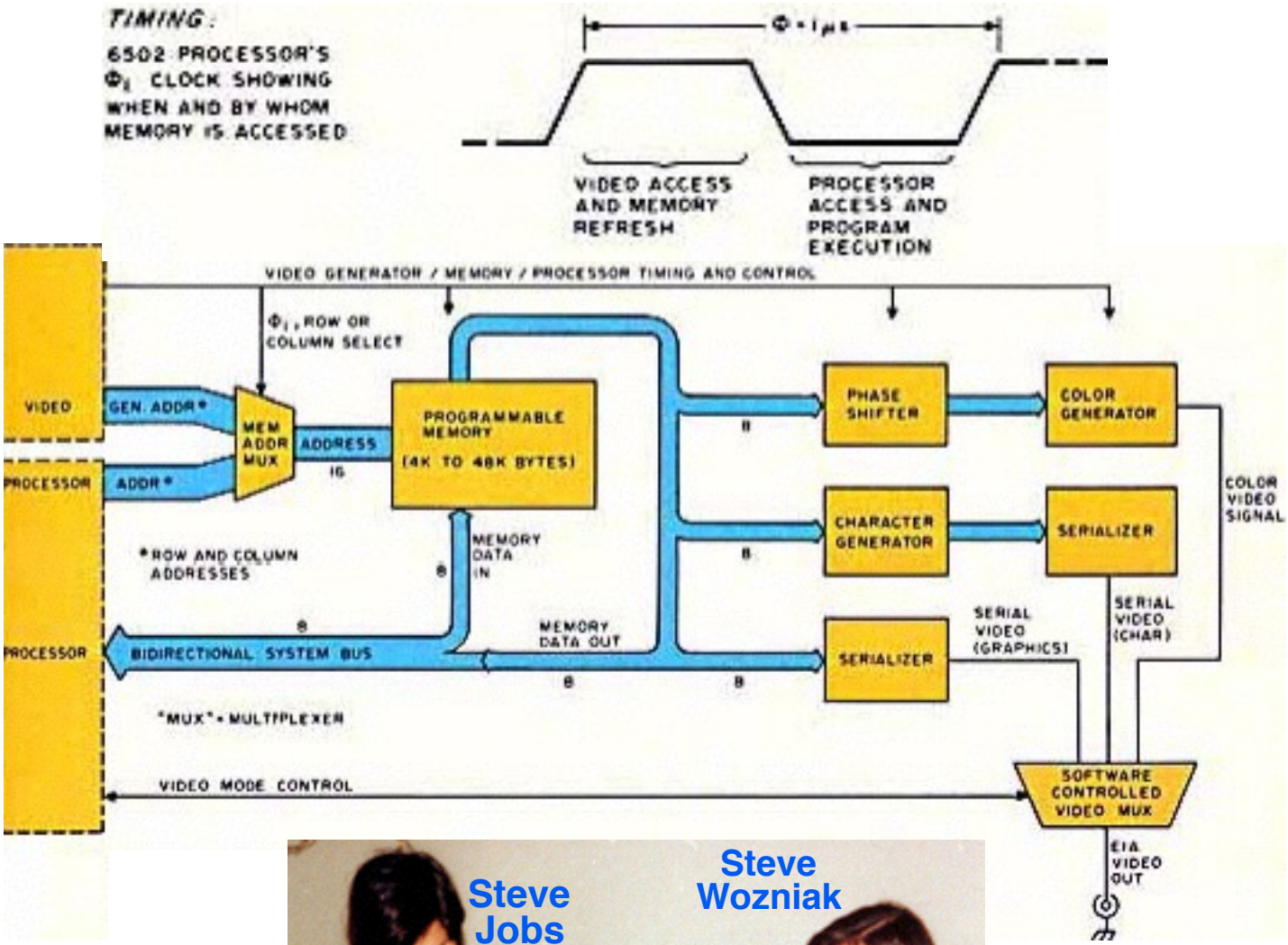
- **How SRAM works.** The memory technology available on logic dies.
- **Memory design patterns.** Ways to use SRAM in your project designs.

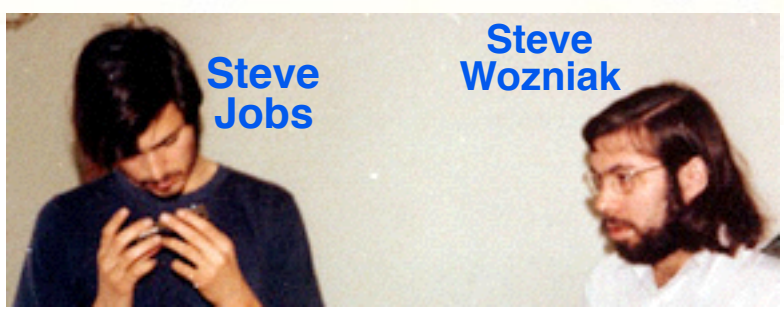
40% of this ARM CPU is devoted to SRAM cache.

But the role of cache in computer design has varied widely over time.



1977: DRAM faster than microprocessors



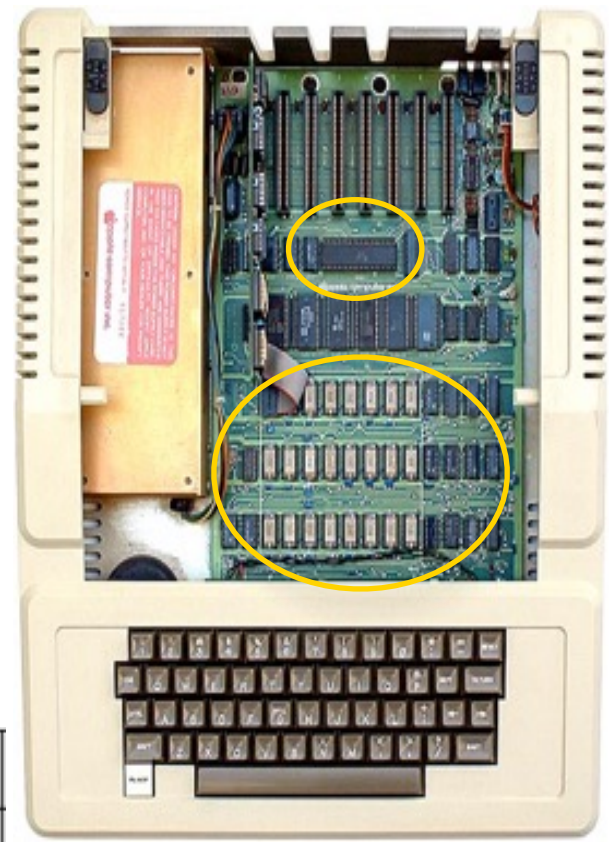


RAM Complement	Apple II System
4K	$ 1,298.00
48K	2,638.00

Apple][(1977)

CPU: 1000 ns

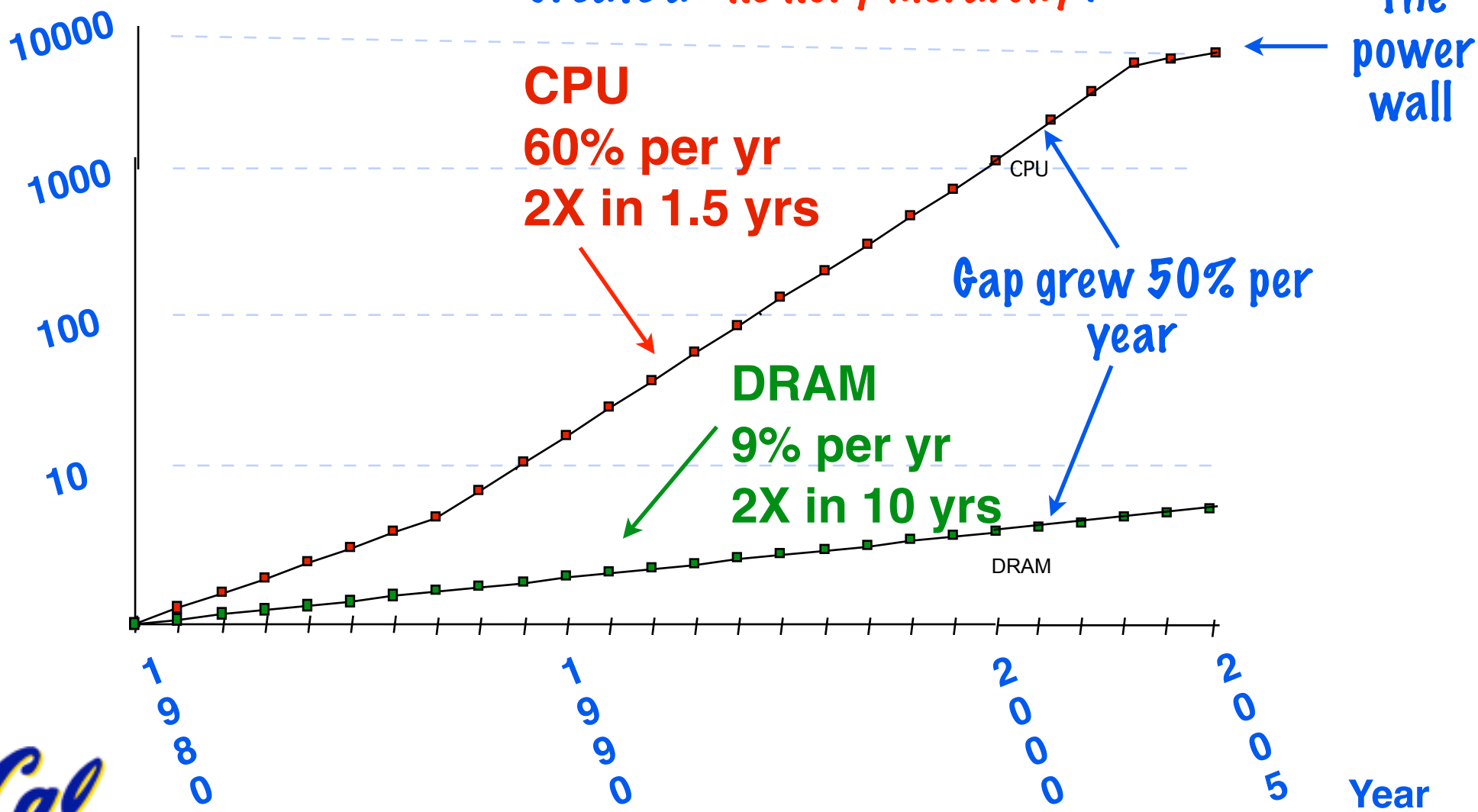
DRAM: 400 ns



1980-2003, CPU speed outpaced DRAM ...

Q. How do architects address this gap?

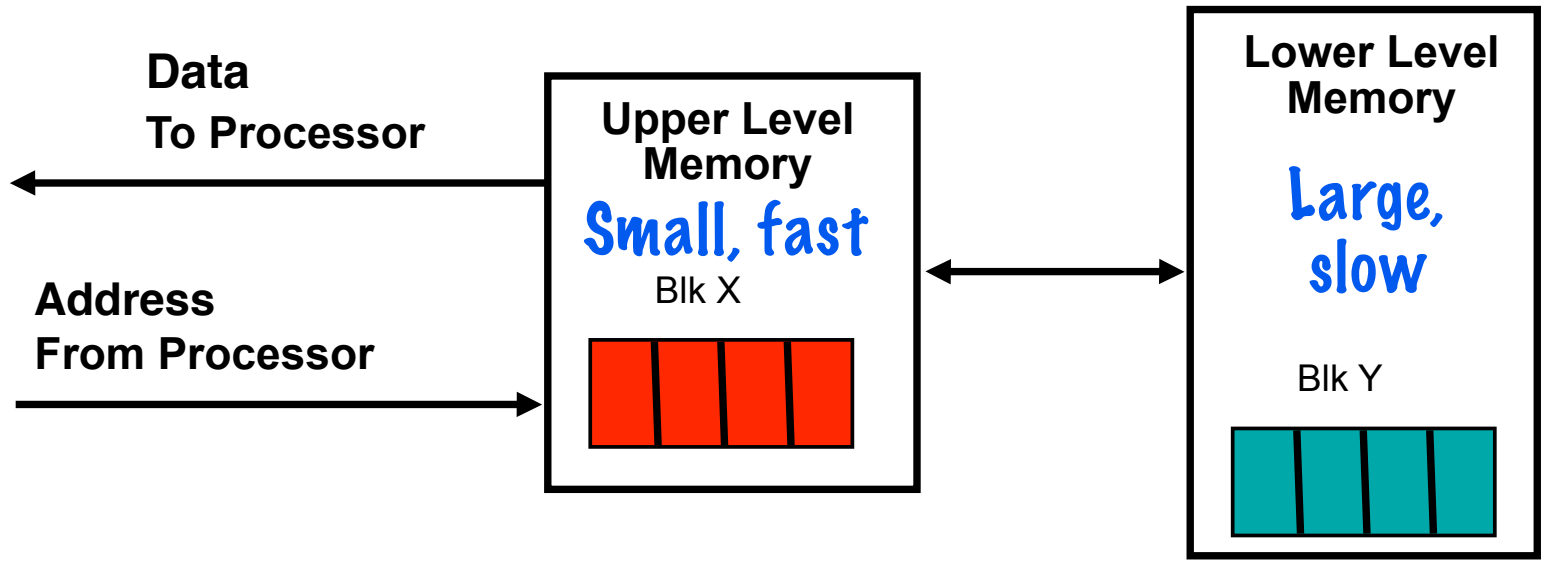
A. Put smaller, faster "cache" memories between CPU and DRAM. Create a "memory hierarchy".

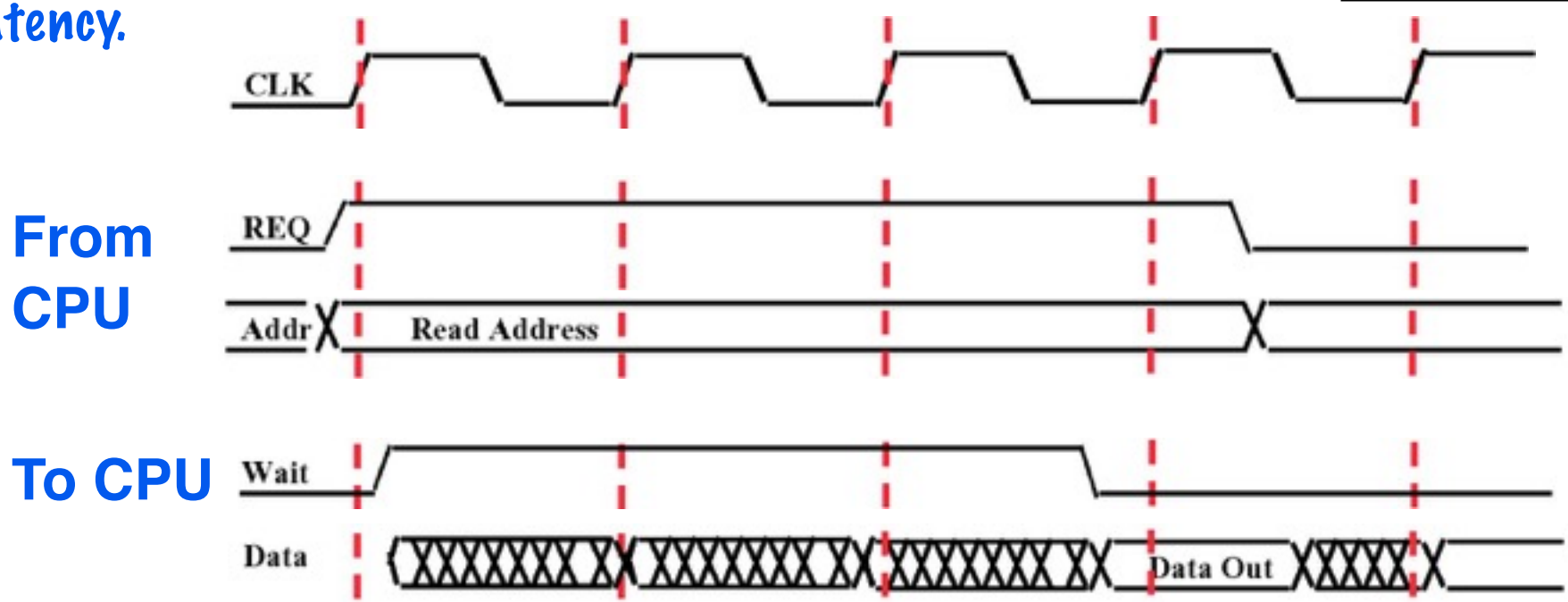


Caches: Variable-latency memory ports

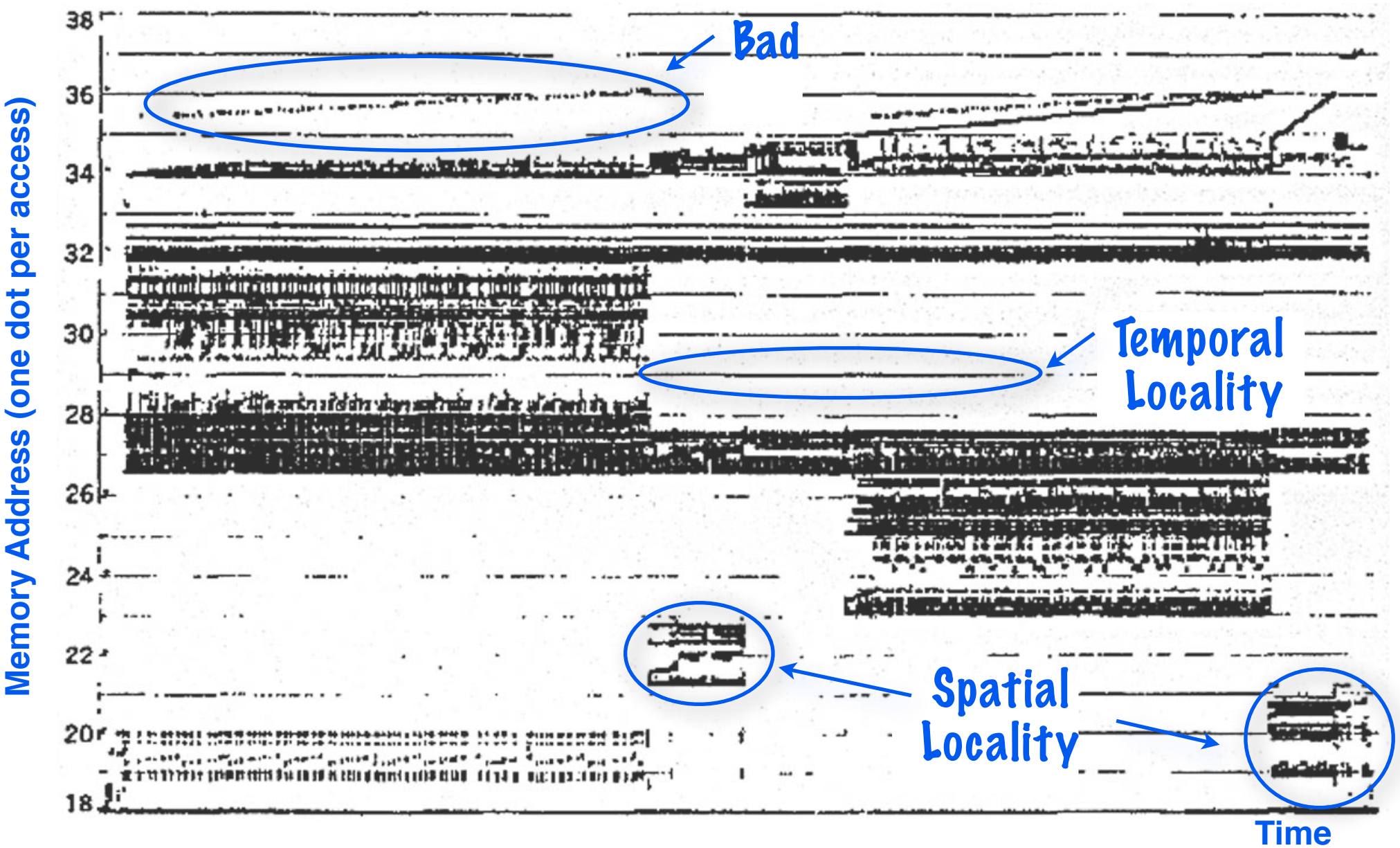
Data in upper memory returned with lower latency.

Data in lower level returned with higher latency.





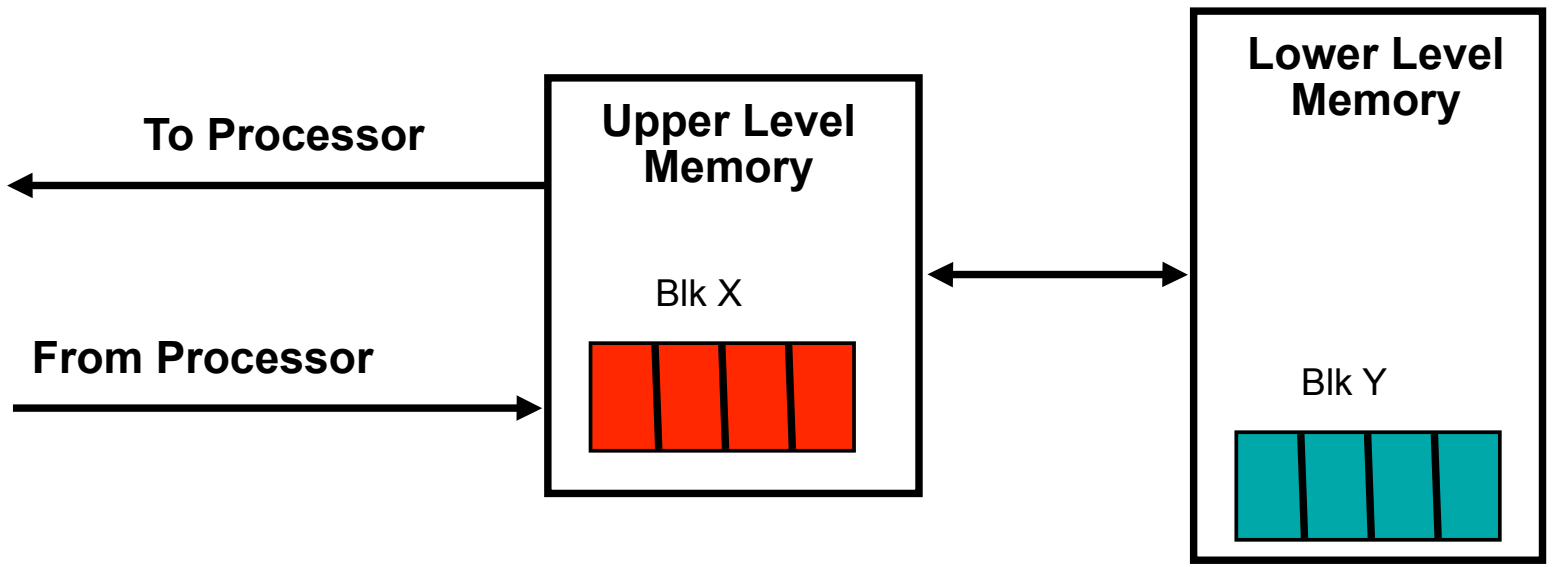
Programs with locality cache well ...



Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)

The caching algorithm in one slide

- **Temporal locality: Keep most recently accessed data closer to processor.**



- **Spatial locality: Move contiguous blocks in the address space to upper levels.**

2005 Memory Hierarchy: Apple iMac G5

Managed by compiler

Managed by hardware

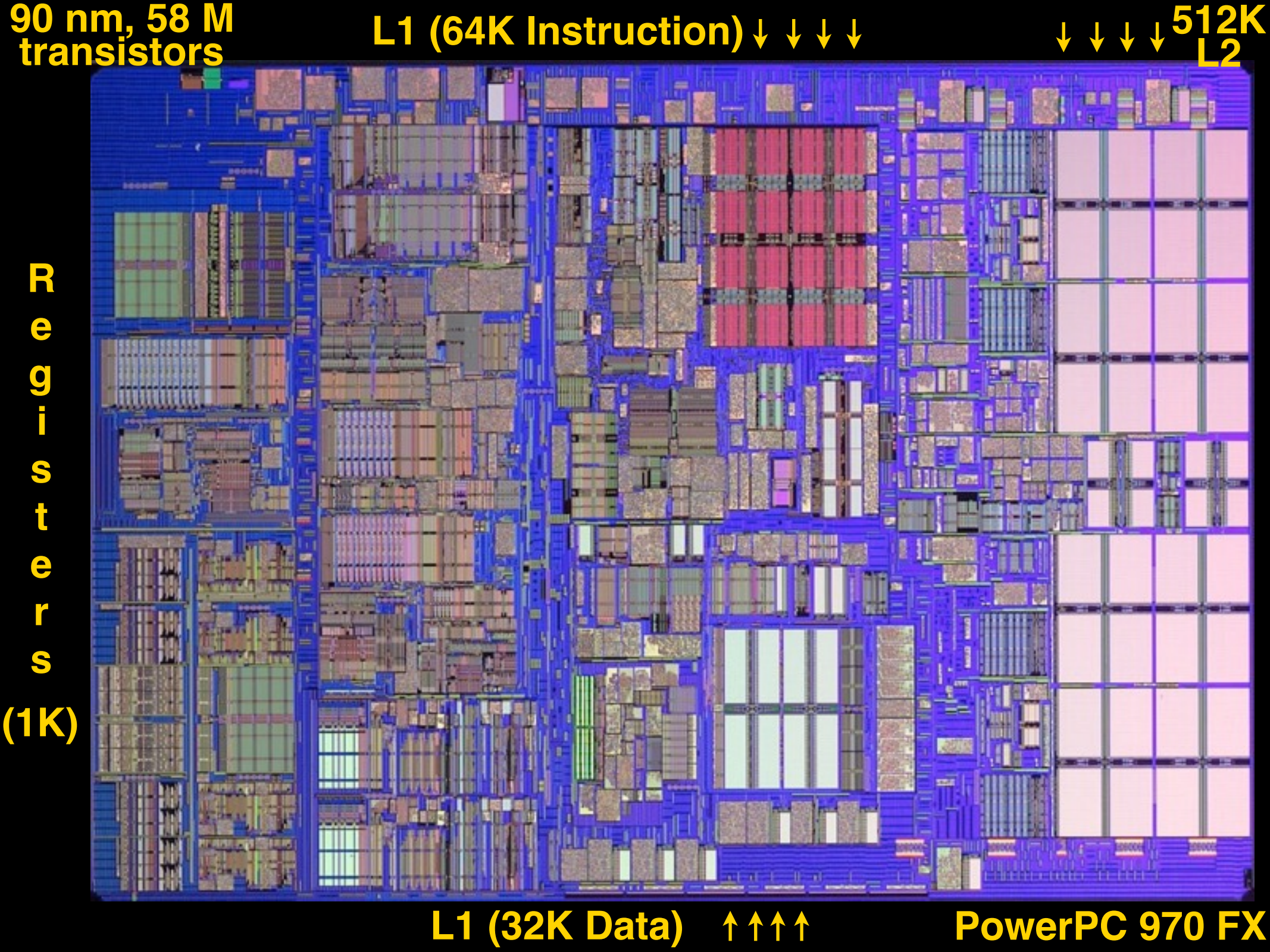
Managed by OS, hardware, application

	Reg	L1 Inst	L1 Data	L2	DRAM	Disk
Size	1K	64K	32K	512K	256M	80G
Latency (cycles)	1	3	3	11	160	1E+07

iMac G5
1.6 GHz
$1299.00

Goal: Illusion of large, fast, cheap memory

Let programs address a memory space that scales to the disk size, at a speed that is usually as fast as register access


90 nm, 58 M transistors
L1 (64K Instruction)
512K L2
Registers (1K)
L1 (32K Data)
PowerPC 970 FX

Latency: A closer look

Read latency: Time to return first byte of a random access

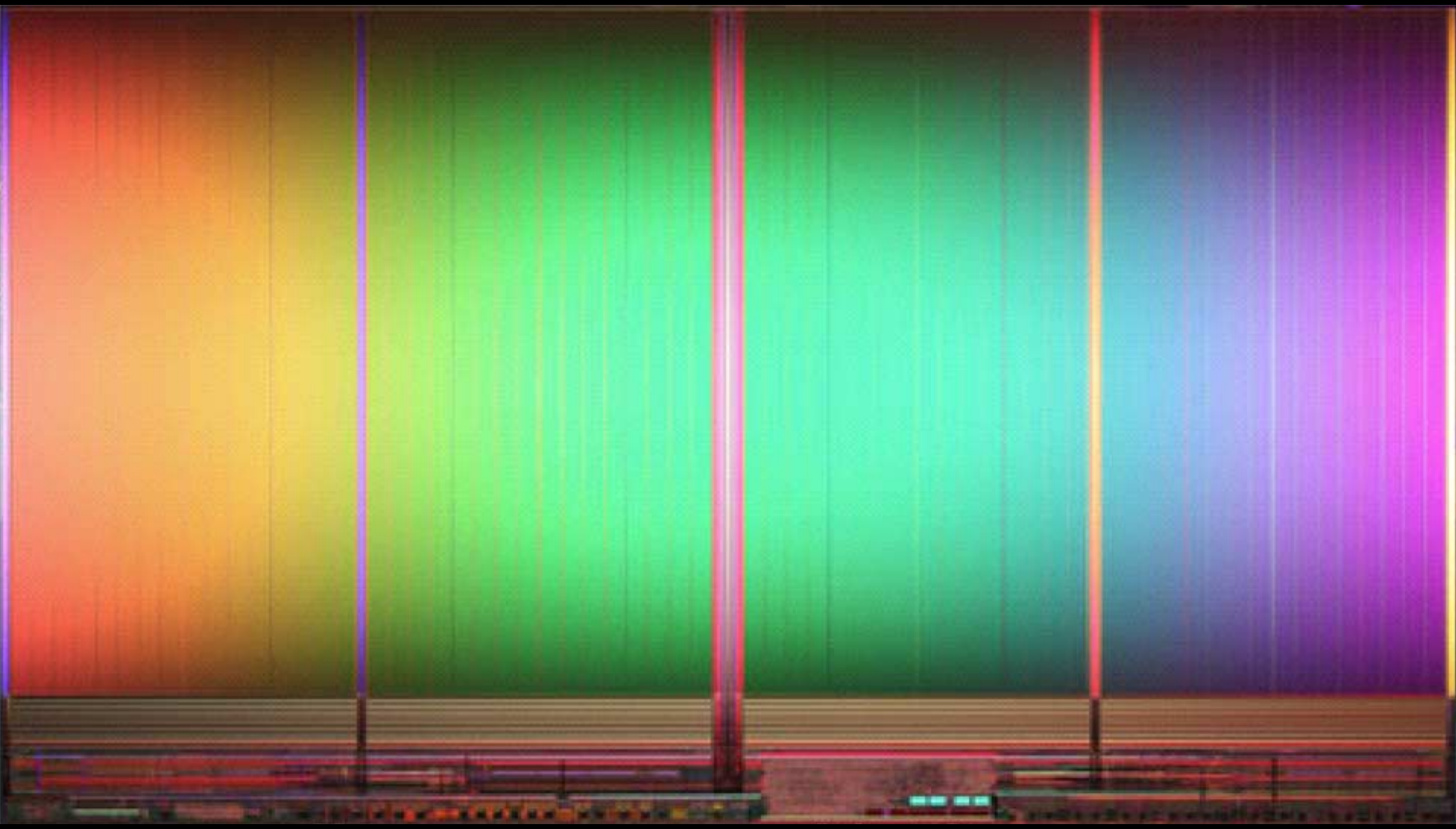
	Reg	L1 Inst	L1 Data	L2	DRAM	Disk
Size	1K	64K	32K	512K	256M	80G
Latency (cycles)	1	3	3	11	160	1E+07
Latency (sec)	0.6n	1.9n	1.9n	6.9n	100n	12.5m
Hz	1.6G	533M	533M	145M	10M	80

Architect's latency toolkit:

(1) Parallelism. Request data from N 1-bit-wide memories at the same time. Overlaps latency cost for all N bits. Provides N times the bandwidth. Requests to N memory banks (interleaving) have potential of N times the bandwidth.

(2) Pipeline memory. If memory has N cycles of latency, issue a request each cycle, receive it N cycles later.

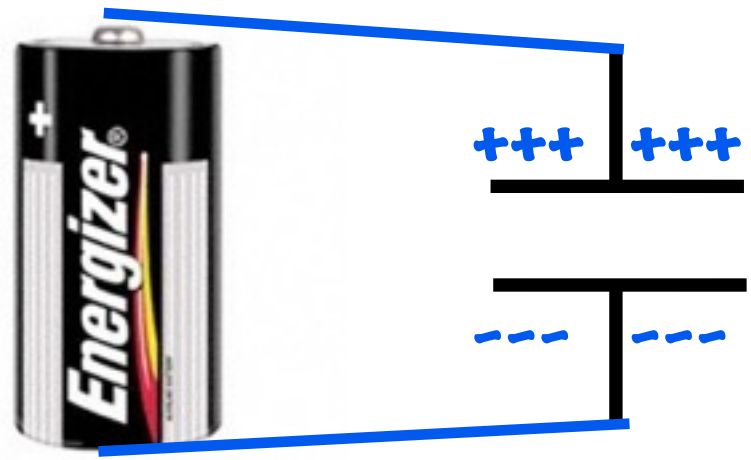
Capacitance and memory

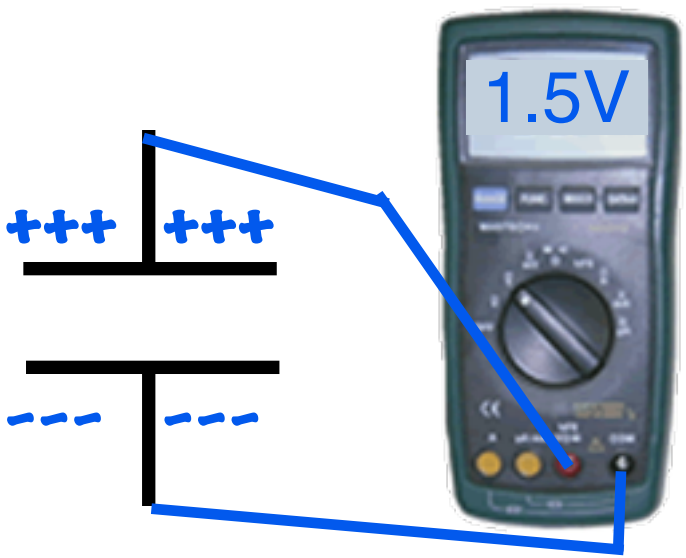


Intel Micron 8 GB NAND flash device, 2 bit per cell, 25 nm minimum feature, 16.5 mm by 10.1 mm.

Storing computational state as charge

State is coded as the amount of energy stored by a device.





State is read by sensing the amount of energy

Problems: noise changes Q (up or down), parasitics leak or source Q. Fortunately, Q cannot change instantaneously, but that only gets us in the ballpark.

How do we fight noise and win?

Store more energy than we expect from the noise.

Q = CV. To store more charge, use a bigger V or make a bigger C.
Cost: Power, chip size.

Represent state as charge in ways that are robust to noise.

Example: 1 bit per capacitor.
Write 1.5 volts on C.
To read C, measure V.
V > 0.75 volts is a "1".
V < 0.75 volts is a "0".

Cost: Could have stored many bits on that capacitor.

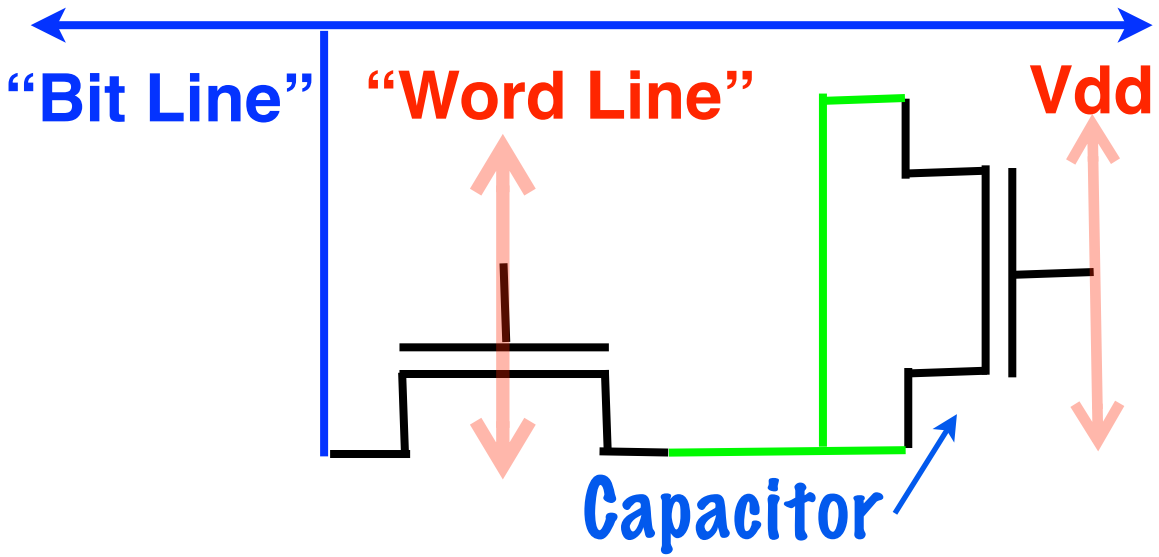
Correct small state errors that are introduced by noise.

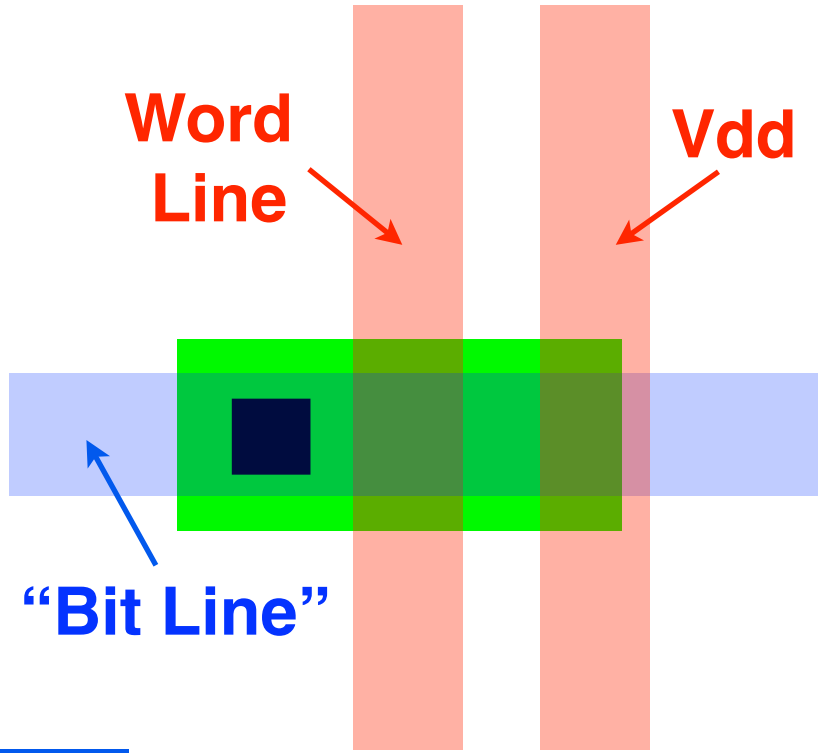
Ex: read C every 1 ms
Is V > 0.75 volts?
Write back 1.5V (yes) or 0V (no).

Cost: Complexity.

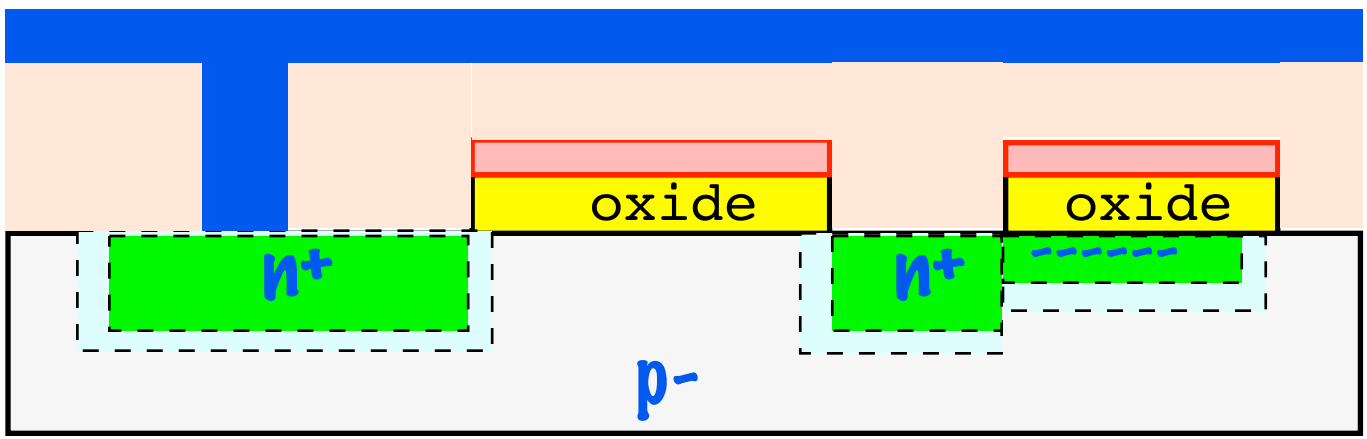
Dynamic Memory Cells

DRAM cell: 1 transistor, 1 capacitor



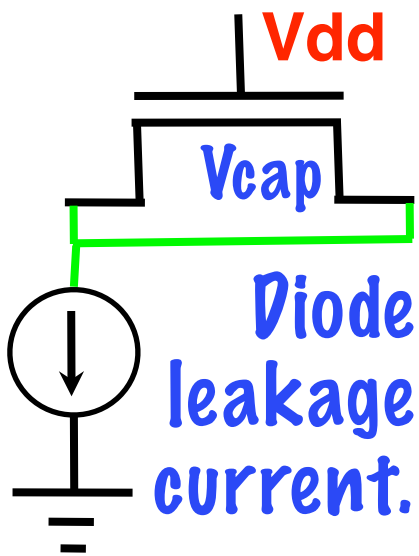


"Bit Line"



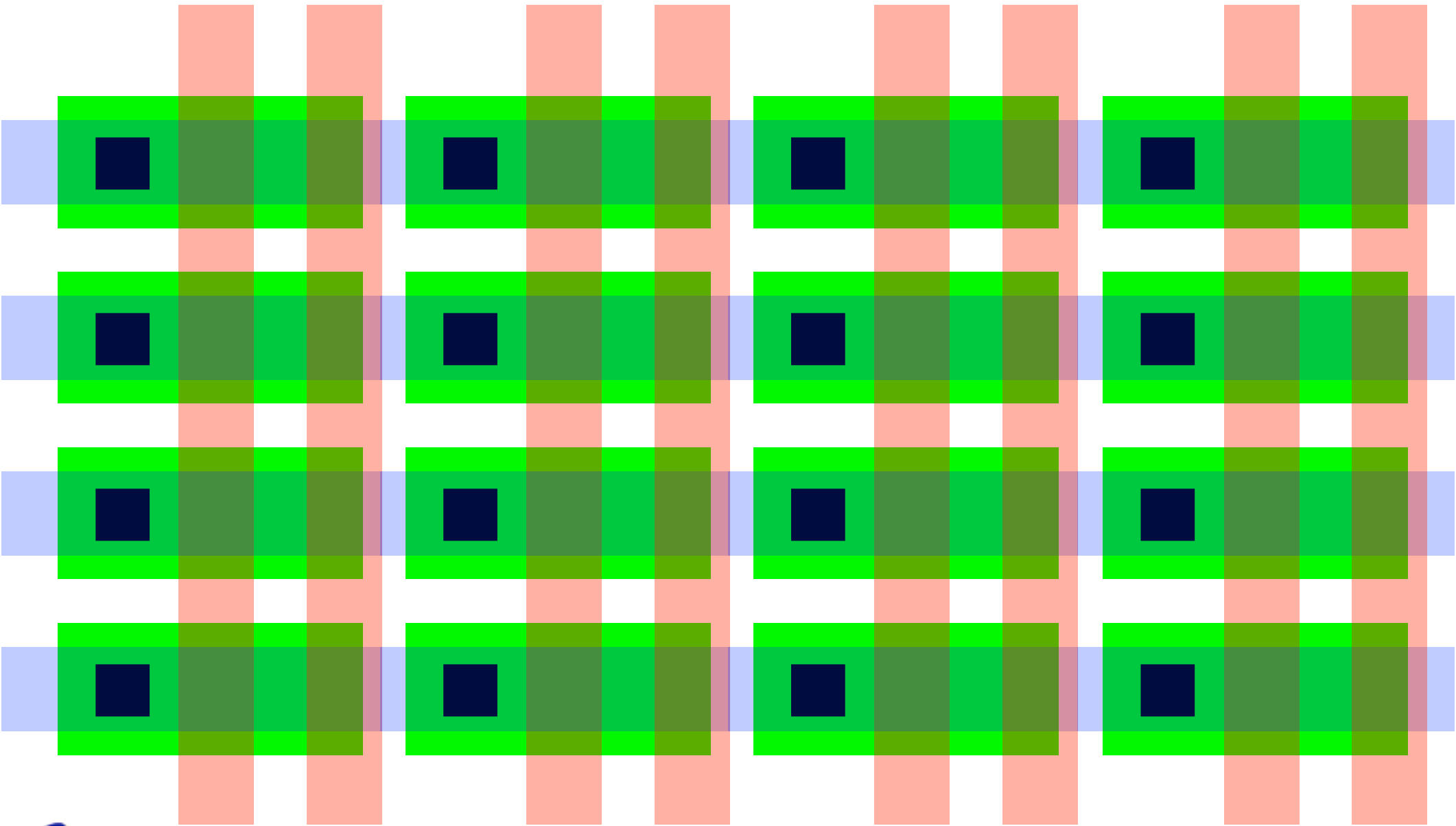
Word Line and Vdd run on "z-axis"

Why Vcap values start out at ground.



Diode leakage current.

A 4 x 4 DRAM array (16 bits) ....



Invented after SRAM, by Robert Dennard

United States Patent Office

3,387,286
Patented June 4, 1968

3,387,286

FIELD-EFFECT TRANSISTOR MEMORY

Robert H. Dennard, Croton-on-Hudson, N.Y., assignor to International Business Machines Corporation, Armonk, N.Y., a corporation of New York

Filed July 14, 1967, Ser. No. 653,415
21 Claims. (Cl. 340—173)

tinent in disclosing various concepts and structures which have been developed in the application of field-effect transistors to different types of memory applications, the primary thrust up to this time in conventional read-write random access memories has been to connect a plurality of field-effect transistors in each cell in a latch configuration. Memories of this type require a large number of active devices in each cell and therefore each cell re-

FIG. 1

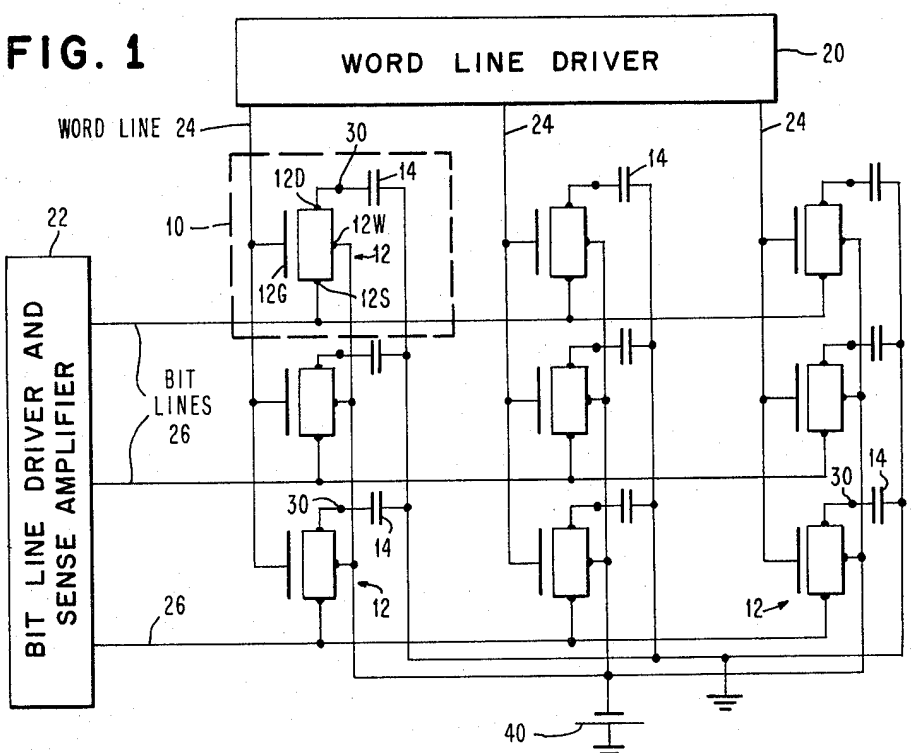
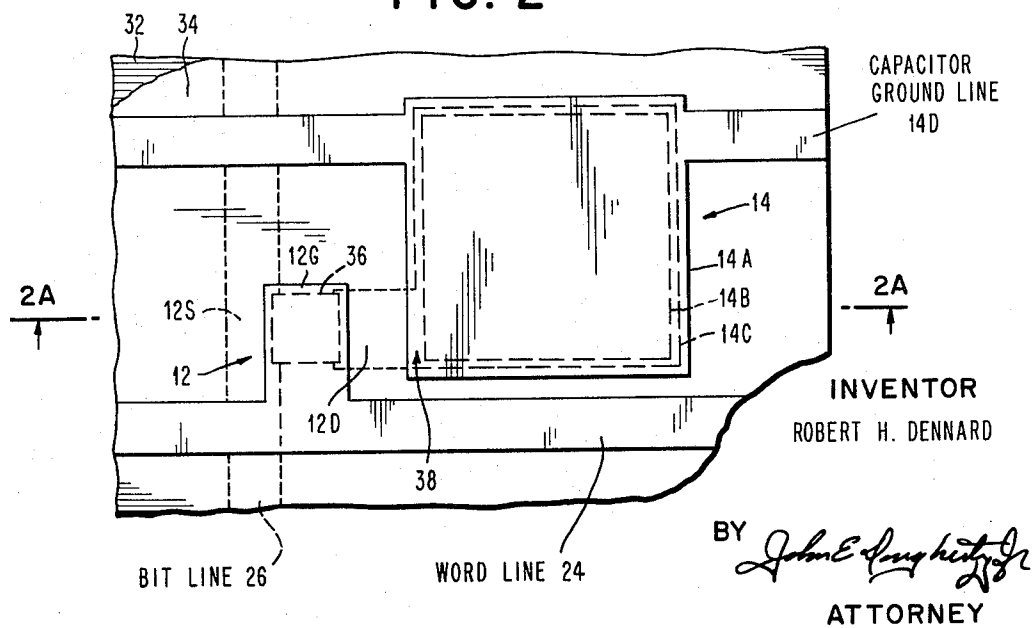
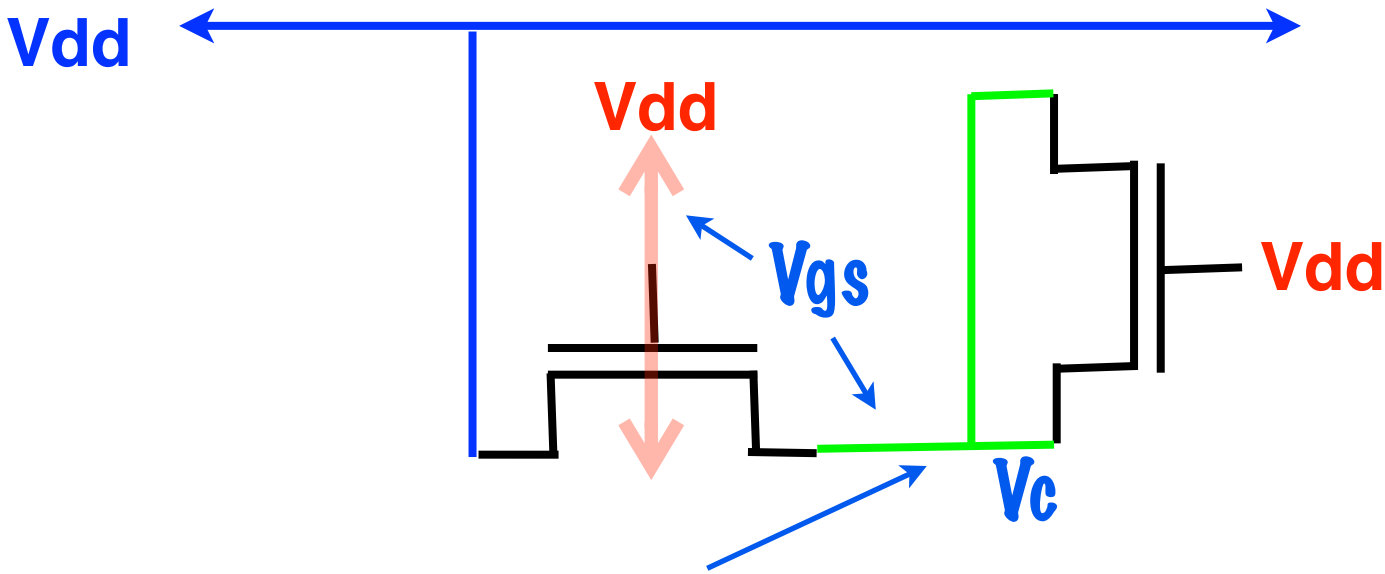


FIG. 2



DRAM Circuit Challenge #1: Writing

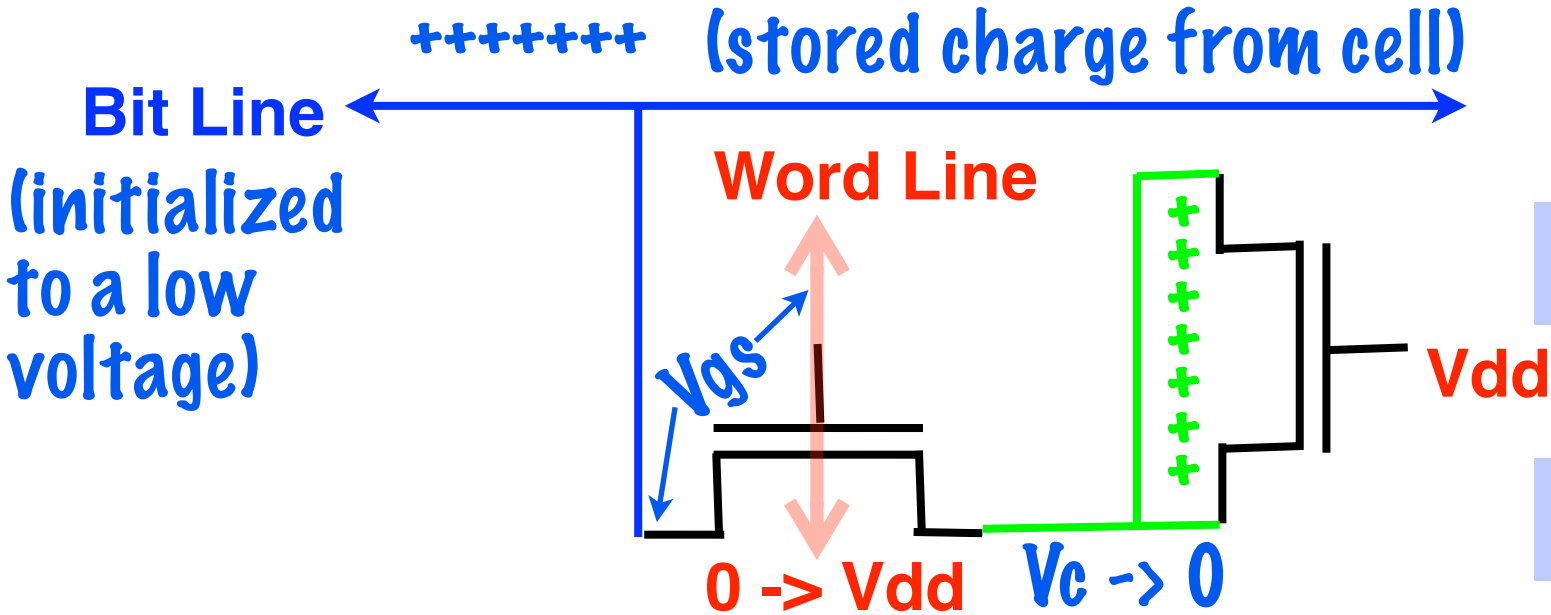


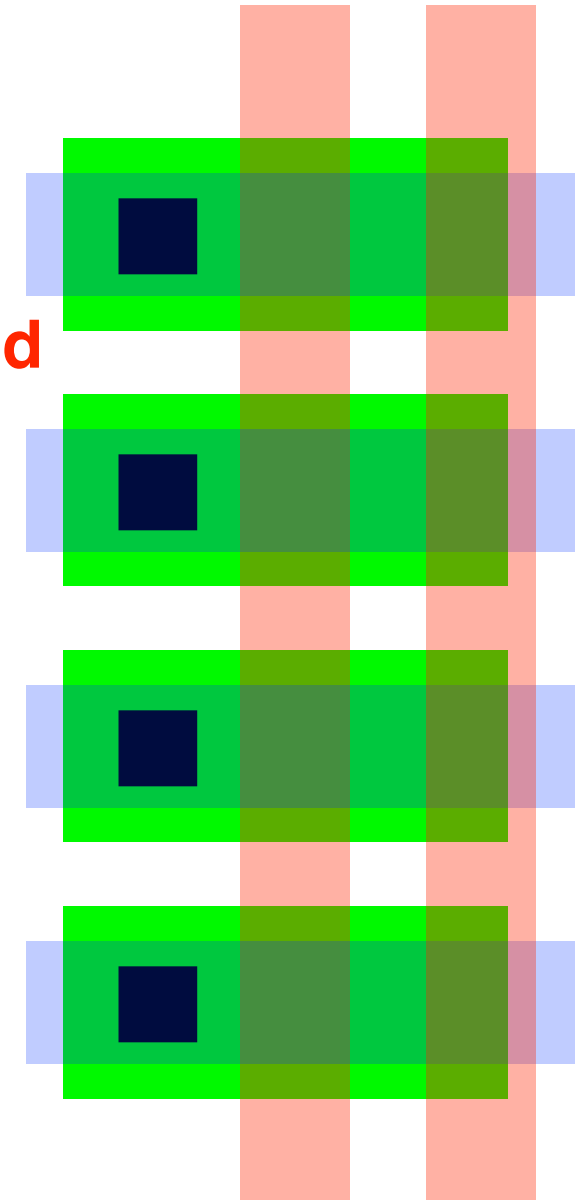
Vdd - Vth. Bad, we store less charge. Why do we not get Vdd?

Ids = k [Vgs -Vth]^2 ,
but "turns off" when Vgs <= Vth!

Vgs = Vdd - Vc. When Vdd - Vc == Vth, charging effectively stops!

DRAM Challenge #2: Destructive Reads

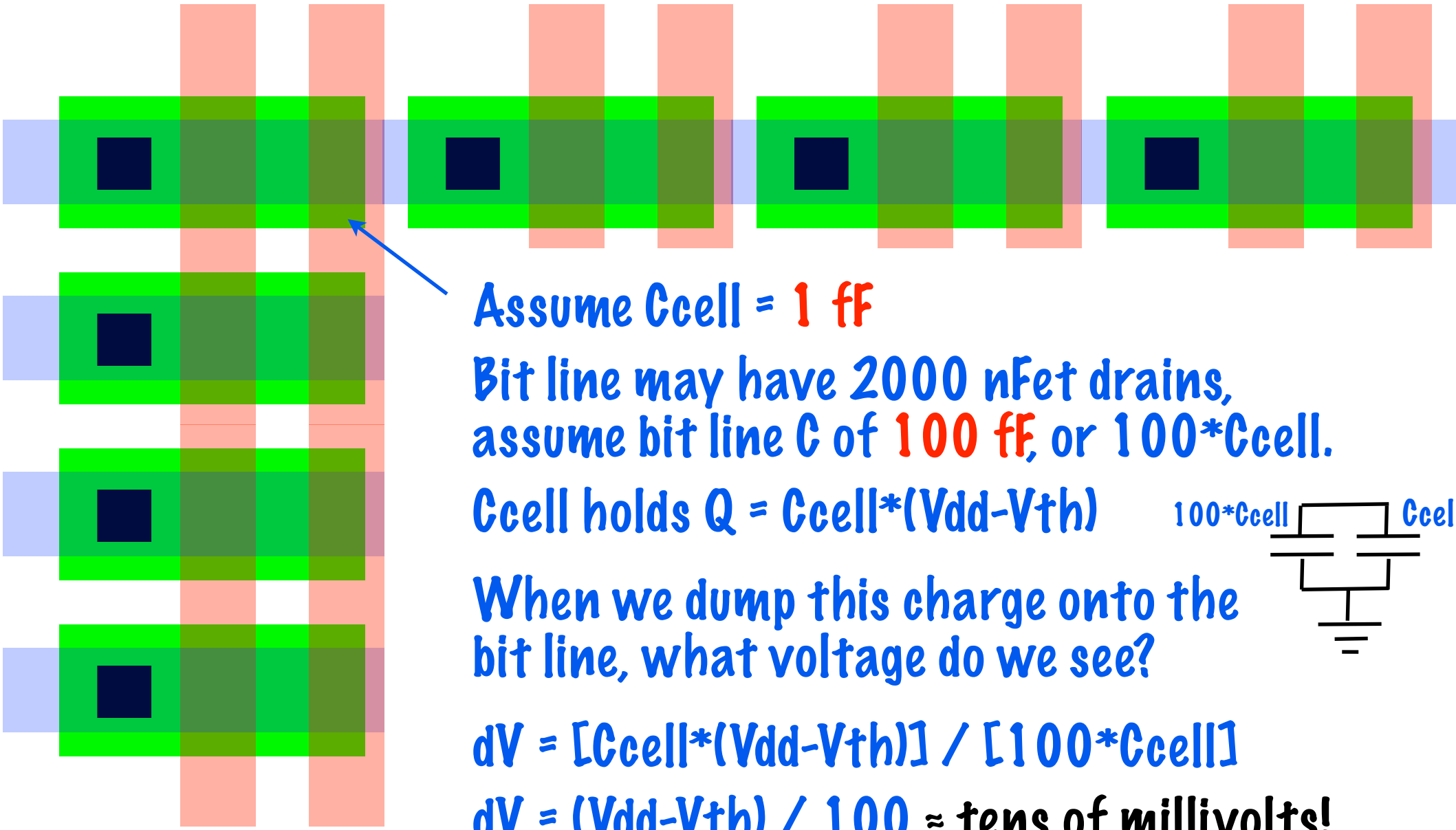




Raising the word line removes the charge from every cell it connects to!

DRAMs write back after each read.

DRAM Circuit Challenge #3a: Sensing



Assume Ccell = 1 fF

Bit line may have 2000 nFet drains, assume bit line C of 100 fF, or 100*Ccell.

Ccell holds Q = Ccell*(Vdd-Vth)

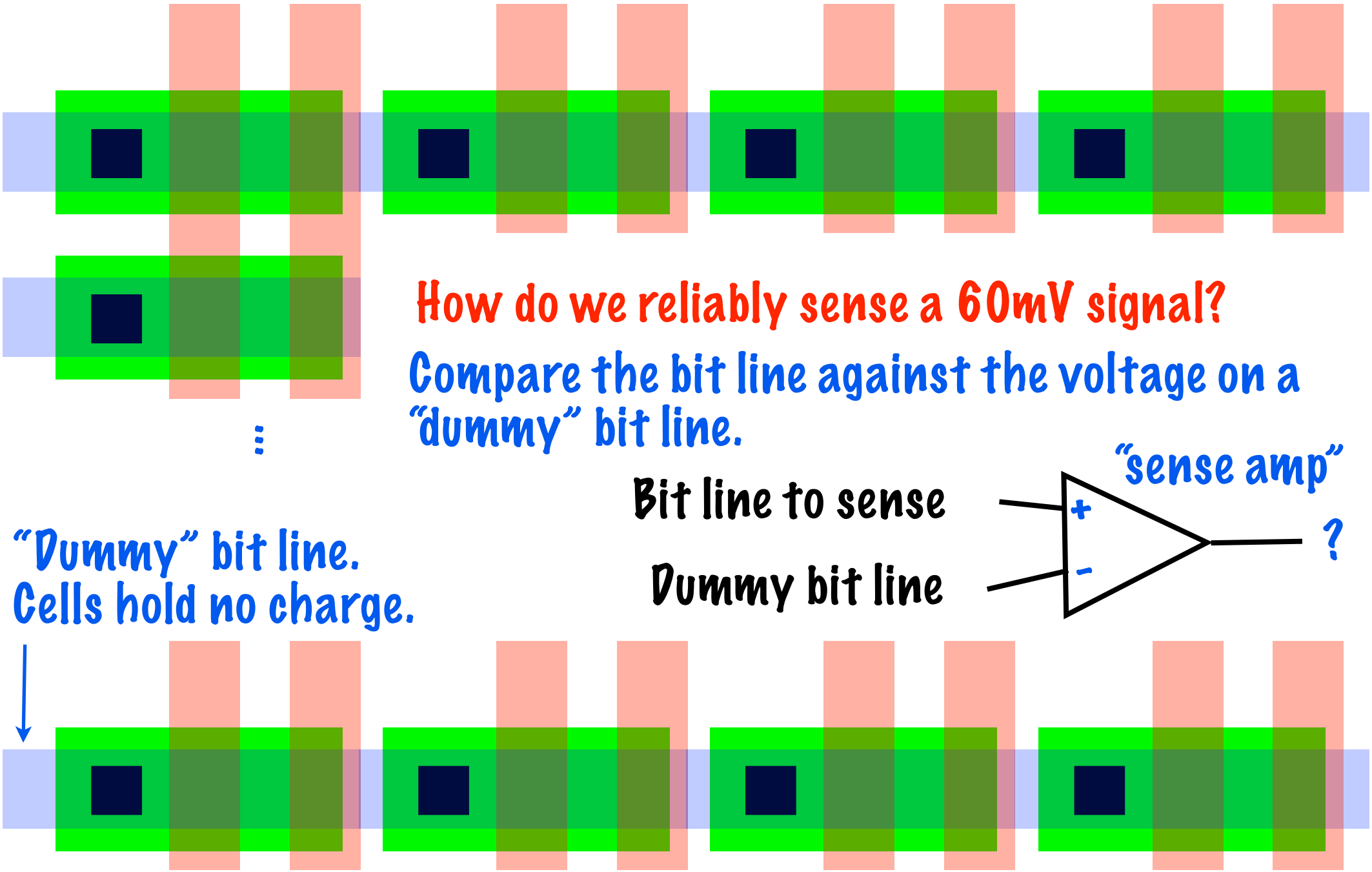
When we dump this charge onto the bit line, what voltage do we see?

dV = [Ccell*(Vdd-Vth)] / [100*Ccell]

dV = (Vdd-Vth) / 100 ≈ tens of millivolts!

In practice, scale array to get a 60mV signal.

DRAM Circuit Challenge #3b: Sensing



How do we reliably sense a 60mV signal?

Compare the bit line against the voltage on a "dummy" bit line.

"sense amp"

Bit line to sense

Dummy bit line

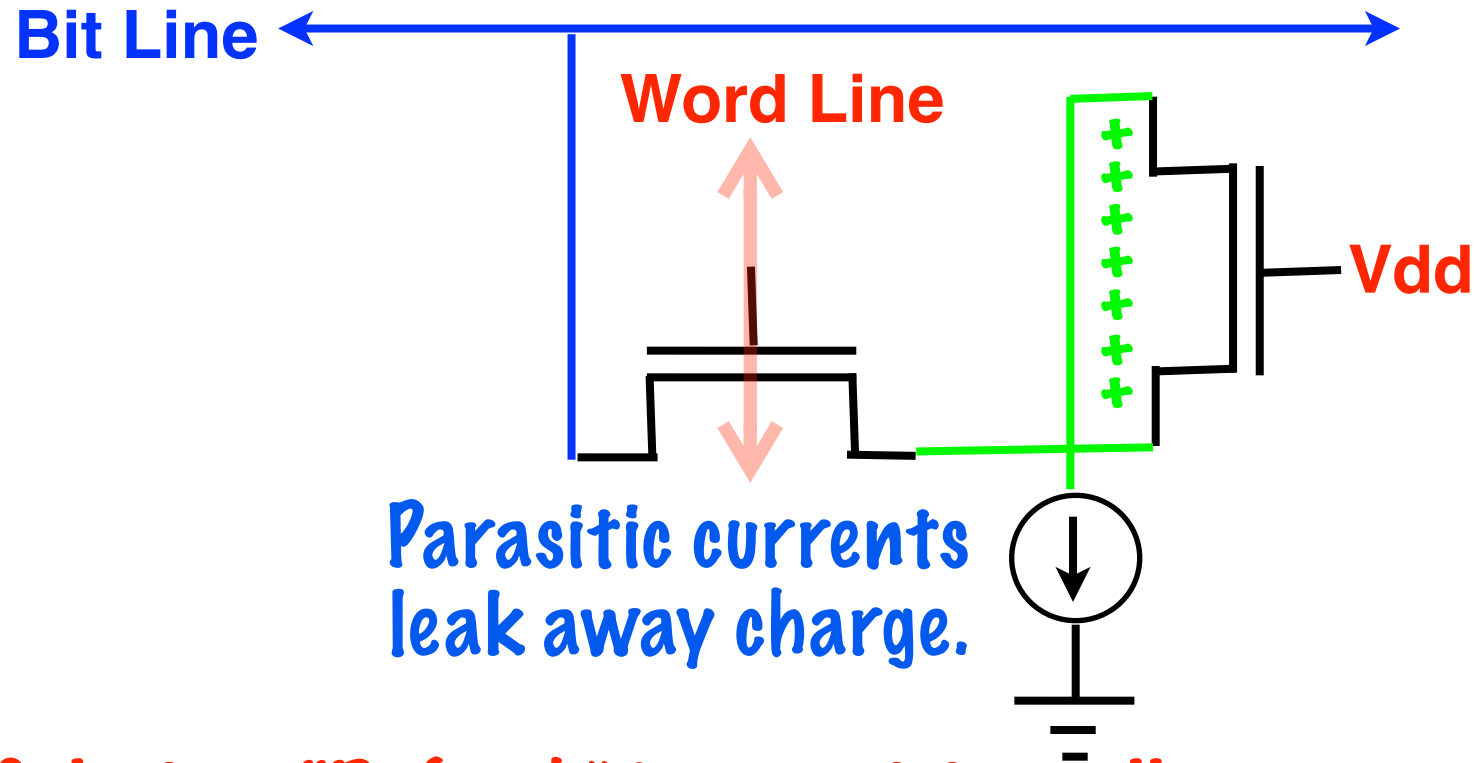
+

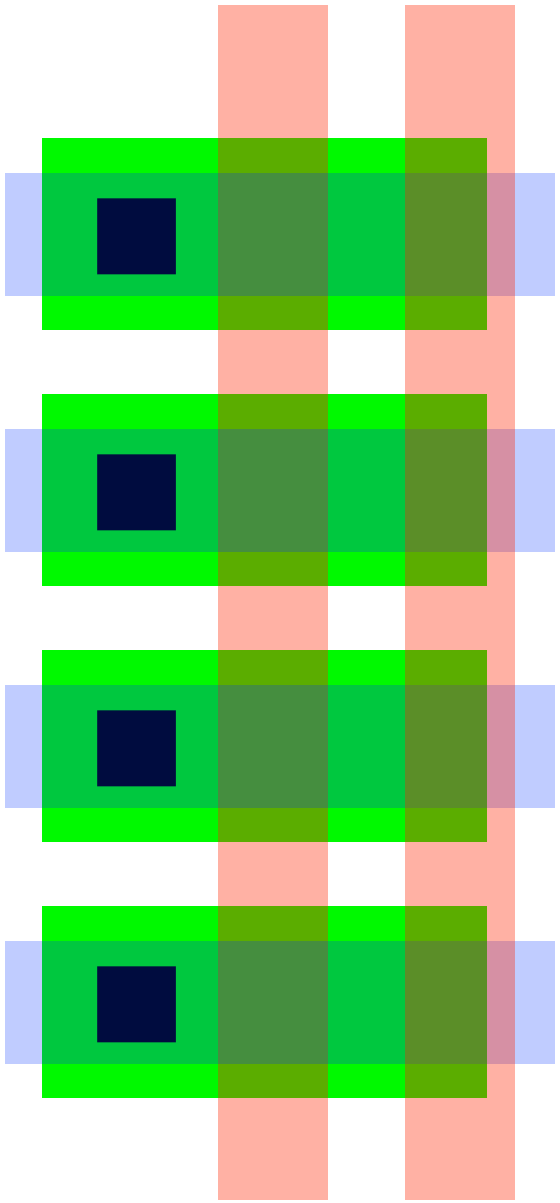
-

?

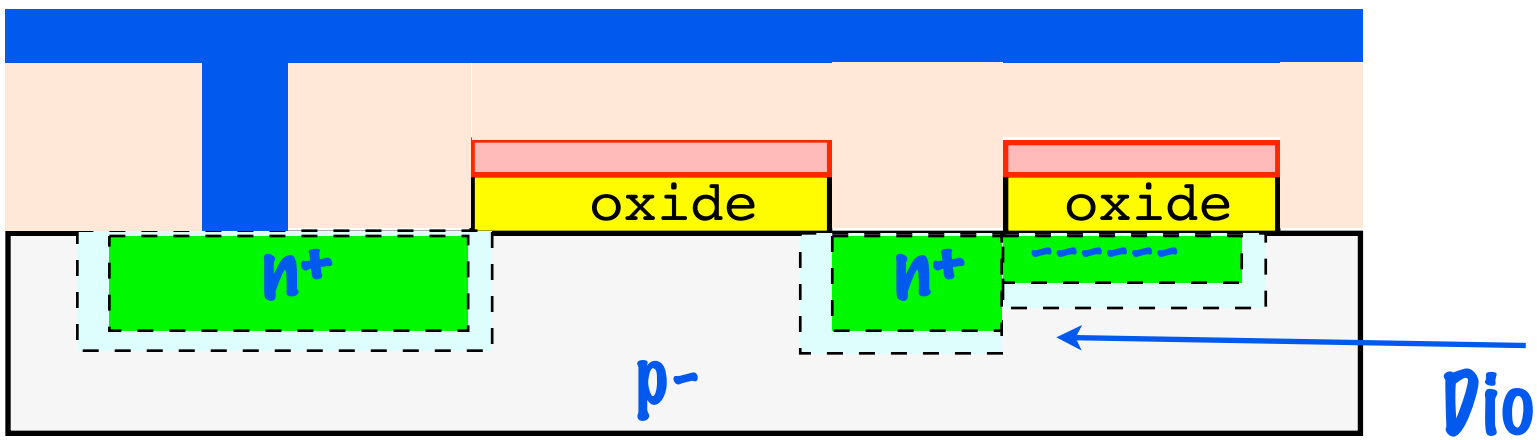
"Dummy" bit line.
Cells hold no charge.

DRAM Challenge #4: Leakage ...



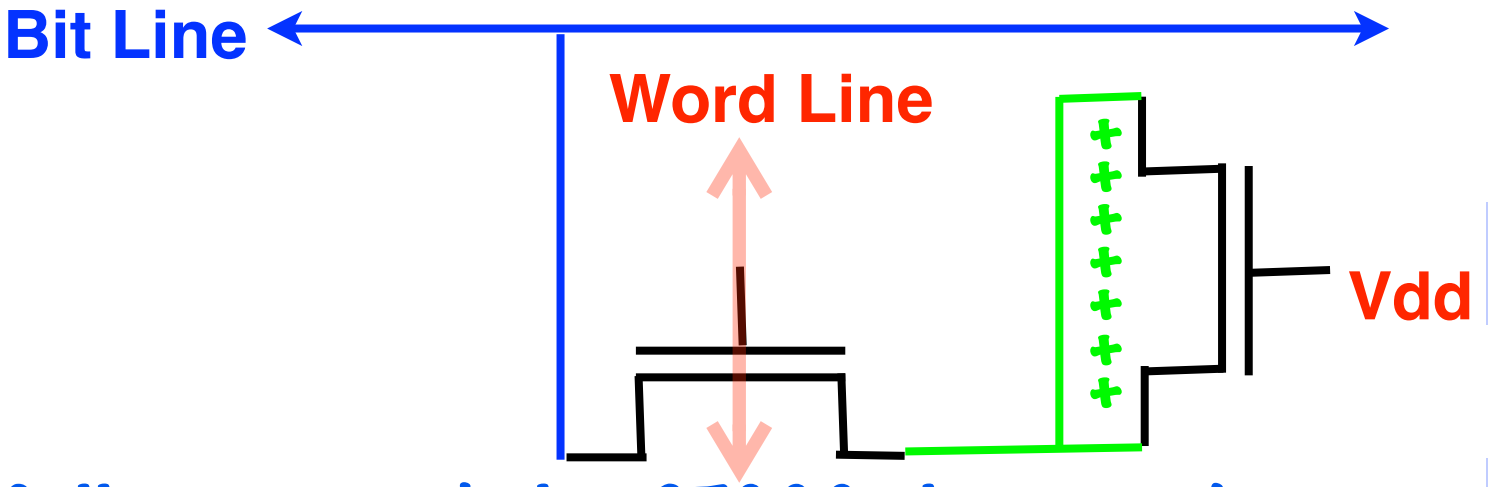


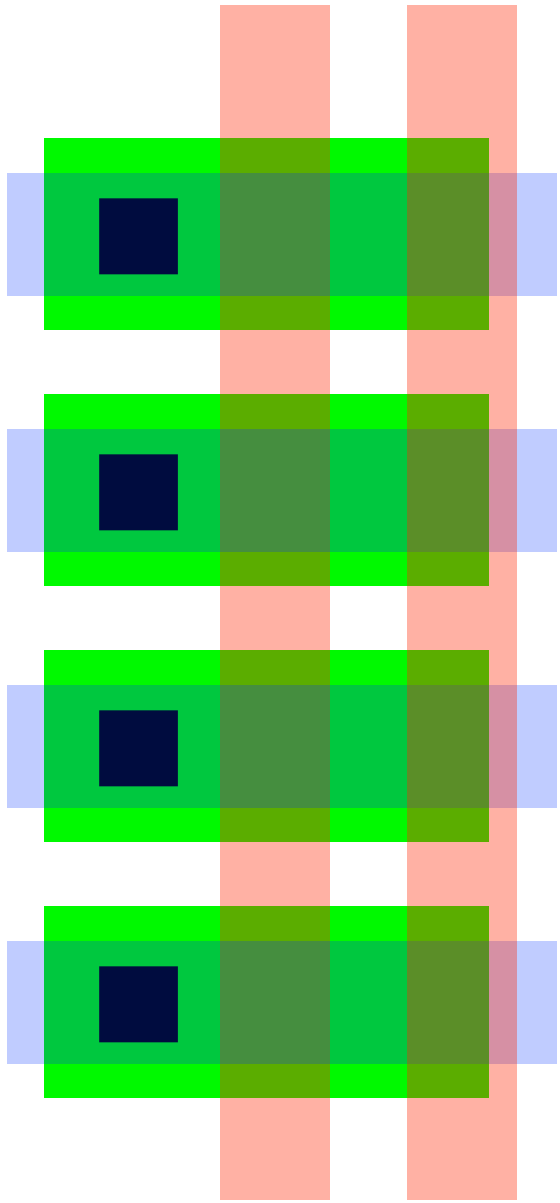
Solution: "Refresh", by rewriting cells at regular intervals (tens of milliseconds)



Diode leakage ...

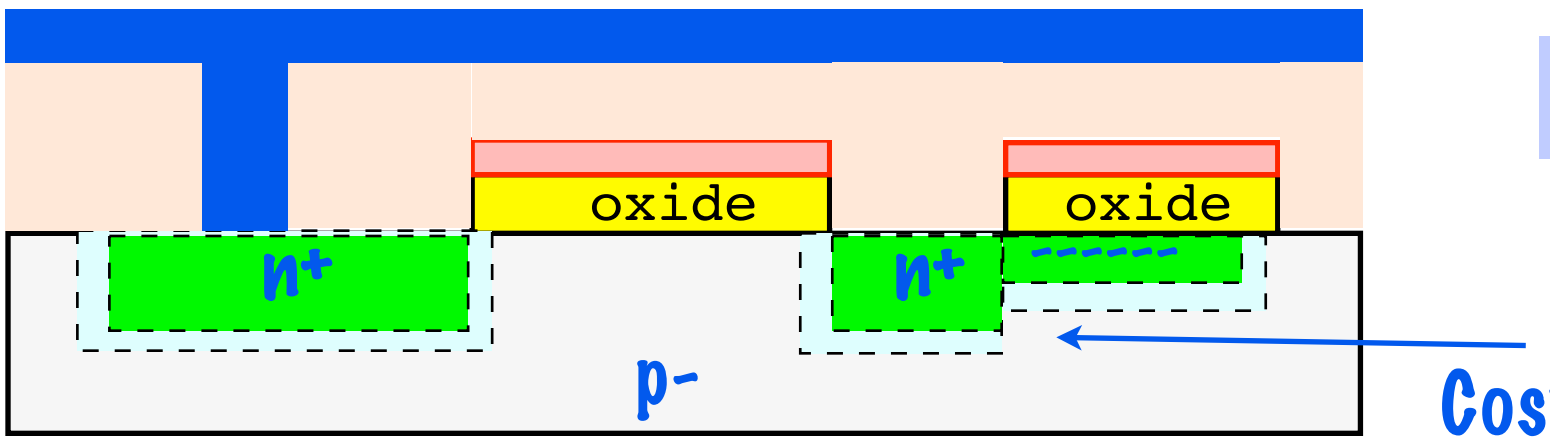
DRAM Challenge #5: Cosmic Rays ...





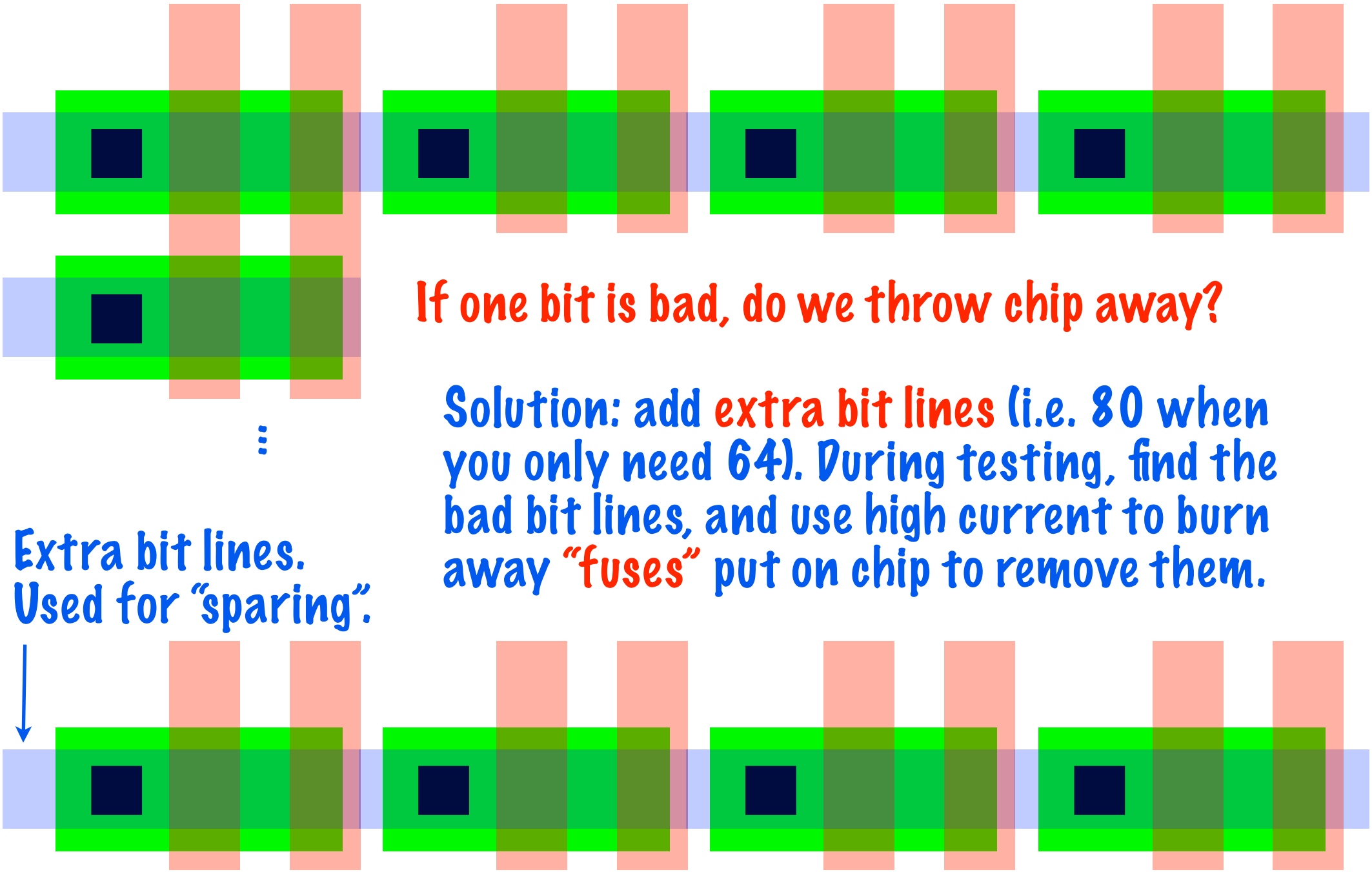
Cell capacitor holds 25,000 electrons (or less). Cosmic rays that constantly bombard us can release the charge!

Solution: Store extra bits to detect and correct random bit flips (ECC).



Cosmic ray hit.

DRAM Challenge #6: Yield

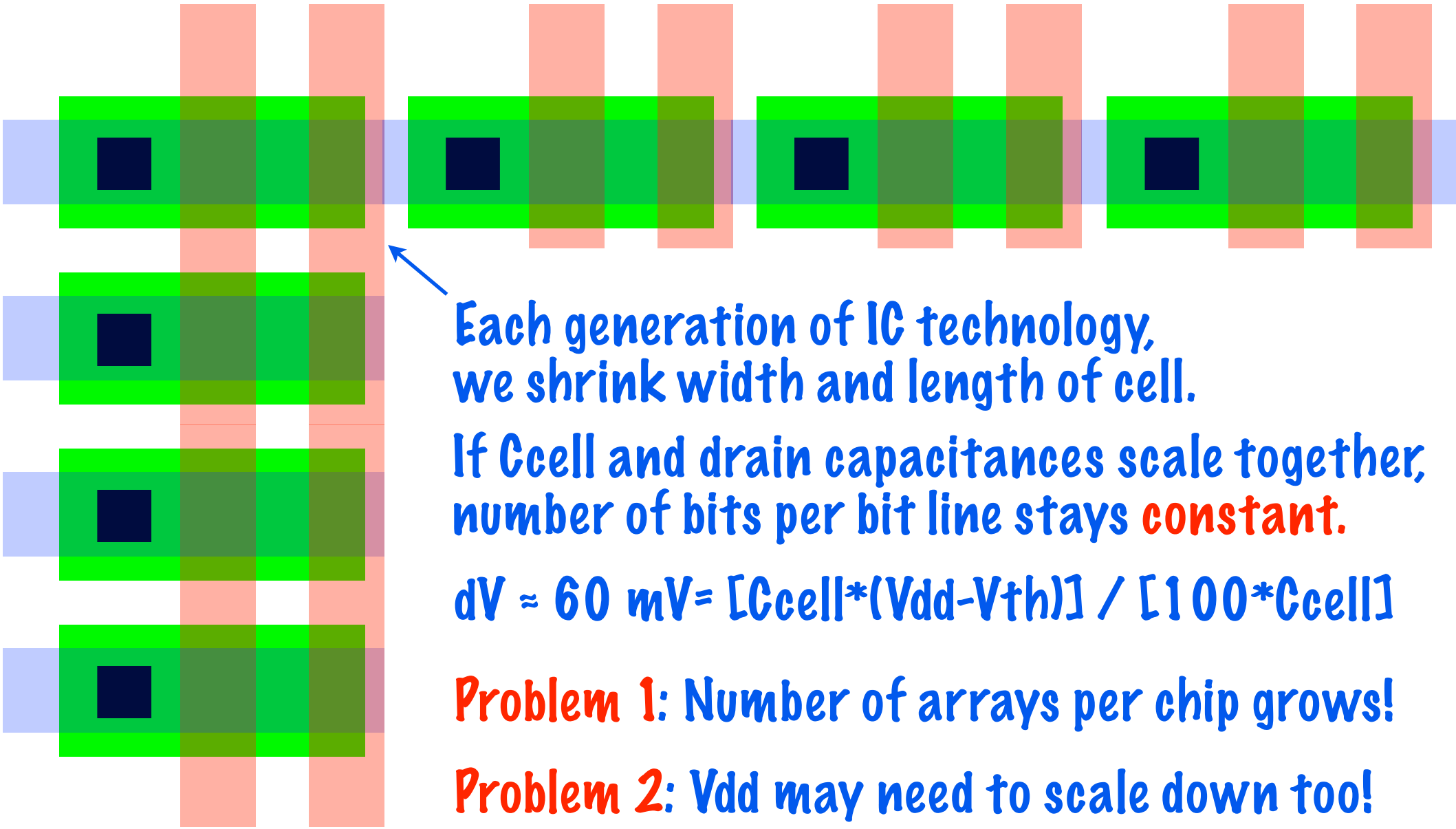


If one bit is bad, do we throw chip away?

Solution: add extra bit lines (i.e. 80 when you only need 64). During testing, find the bad bit lines, and use high current to burn away "fuses" put on chip to remove them.

Extra bit lines.
Used for "sparing".

DRAM Challenge #7: Scaling



Each generation of IC technology, we shrink width and length of cell.

If Ccell and drain capacitances scale together, number of bits per bit line stays constant.

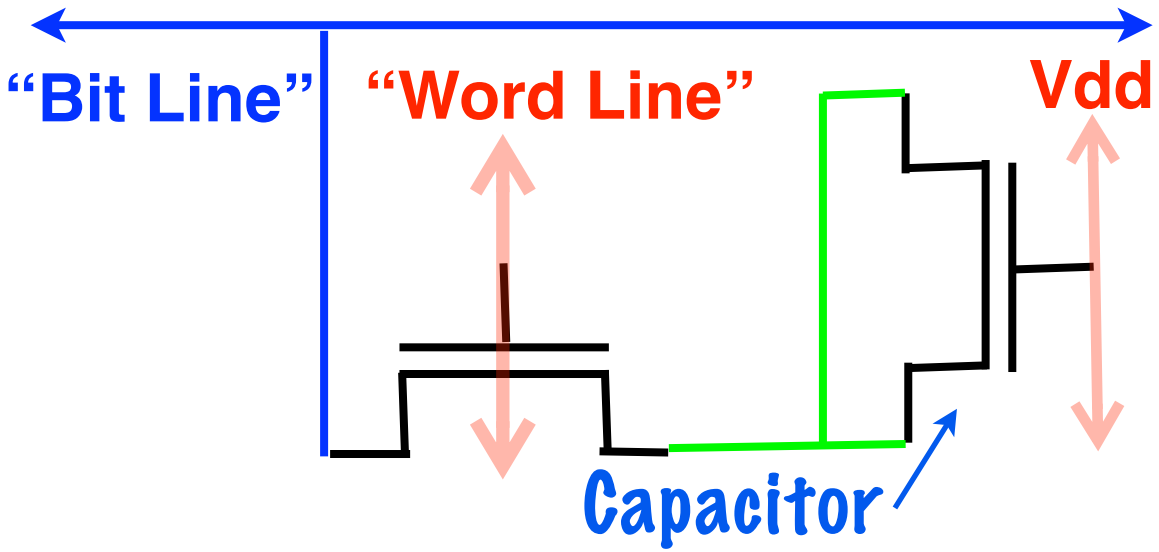
dV ≈ 60 mV= [Ccell*(Vdd-Vth)] / [100*Ccell]

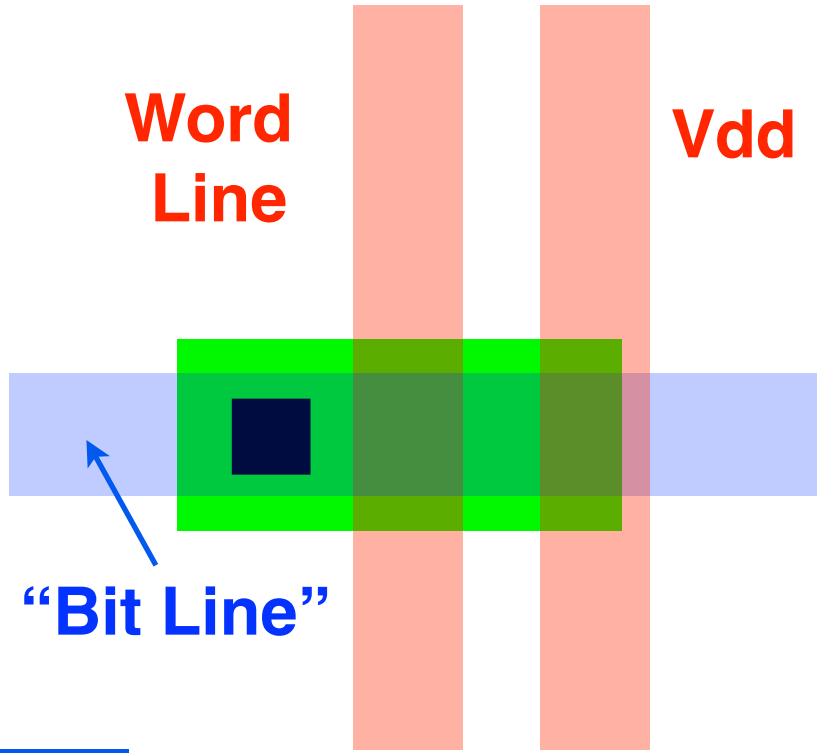
Problem 1: Number of arrays per chip grows!

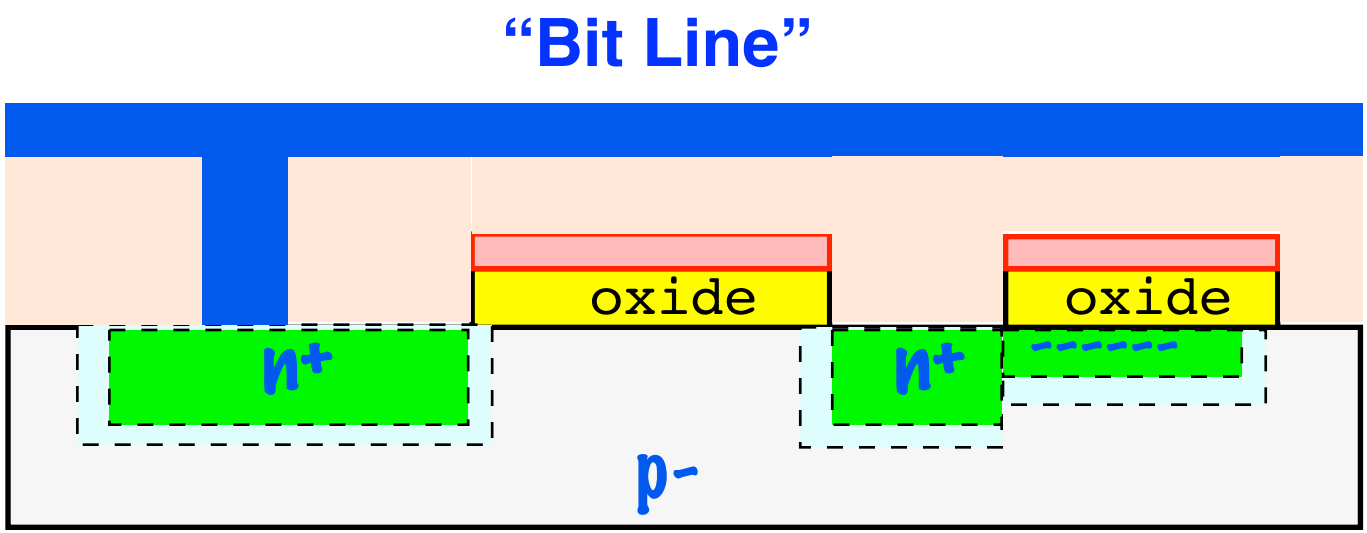
Problem 2: Vdd may need to scale down too!

Solution: Constant Innovation of Cell Capacitors!

Poly-diffusion Ccell is ancient history







Word Line and Vdd run on "z-axis"

Early replacement: “Trench” capacitors

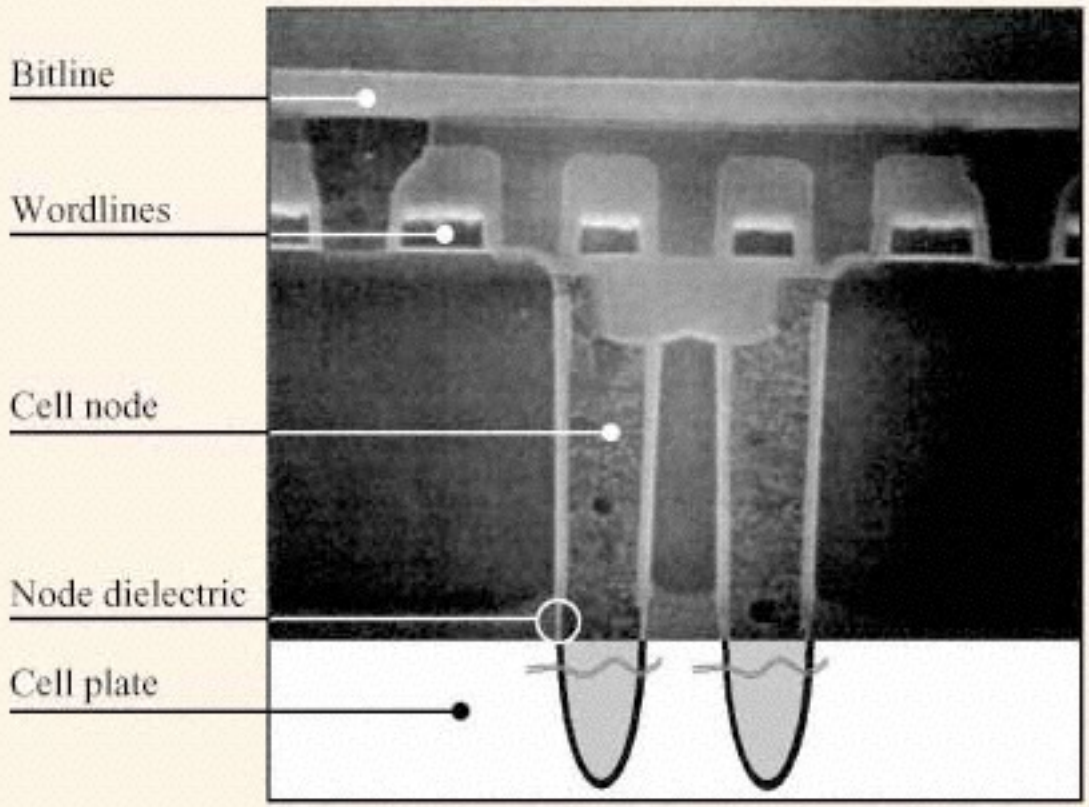
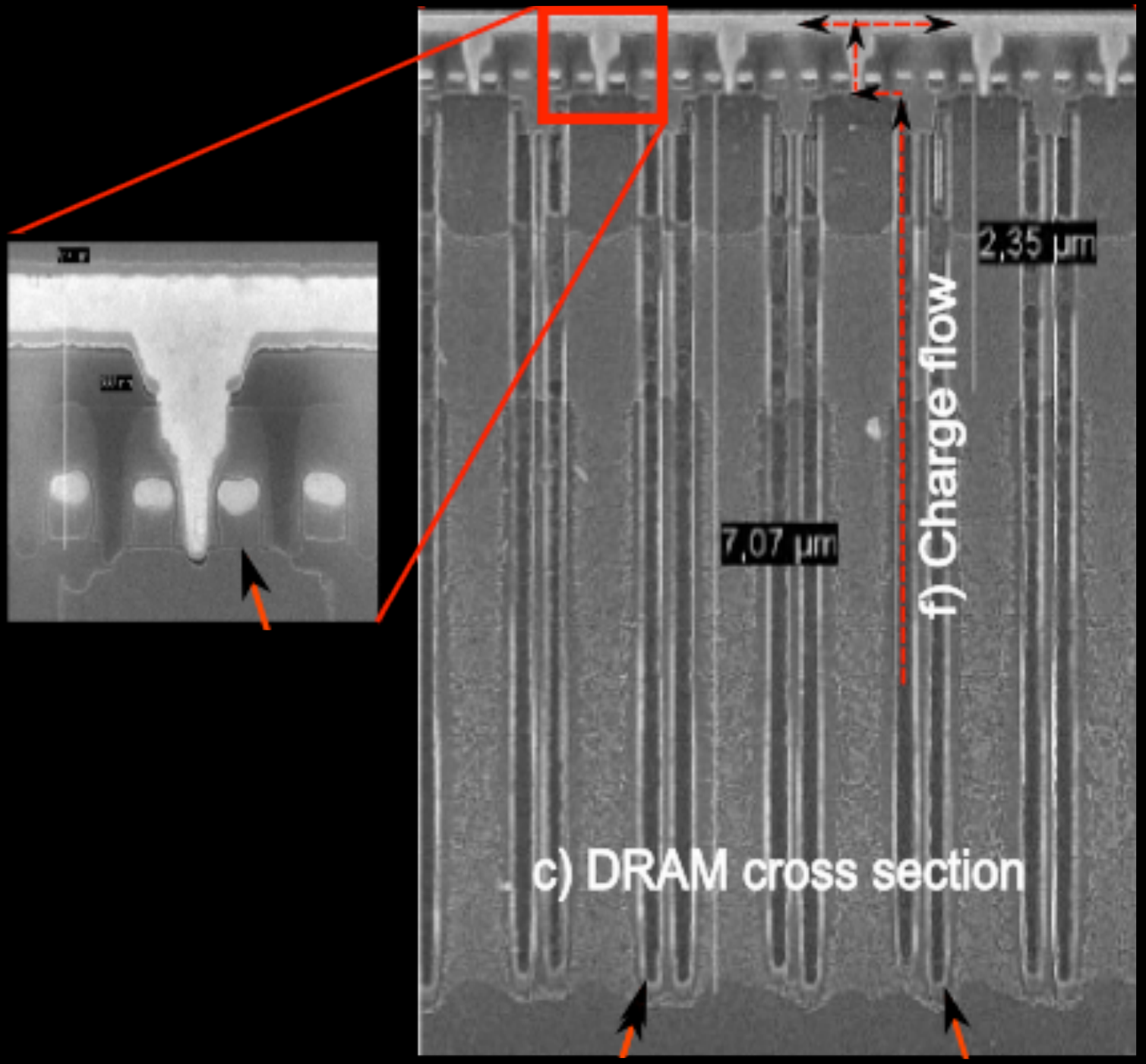


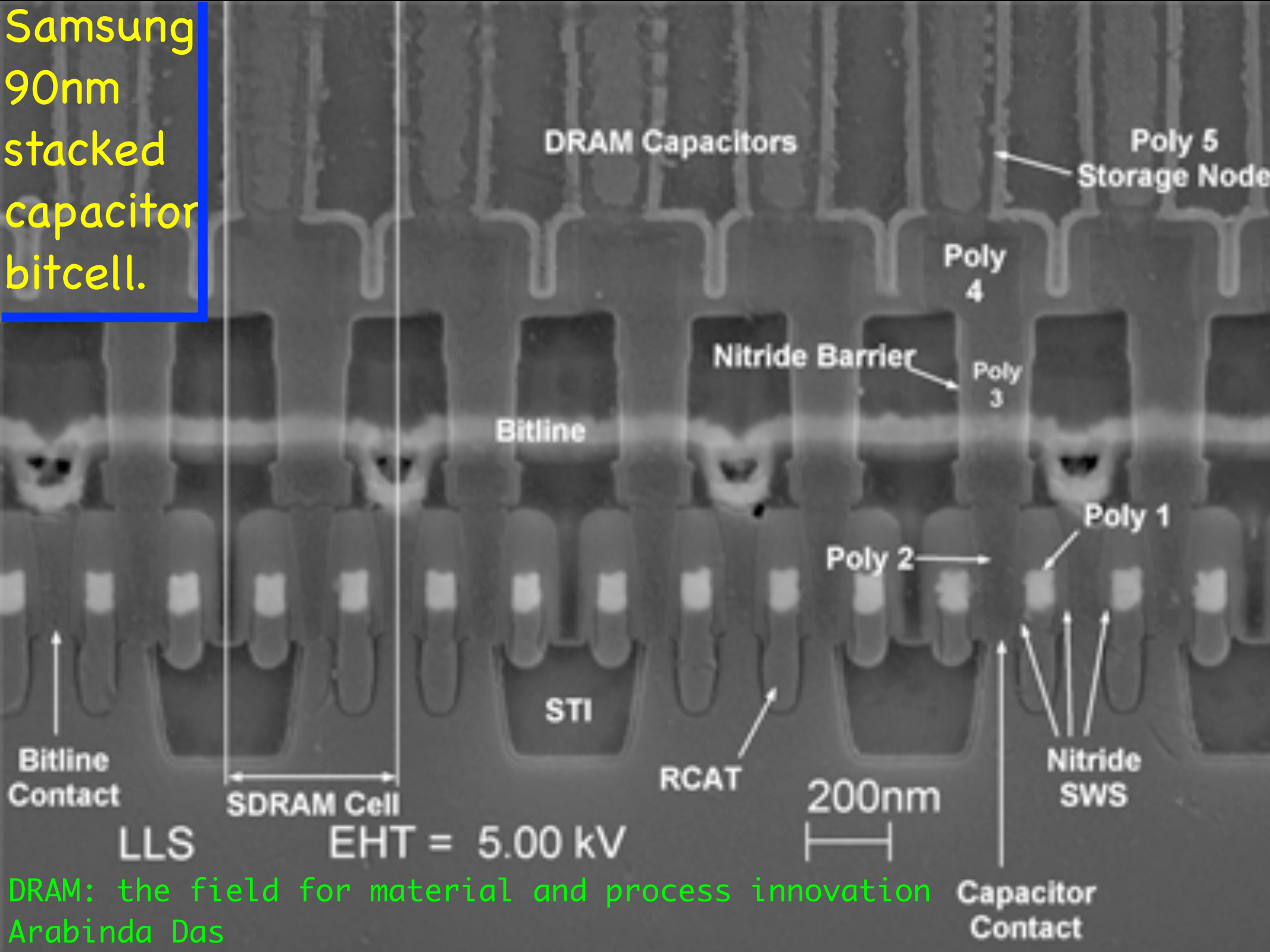
Figure 4

SEM photomicrograph of 0.25-μm trench DRAM cell suitable for scaling to 0.15μm and below. Reprinted with permission from [17]; © 1995 IEEE.

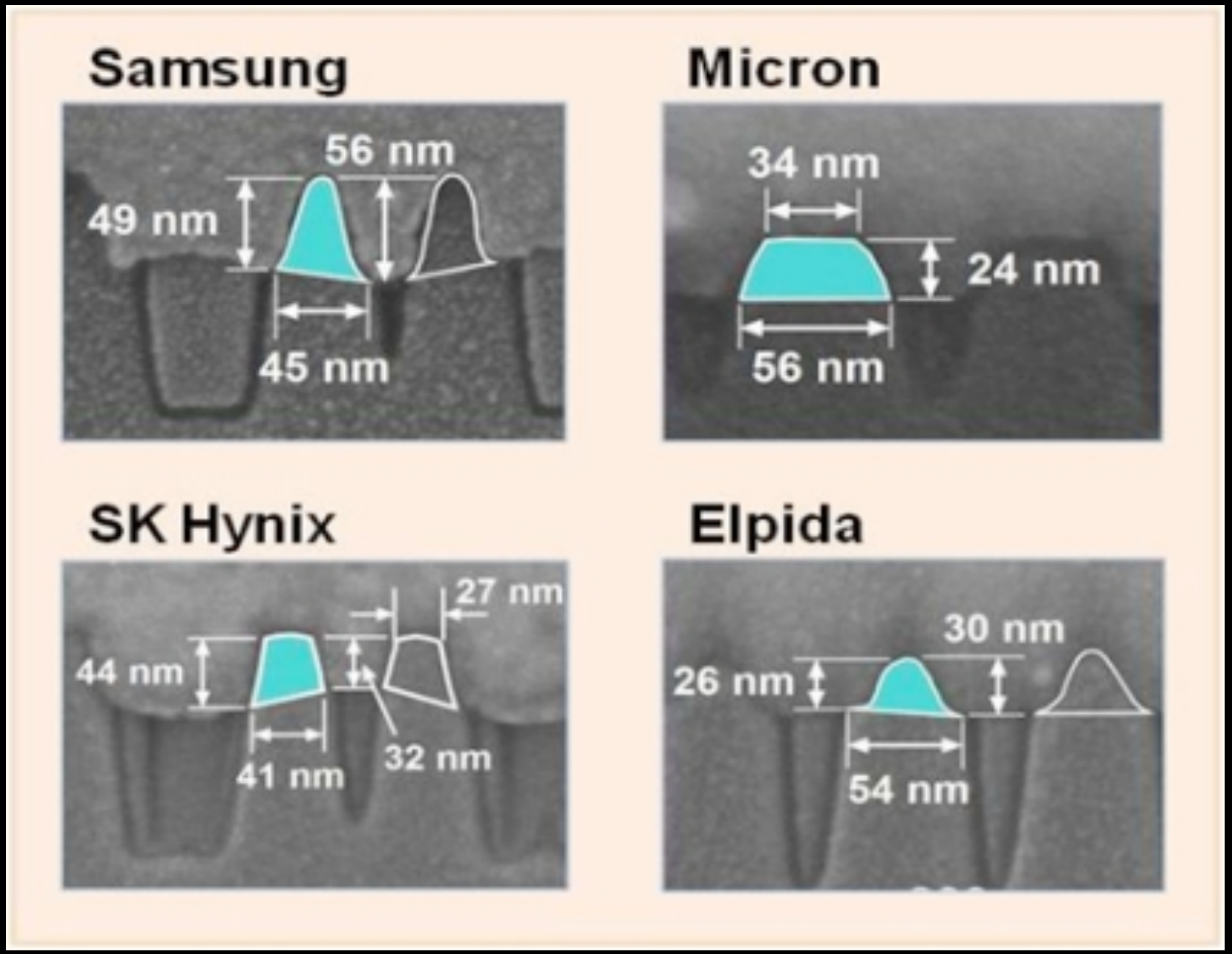
Final generation of trench capacitors



The companies that kept scaling trench capacitors for commodity DRAM chips went out of business.


Samsung 90nm stacked capacitor bitcell.
DRAM Capacitors
Poly 5 Storage Node
Poly 4
Nitride Barrier
Poly 3
Bitline
Poly 1
Poly 2
STI
Bitline Contact
SDRAM Cell
RCAT
200nm
Nitride SWS
Capacitor Contact
LLS
EHT = 5.00 kV
DRAM: the field for material and process innovation
Arabinda Das

Cell access transistors for the 4 leading vendors

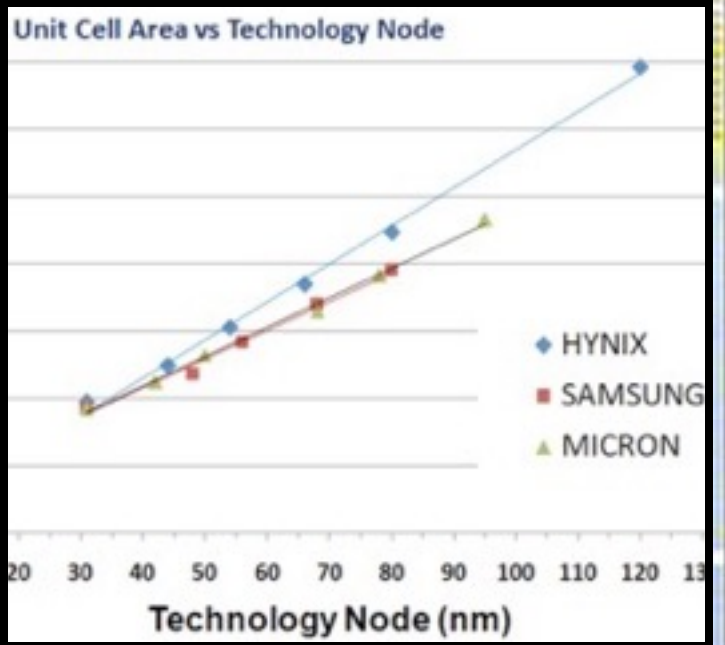


Chipmakers turn to new process for sub-nm DRAM cells

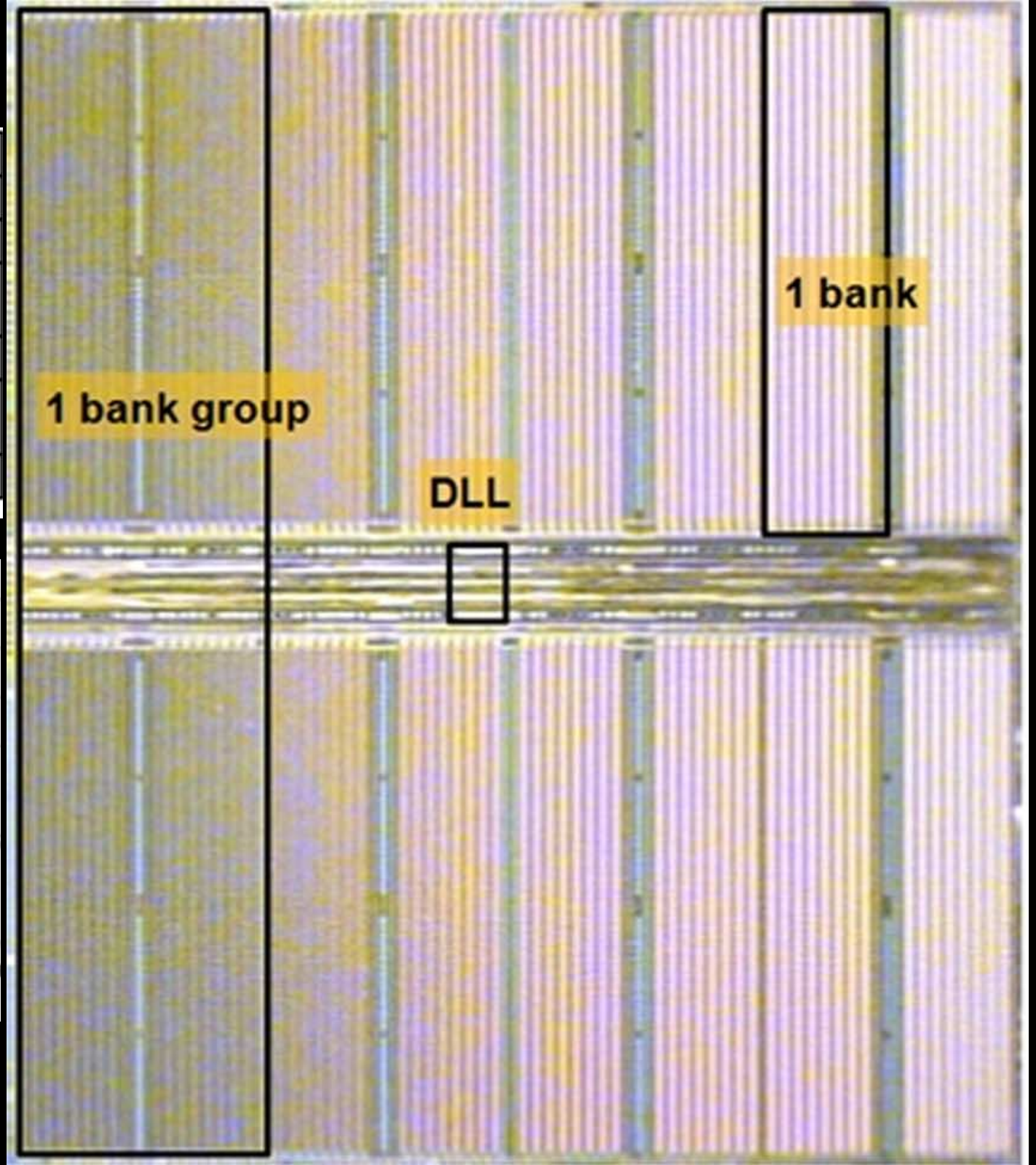
Jeongdong Choe, TechInsights

Samsung 30nm

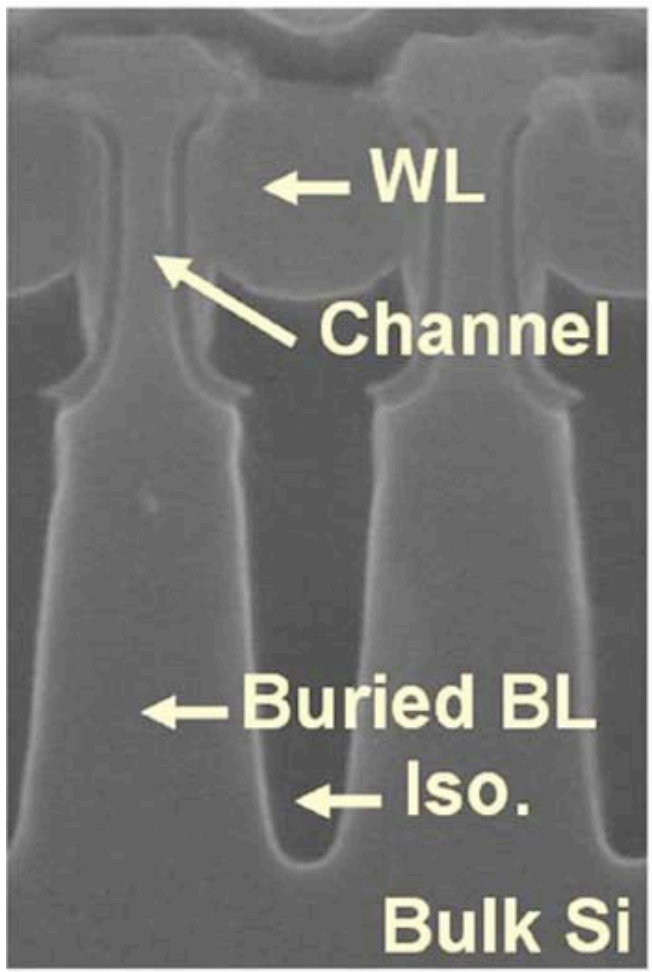
Process	30nm CMOS 3 metal
Capacity	4Gb
Chip configuration	512Mx8b or 1Gx4b
Supply voltage	1.2V VDD/VDDQ 2.5V VPP
Data rate	3.2Gb/s/pin@1.14V
Operating current (IDD4R@1.26V)	62.0mA@DDR1600 109.8mA@DDR3200
Chip size	76mm^2

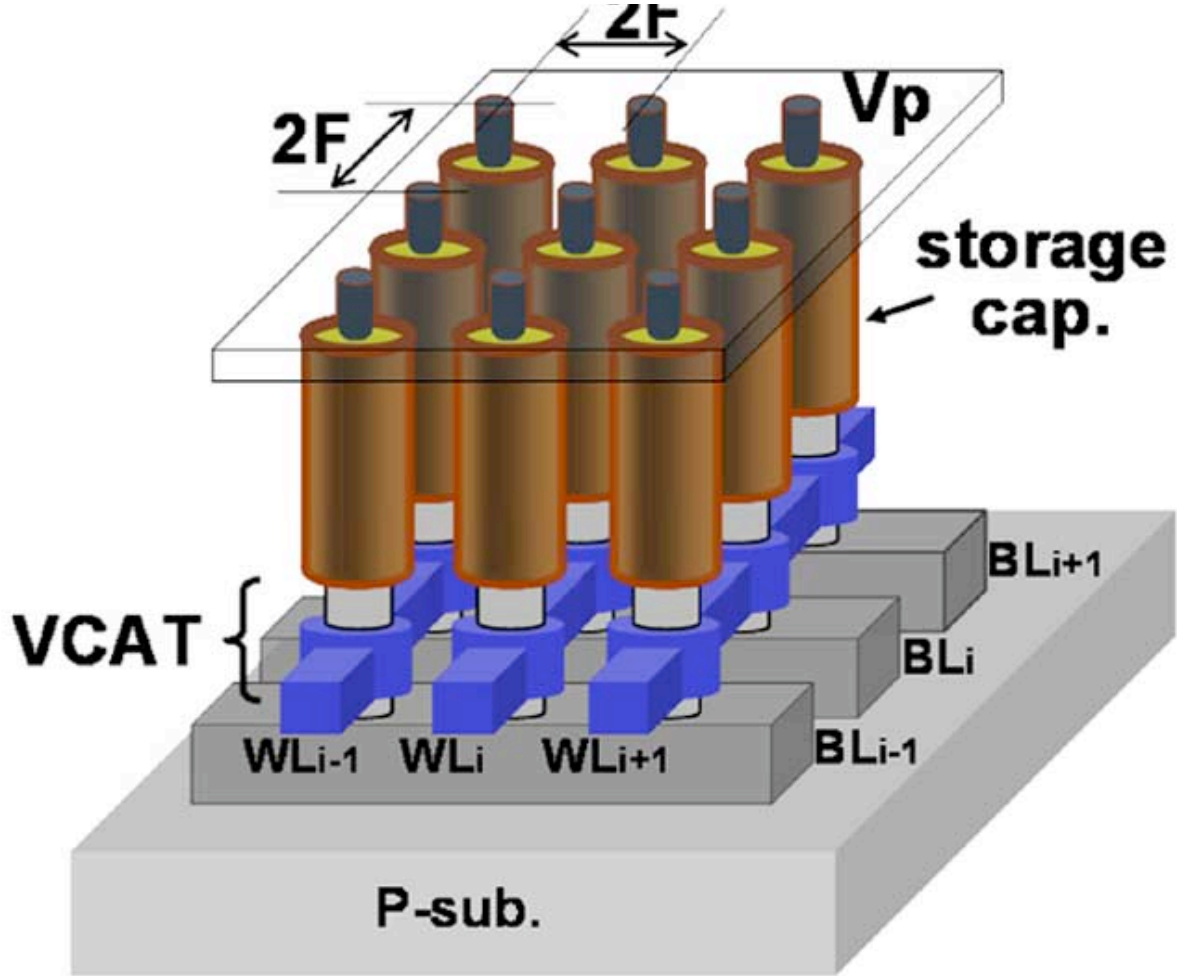


From JSSC, and Arabinda Das



In the labs: Vertical cell transistors ...



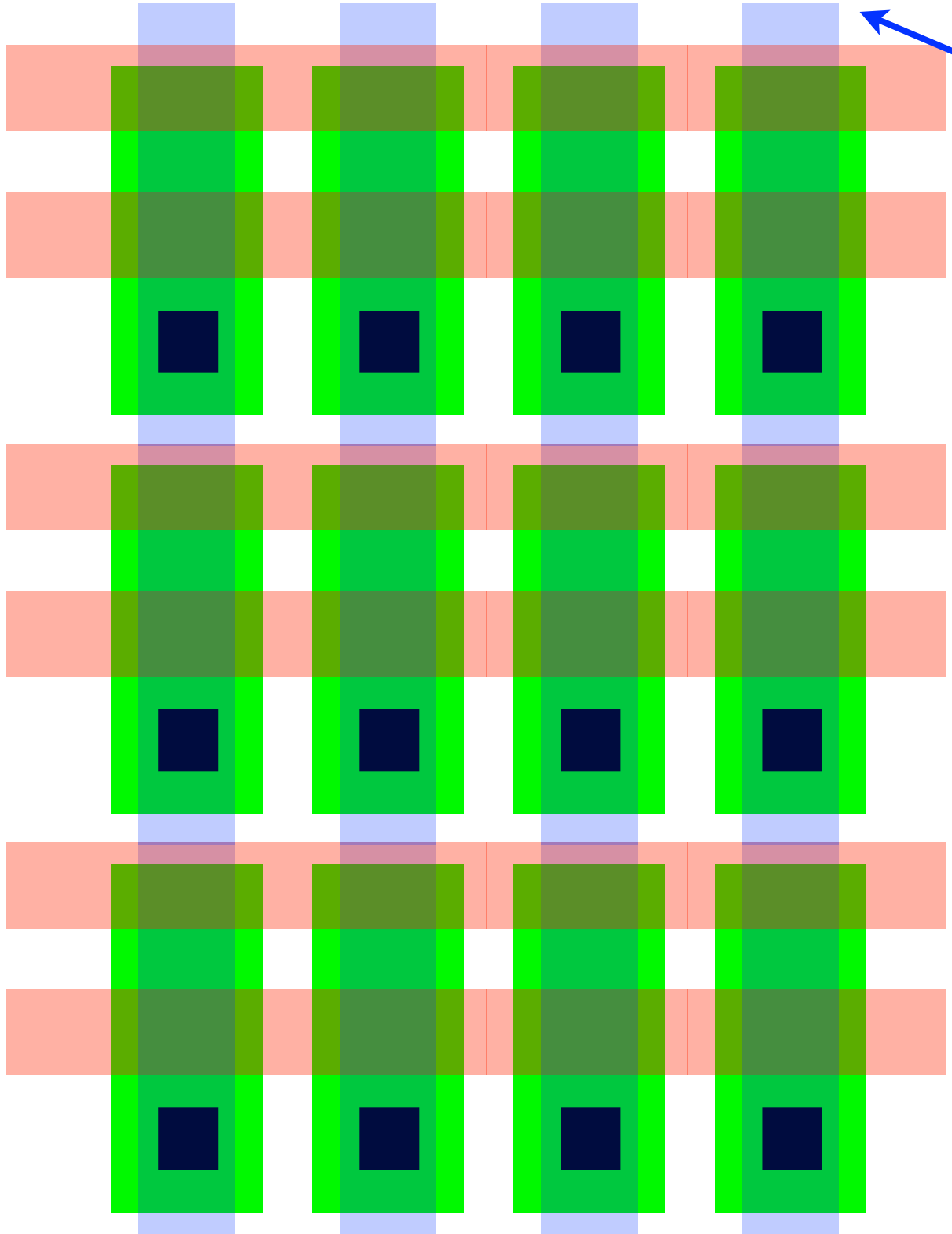


880 IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 45, NO. 4, APRIL 2010

A 31 ns Random Cycle VCAT-Based $4F^2$ DRAM With Manufacturability and Enhanced Cell Efficiency

Ki-Whan Song, Jin-Young Kim, Jae-Man Yoon, Sua Kim, Huijung Kim, Hyun-Woo Chung, Hyungi Kim, Kanguk Kim, Hwan-Wook Park, Hyun Chul Kang, Nam-Kyun Tak, Dukha Park, Woo-Seop Kim, *Member, IEEE*, Yeong-Taek Lee, Yong Chul Oh, Gyo-Young Jin, Jeihwan Yoo, Donggun Park, *Senior Member, IEEE*, Kyungseok Oh, Changhyun Kim, *Senior Member, IEEE*, and Young-Hyun Jun

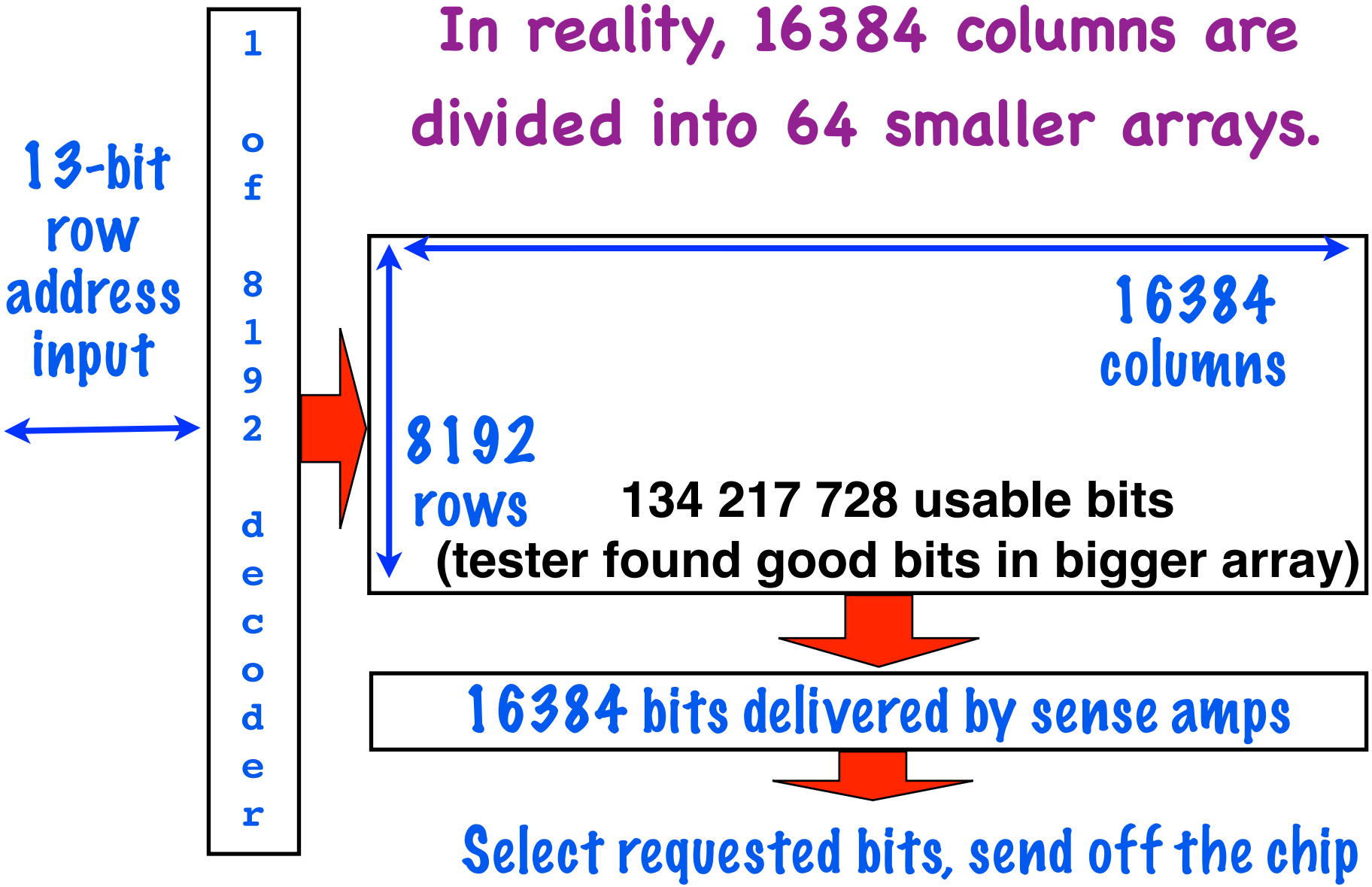
Memory Arrays



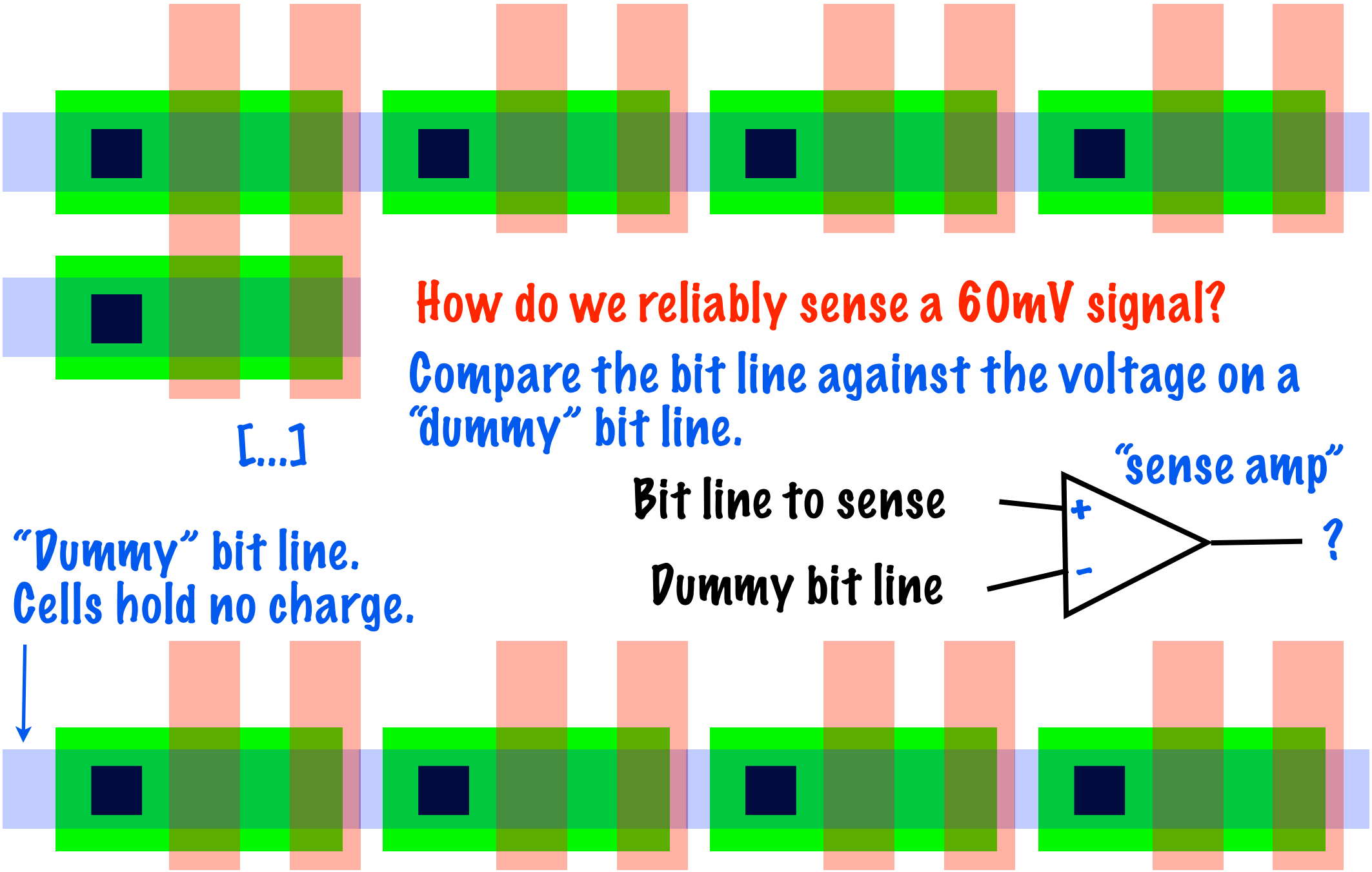
People buy DRAM for the bits. "Edge" circuits are overhead.

So, we amortize the edge circuits over big arrays.

A "bank" of 128 Mb (512Mb chip -> 4 banks)



Recall DRAM Challenge #3b: Sensing



How do we reliably sense a 60mV signal?

Compare the bit line against the voltage on a "dummy" bit line.

[...]

"Dummy" bit line.
Cells hold no charge.

"sense amp"

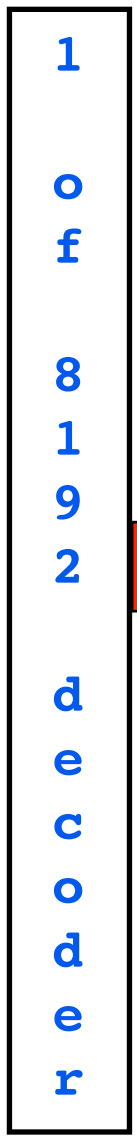
Bit line to sense

Dummy bit line

“Sensing” is row read into sense amps

Slow! This 2.5ns period DRAM (400 MT/s) can do row reads at only 55 ns (18 MHz).

13-bit row address input



DRAM has high latency to first bit out. A fact of life.

16384 columns

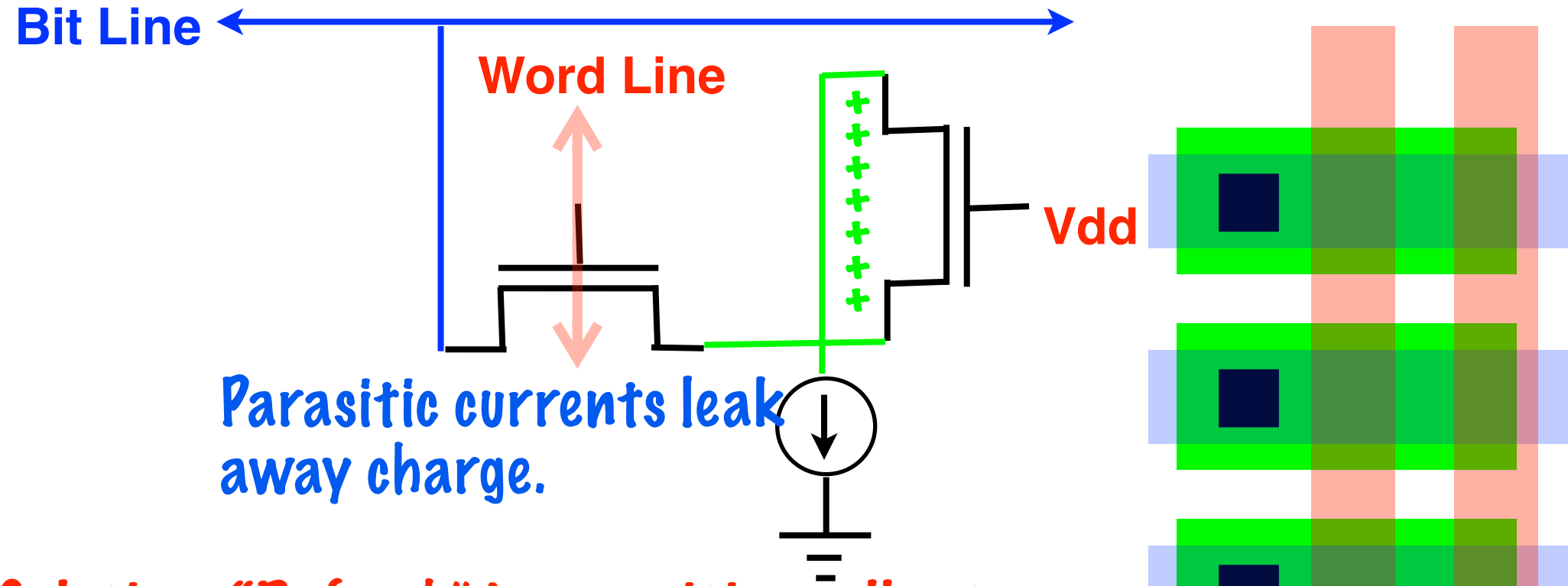
8192 rows

134 217 728 usable bits
(tester found good bits in bigger array)

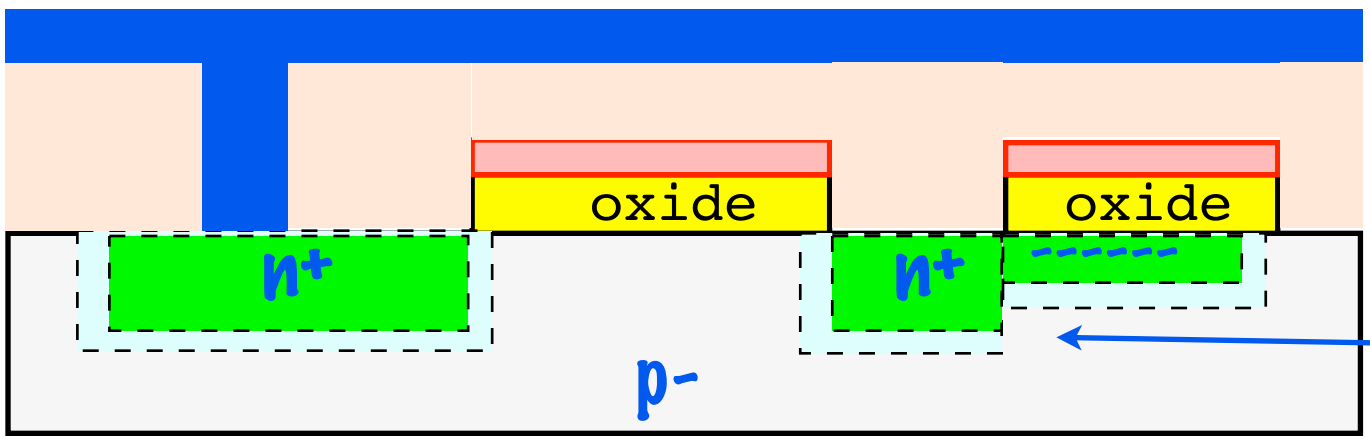
16384 bits delivered by sense amps

Select requested bits, send off the chip

An ill-timed refresh may add to latency



Solution: "Refresh", by rewriting cells at regular intervals (tens of milliseconds)



Diode leakage ...

Latency is not the same as bandwidth!

Thus, push to faster DRAM interfaces

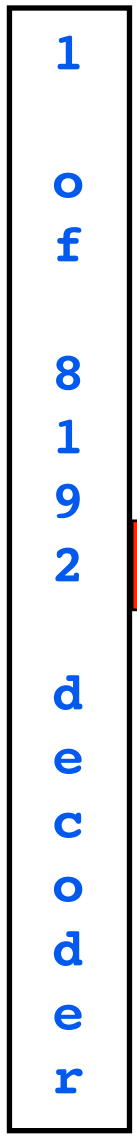
What if we want all of the 16384 bits?

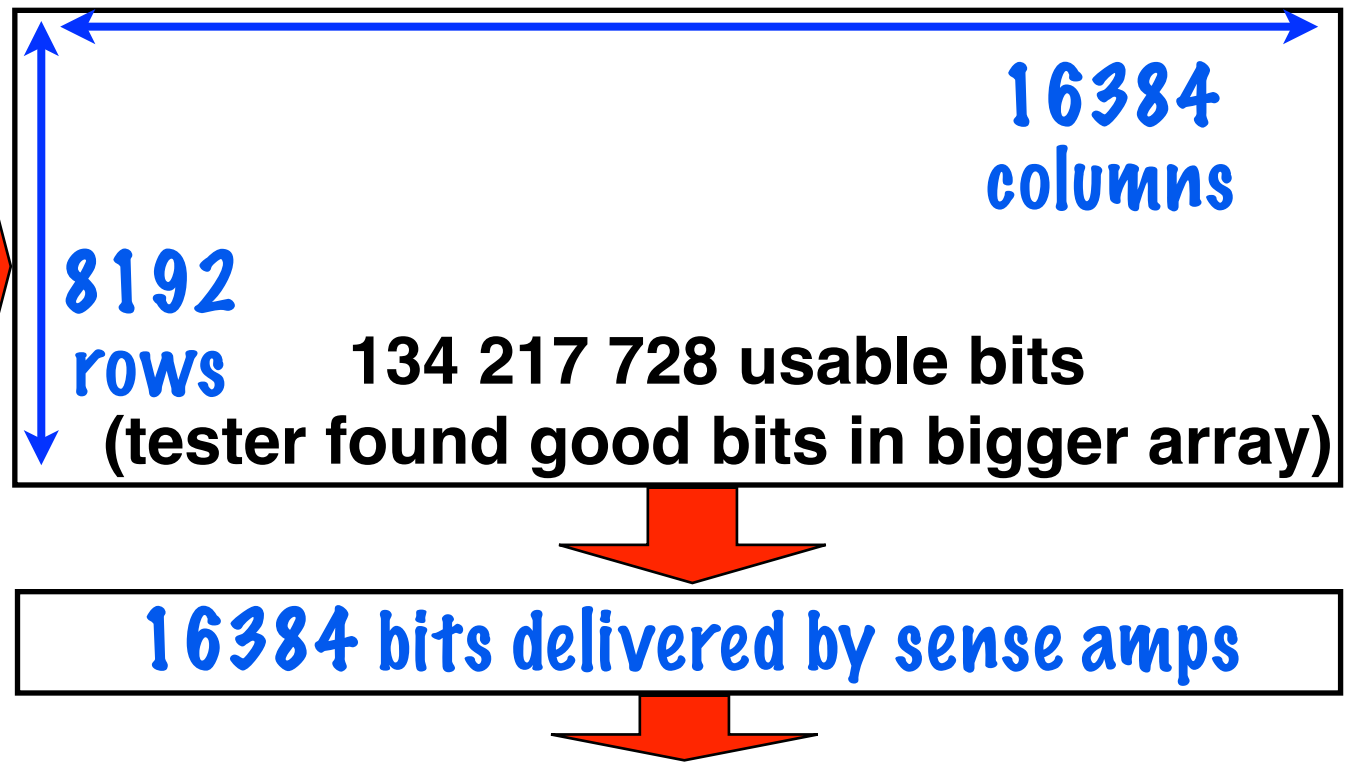
In row access time (55 ns) we can do 22 transfers at 400 MT/s.

16-bit chip bus -> 22 x 16 = 352 bits << 16384

Now the row access time looks fast!

13-bit row address input





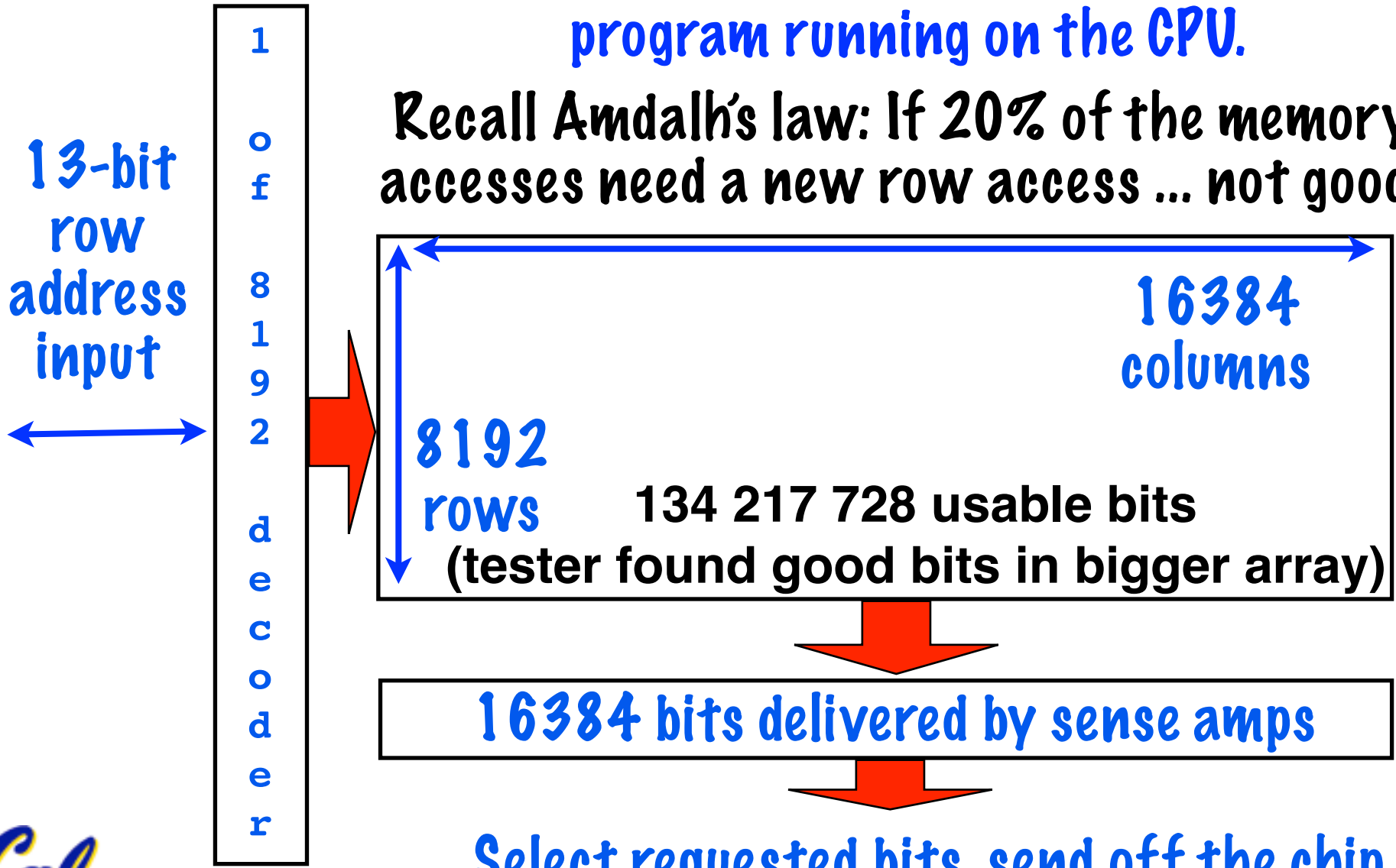
Select requested bits, send off the chip

Sadly, it's rarely this good ...

What if we want all of the 16384 bits?

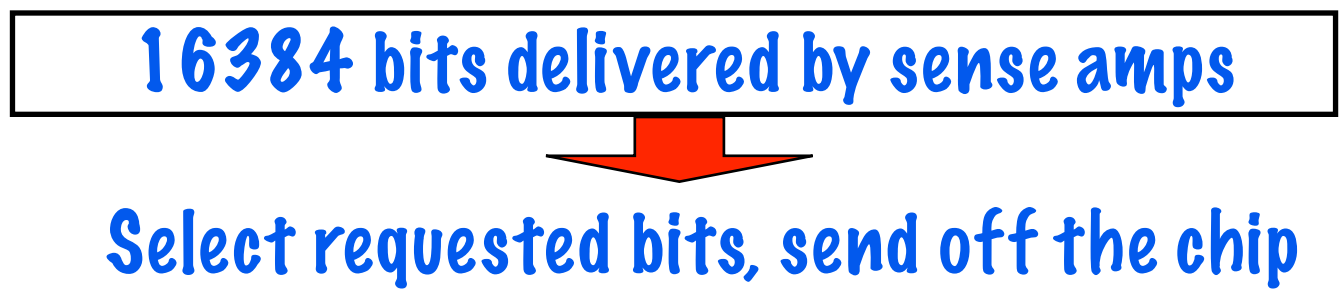
The "we" for a CPU would be the program running on the CPU.

Recall Amdalh's law: If 20% of the memory accesses need a new row access ... not good.

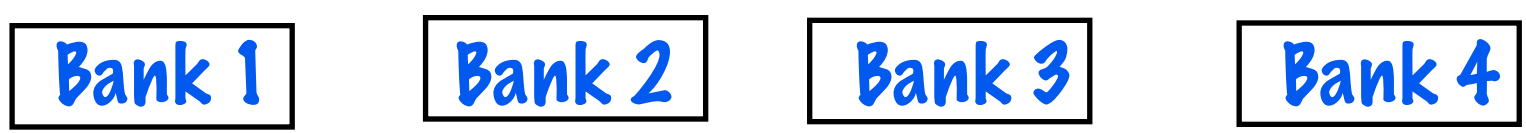


DRAM latency/bandwidth chip features

- **Columns:** Design the right interface for CPUs to request the subset of a column of data it wishes:



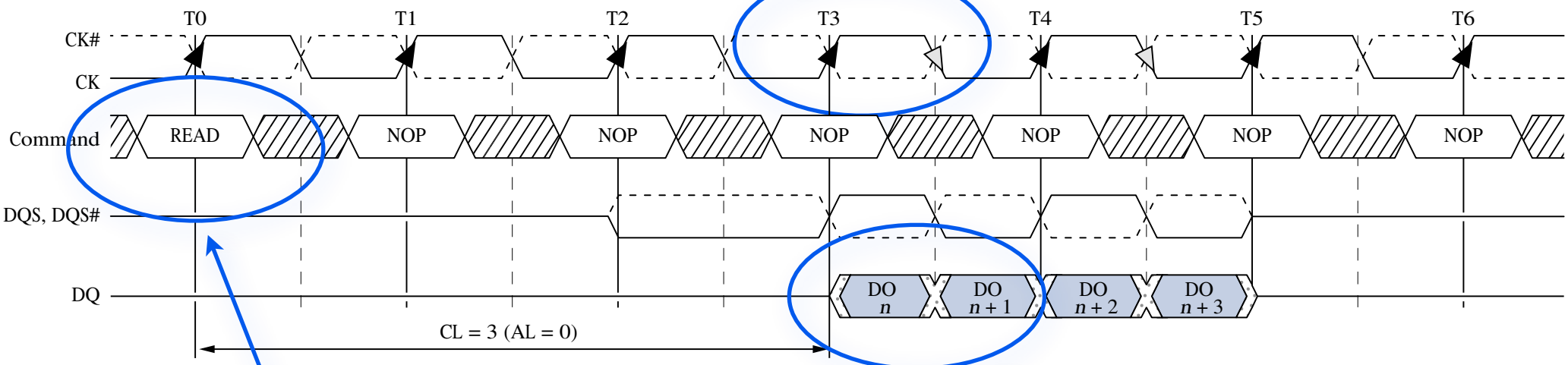
- **Interleaving:** Design the right interface to the 4 memory banks on the chip, so several row requests run in parallel.



Off-chip interface for the Micron part ...

A clocked bus: 200 MHz clock, data transfers on both edges (DDR).

Note! This example is best-case! To access a new row, a slow ACTIVE command must run before the READ.

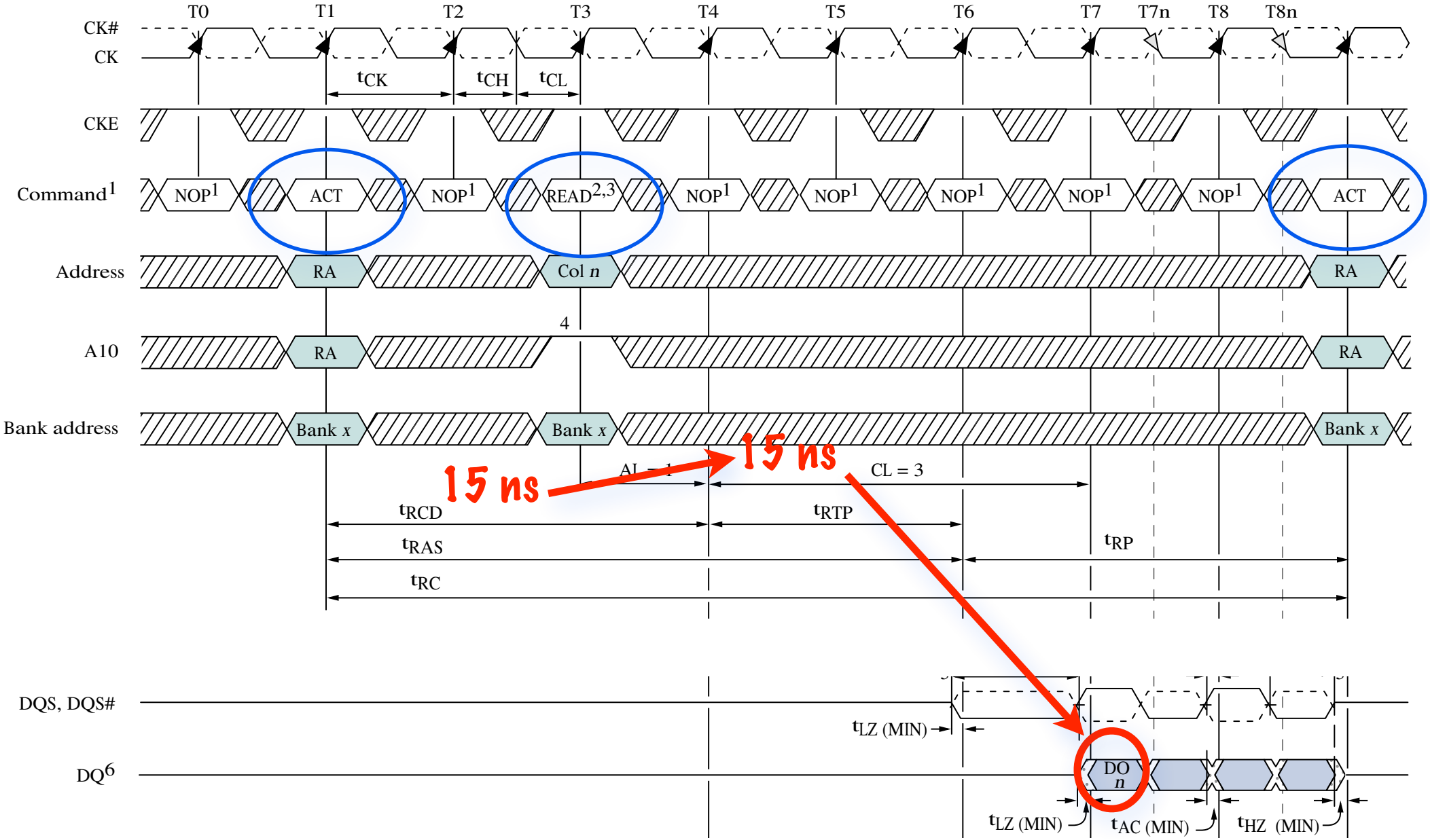


DRAM is controlled via commands (READ, WRITE, REFRESH, ...)

Synchronous data output.

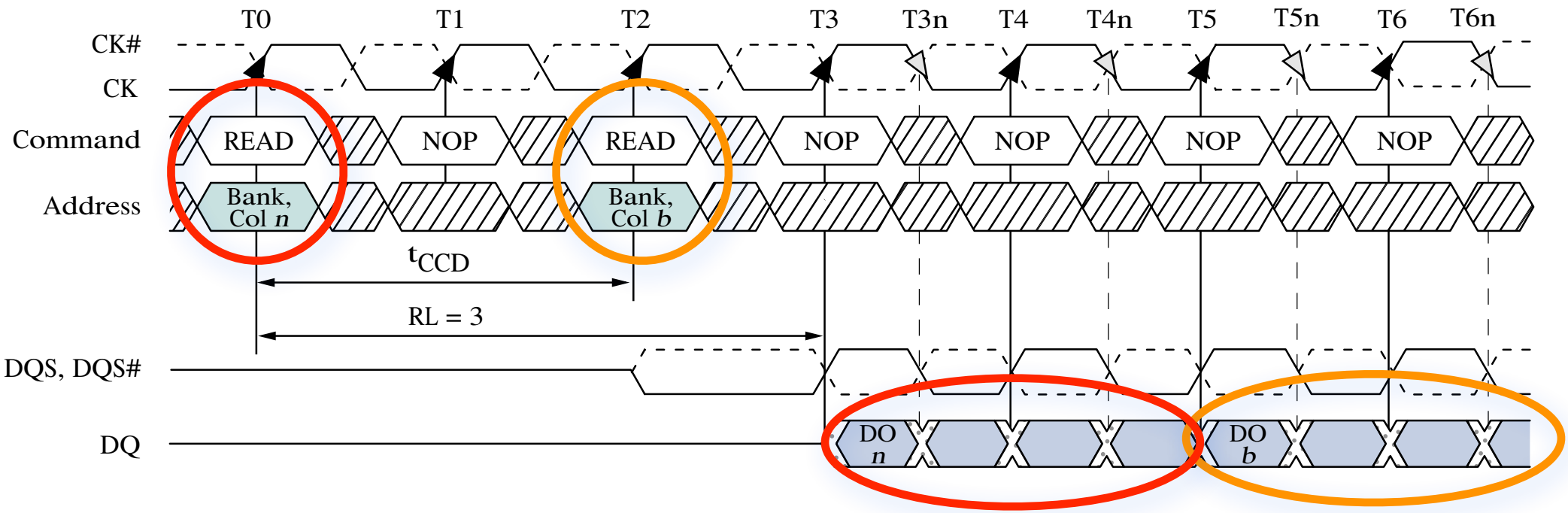
Opening a row before reading ...

Auto-Precharge READ



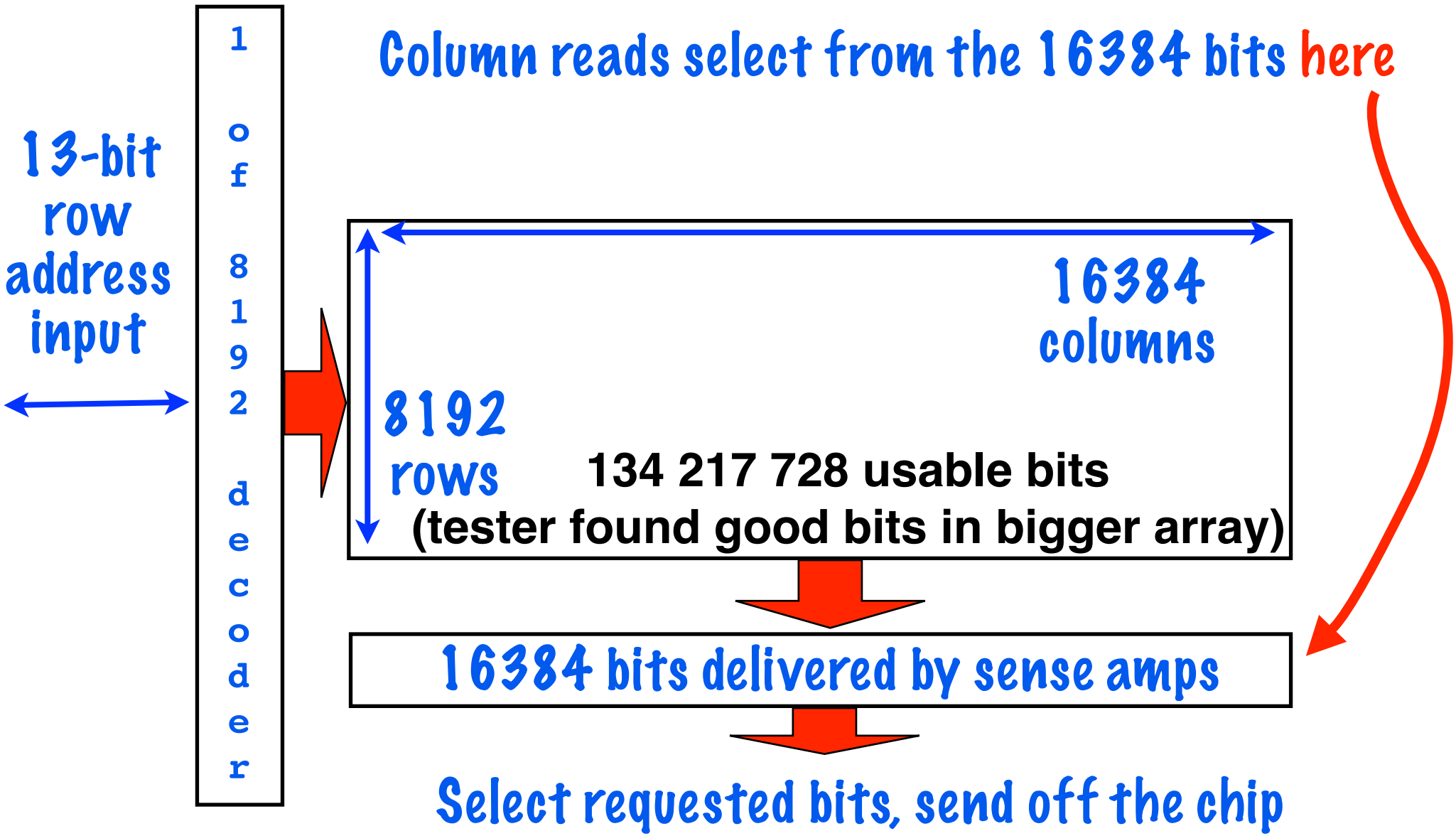
55 ns between row opens.

However, we can read columns quickly

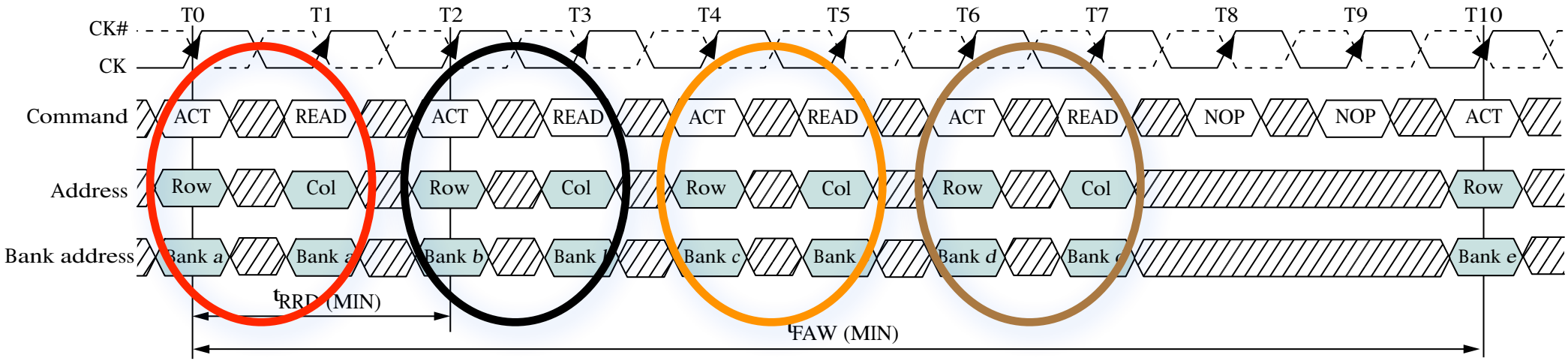


Note: This is a "normal read" (not Auto-Precharge).
Both READs are to the same bank, but different columns.

Why can we read columns quickly?



Interleave: Access all 4 banks in parallel

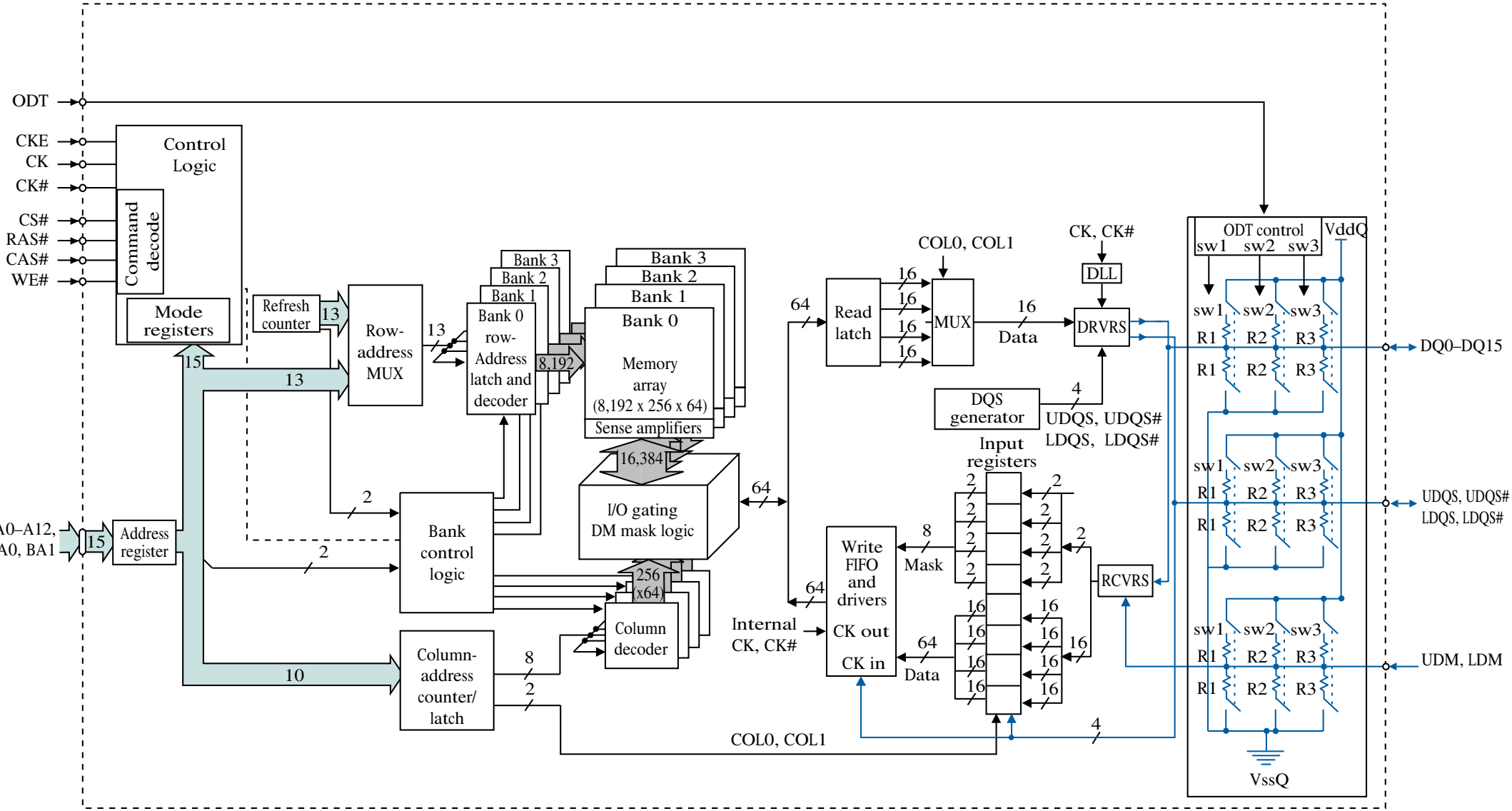


Interleaving: Design the right interface to the 4 memory banks on the chip, so several row requests run in parallel.

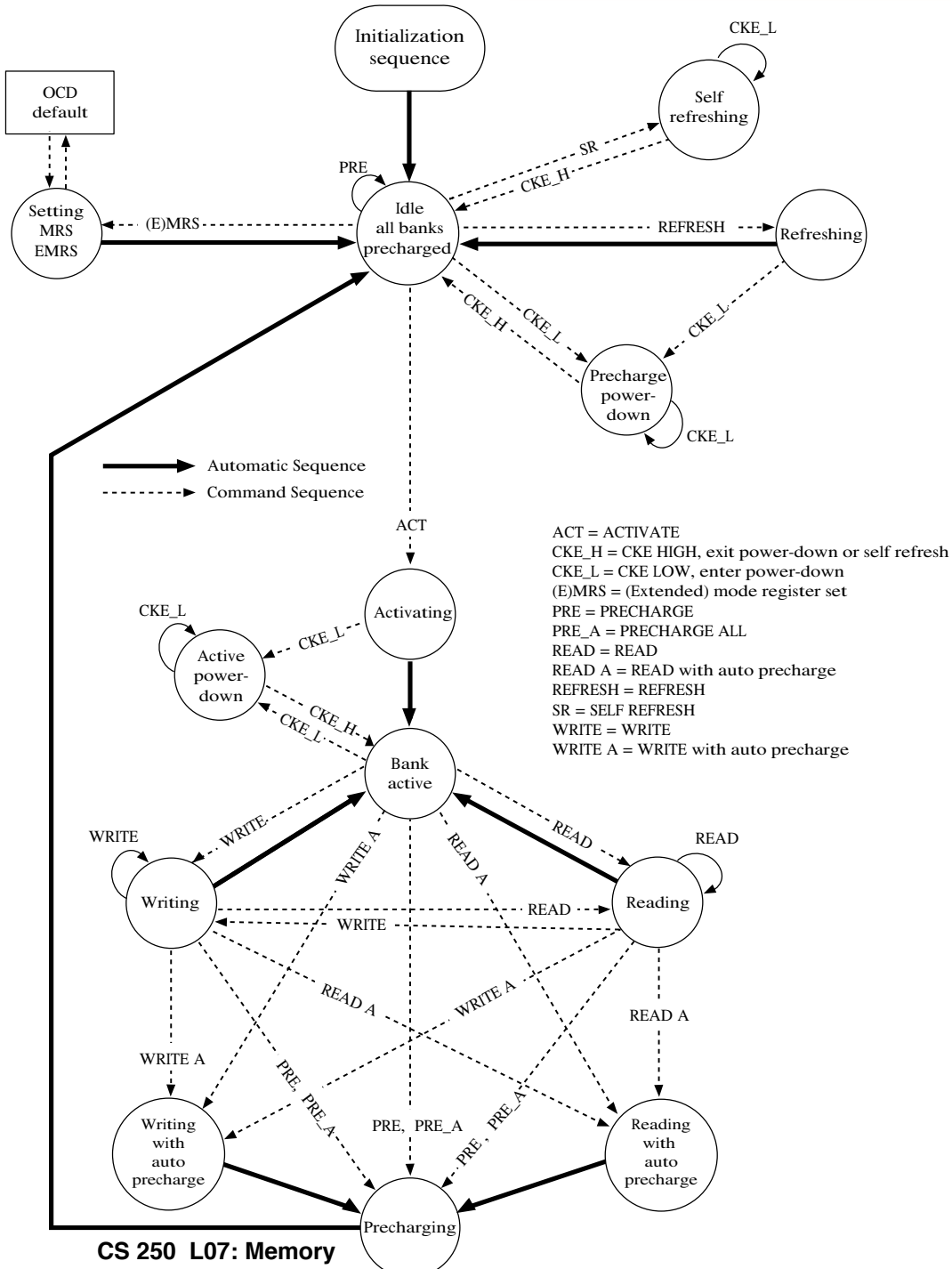
Bank a | Bank b | Bank c | Bank d

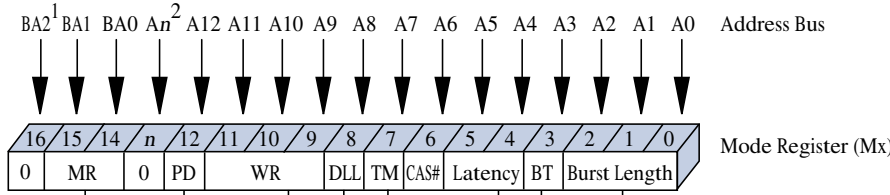
Can also do other commands on banks concurrently.

Only part of a bigger story ...



Only part of a bigger story ...





M12	PD Mode
0	Fast exit (normal)
1	Slow exit (low power)

M7	Mode
0	Normal
1	Test

M8	DLL Reset
0	No
1	Yes

M2	M1	M0	Burst Length
0	0	0	Reserved
0	0	1	Reserved
0	1	0	4
0	1	1	8
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

M11	M10	M9	Write Recovery
0	0	0	Reserved
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

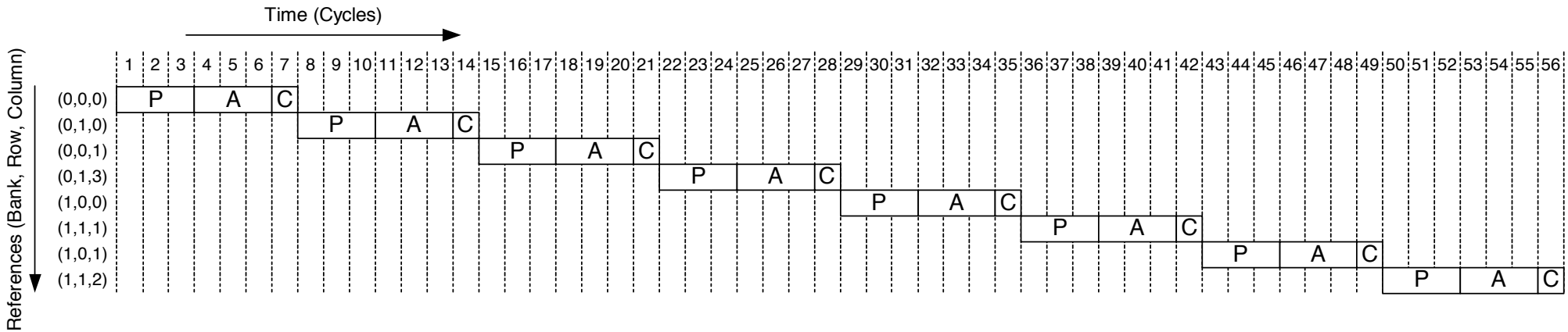
M3	Burst Type
0	Sequential
1	Interleaved

M15	M14	Mode Register Definition
0	0	Mode register (MR)
0	1	Extended mode register (EMR)
1	0	Extended mode register (EMR2)
1	1	Extended mode register (EMR3)

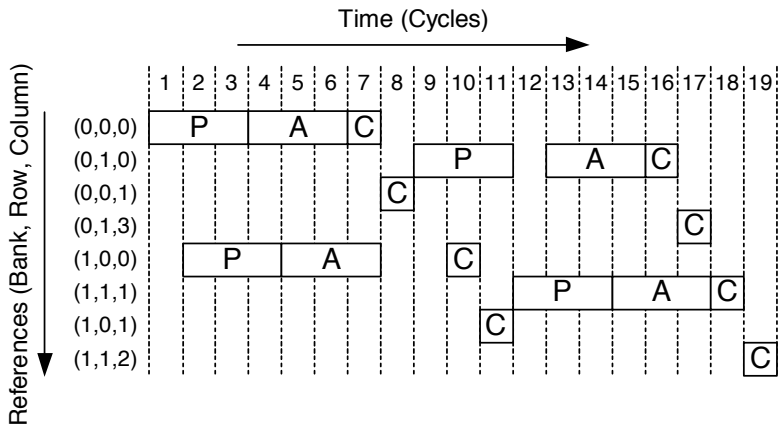
M6	M5	M4	CAS Latency (CL)
0	0	0	Reserved
0	0	1	Reserved
0	1	0	Reserved
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

DRAM controllers: reorder requests

(A) Without access scheduling (56 DRAM Cycles)



(B) With access scheduling (19 DRAM Cycles)



DRAM Operations:

P: bank precharge (3 cycle occupancy)
A: row activation (3 cycle occupancy)
C: column access (1 cycle occupancy)

From: **Memory Access Scheduling**

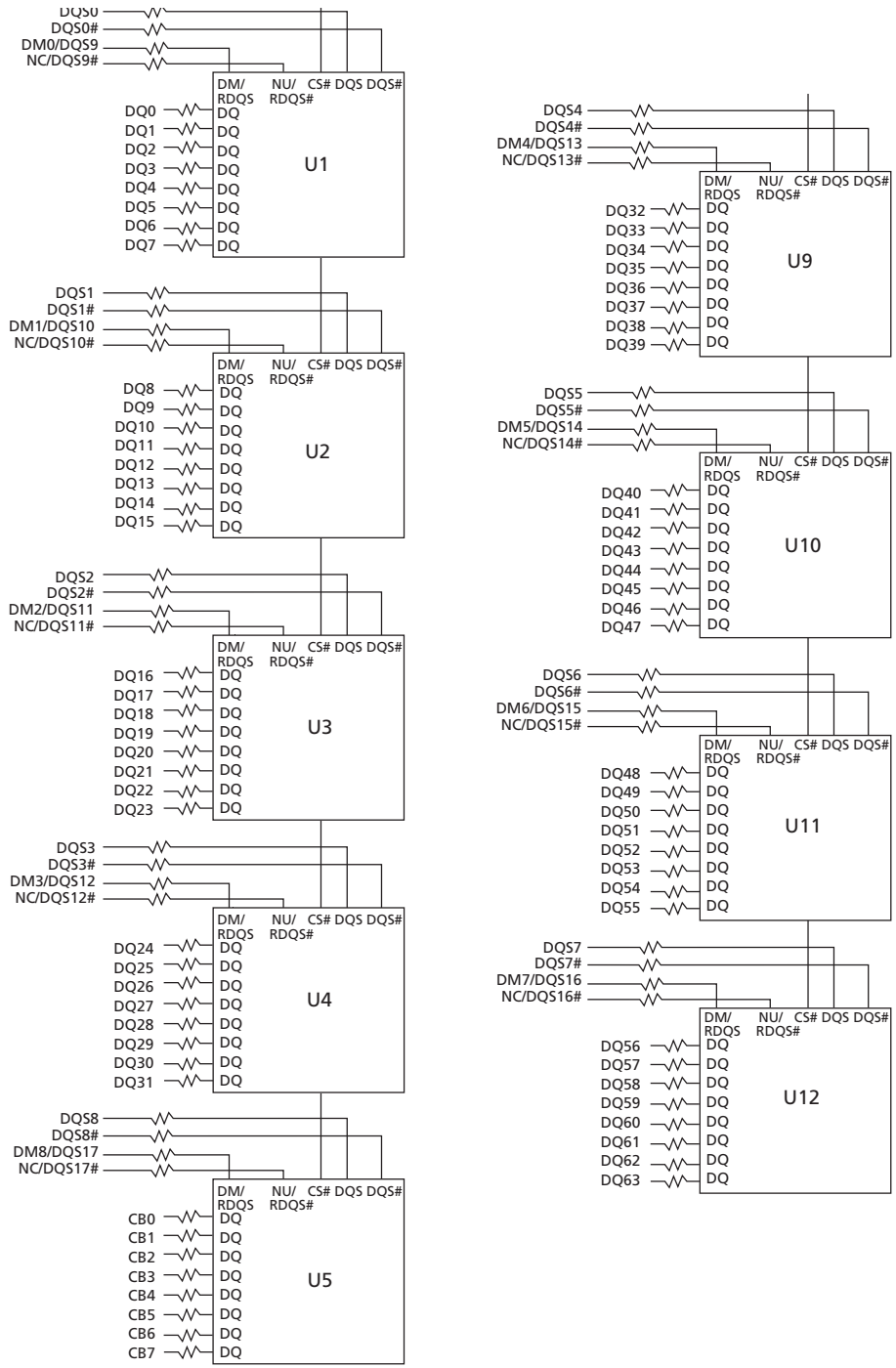
Scott Rixner[1], William J. Dally, Ujval J. Kapasi, Peter Mattson, and John D. Owens

Memory Packaging

From DRAM chip to DIMM module ...

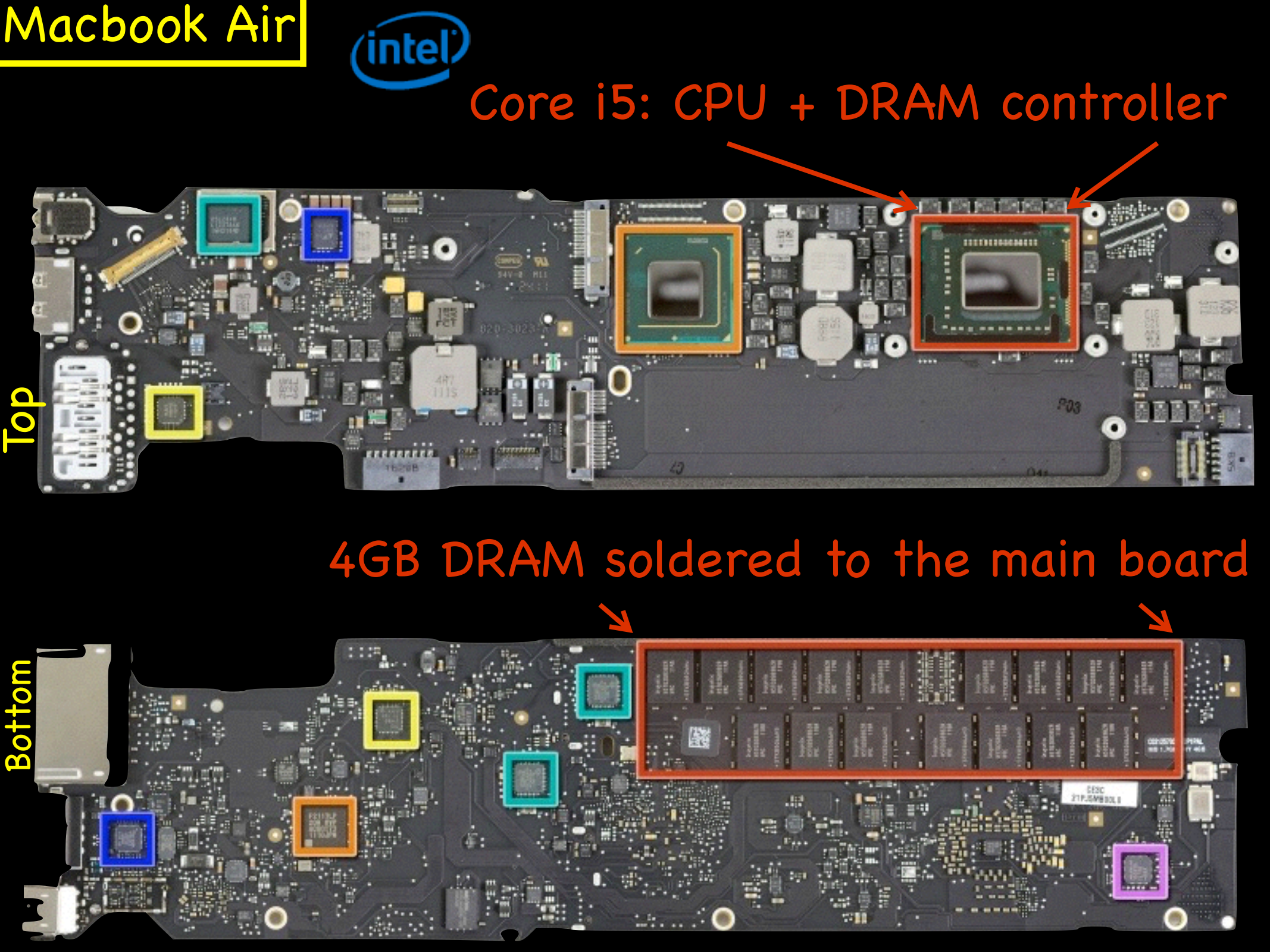
Each RAM chip responsible for 8 lines of the 64 bit data bus (U5 holds the check bits).

Commands sent to all 9 chips, qualified by per-chip select lines.



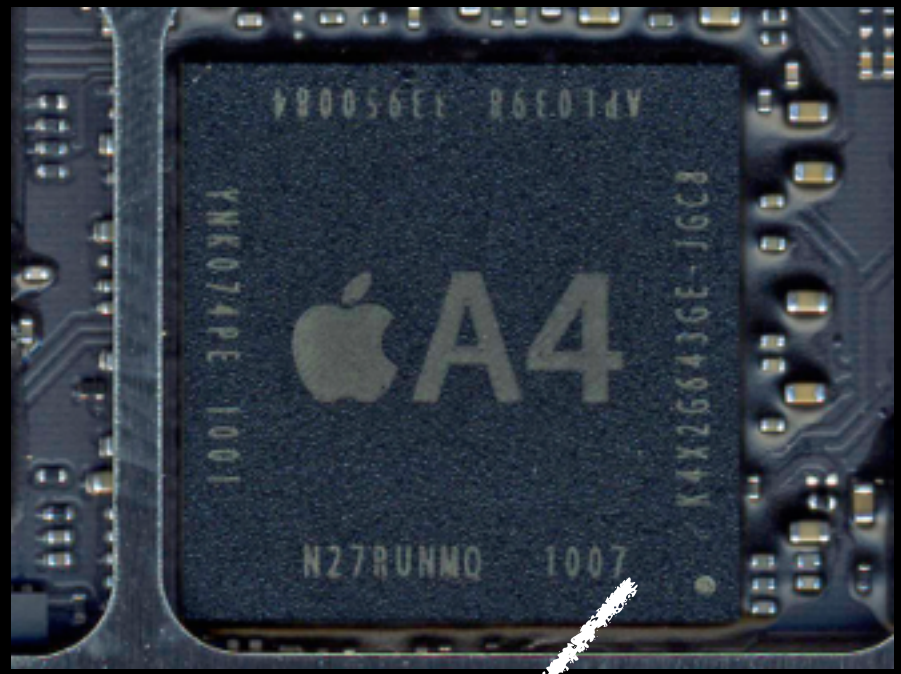
MacBook Air … too thin to use DIMMs

Macbook Air


Core i5: CPU + DRAM controller
Top
4GB DRAM soldered to the main board
Bottom

Original iPad (2010)

"Package-in-Package"



128MB SDRAM dies (2)

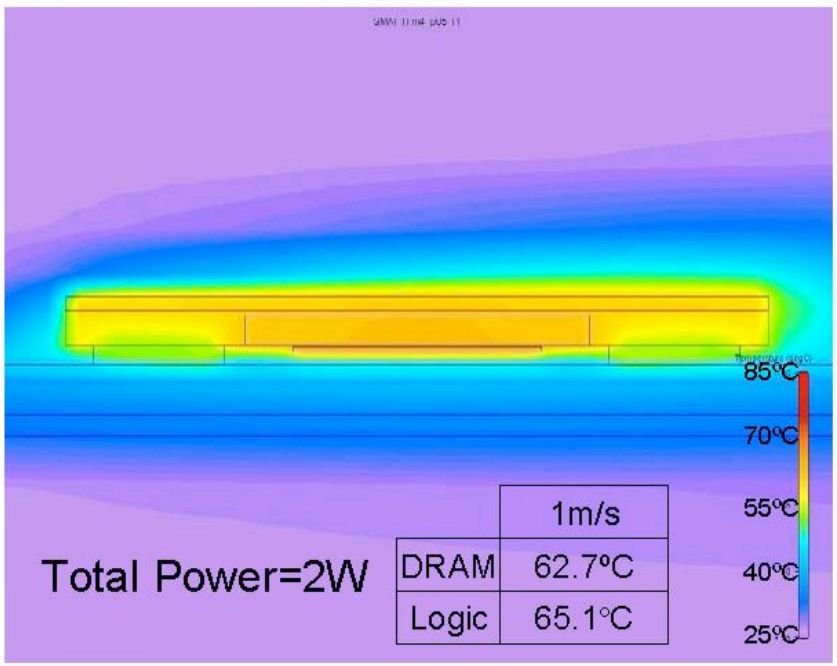
Cut-away side view

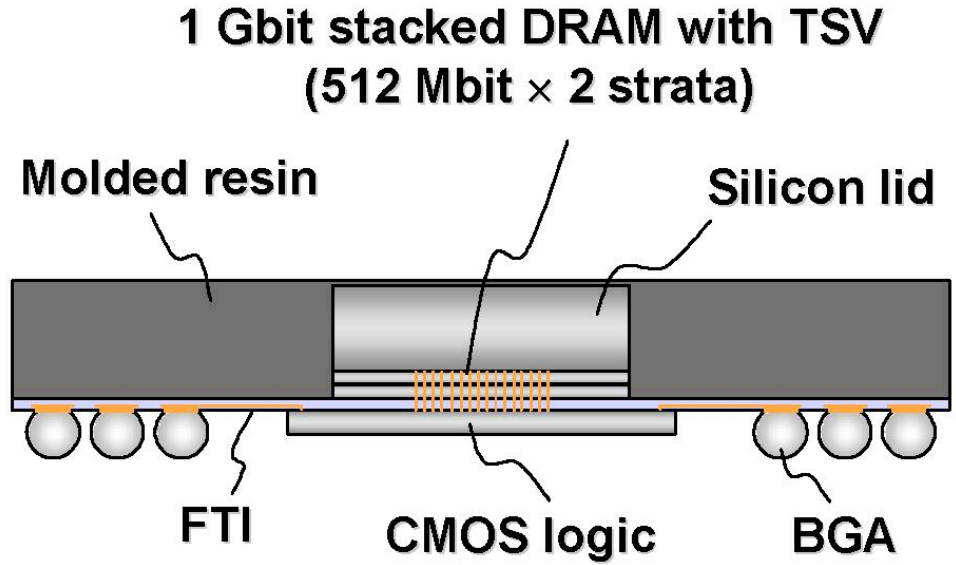
Apple A4 SoC

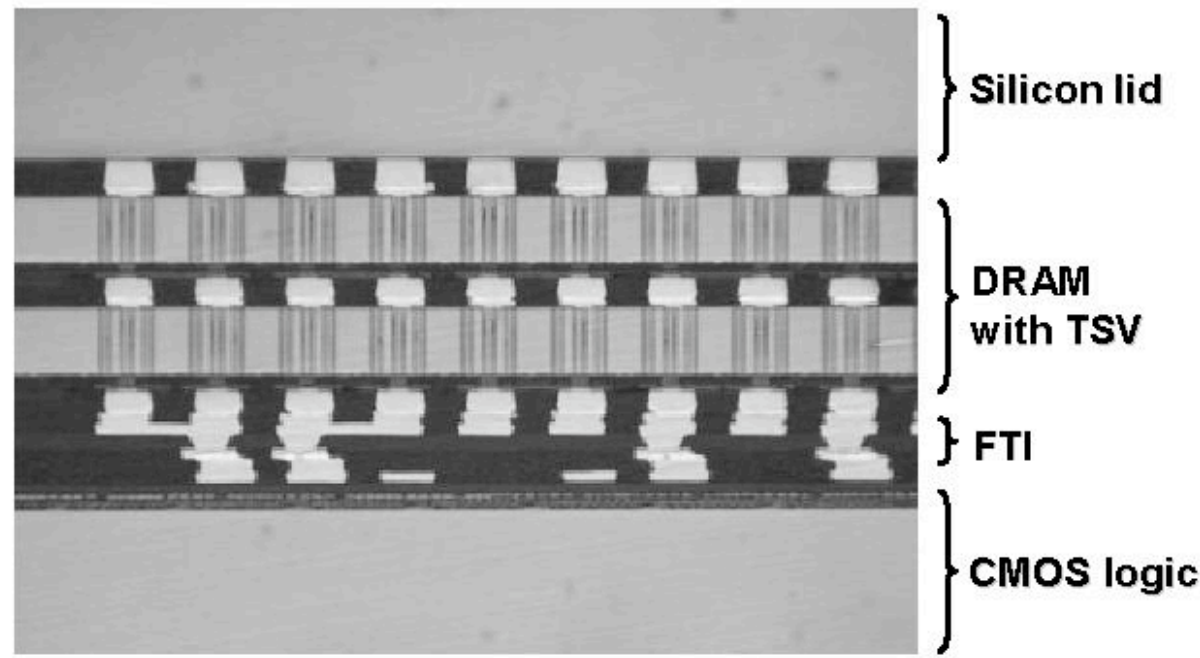
Dies connect using bond wires and solder balls ...

3-D memory stack

DRAM die size	10.7 mm × 13.3 mm
DRAM die thickness	50 μm
TSV count in DRAM	1,560
DRAM capacity	512 Mbit/die × 2 strata
CMOS logic die size	17.5 mm × 17.5 mm
CMOS logic die thickness	200 μm
CMOS logic bump count	3,497
CMOS logic process	0.18 μm CMOS
DRAM-logic FTI via pitch	50 μm
Package size	33 mm × 33 mm
BGA terminal	520 pin / 1mm pitch







A 3D Stacked Memory Integrated on a Logic Device Using SMAFTI Technology

Yoichiro Kurita[1], Satoshi Matsui[1], Nobuaki Takahashi[1], Koji Soejima[1], Masahiro Komuro[1], Makoto Itou[1], Chika Kakegawa[1], Masaya Kawano[1], Yoshimi Egawa[2], Yoshihiro Saeki[2], Hidekazu Kikuchi[2], Osamu Kato[2], Azusa Yanagisawa[2], Toshiro Mitsuhashi[2], Masakazu Ishino[3], Kayoko Shibata[3], Shiro Uchiyama[3], Junji Yamada[3], and Hiroaki Ikeda[3]
[1]NEC Electronics, [2]Oki Electric Industry, and [3]Elpida Memory
1120 Shimokuzawa, Sagamihara, Kanagawa 229-1198, Japan
y.kurita@necel.com

Break

Play:

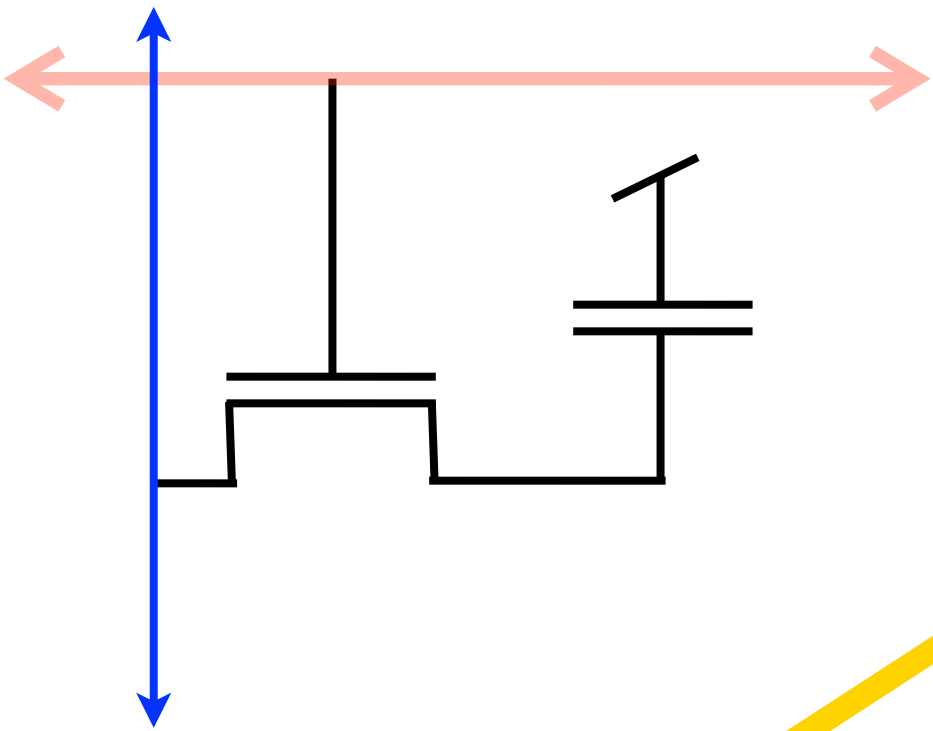
Static Memory Circuits

Dynamic Memory: Circuit remembers for a fraction of a second.

Static Memory: Circuit remembers as long as the power is on.

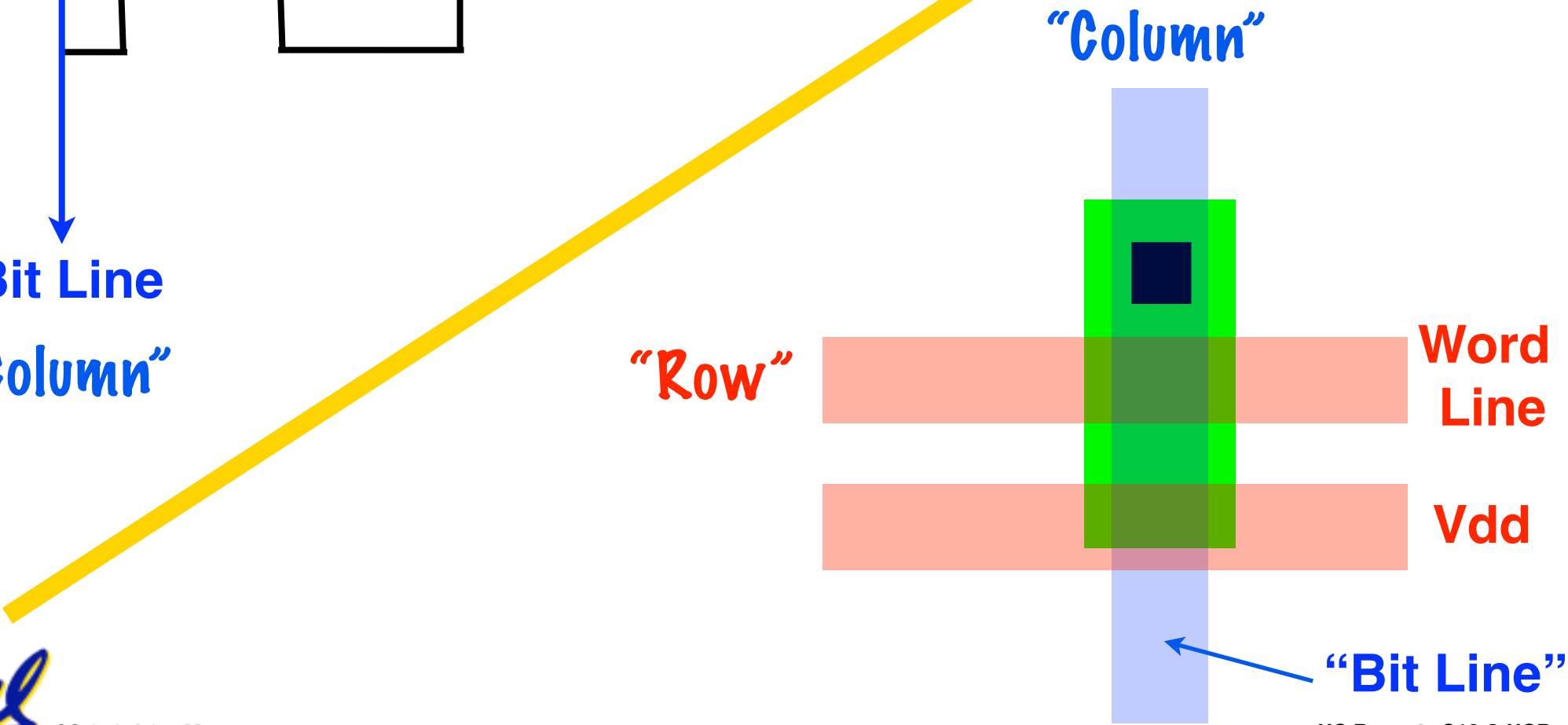
Non-volatile Memory: Circuit remembers for many years, even if power is off.

Recall DRAM cell: 1 T + 1 C

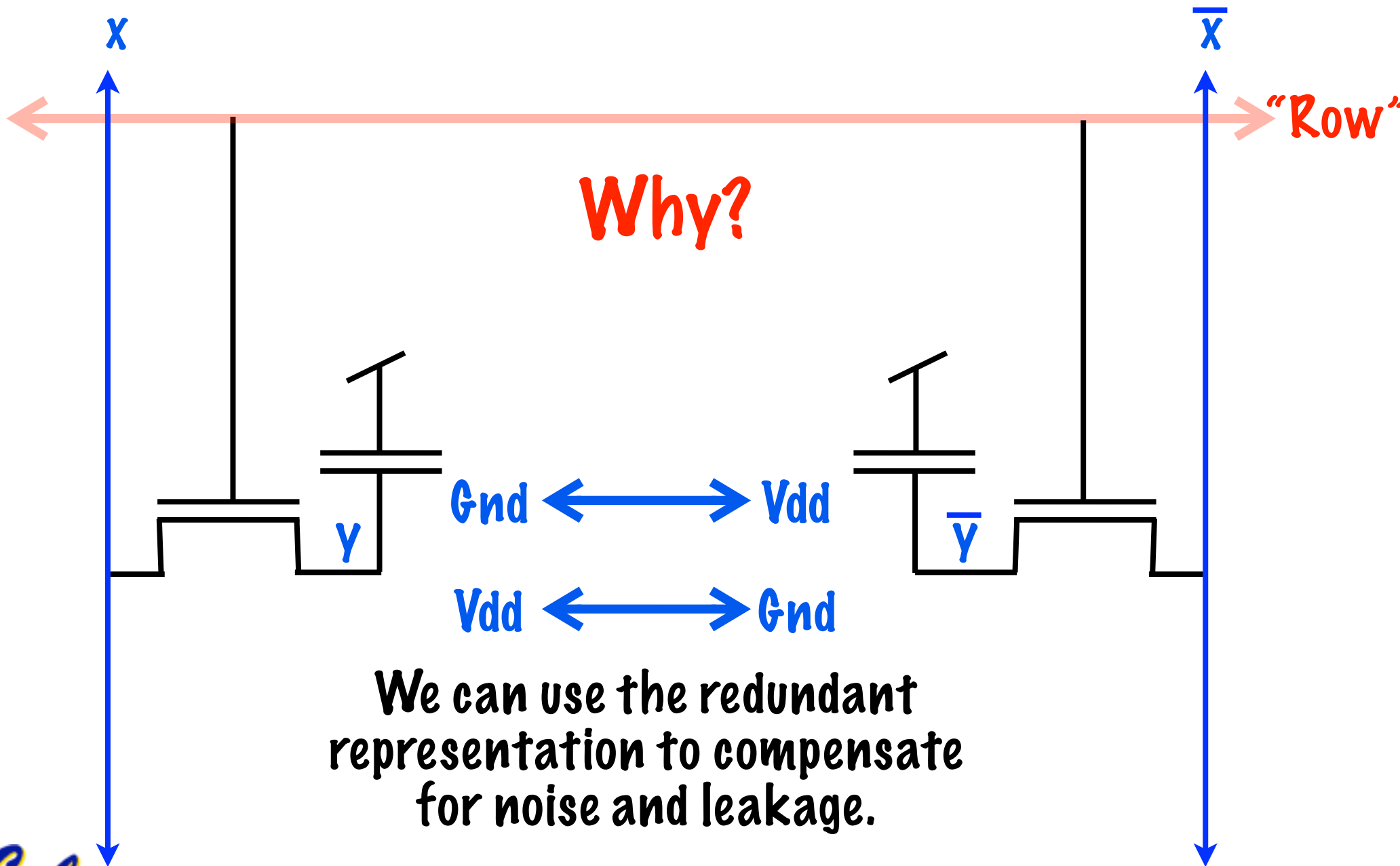


"Word Line"
"Row"

Bit Line
"Column"

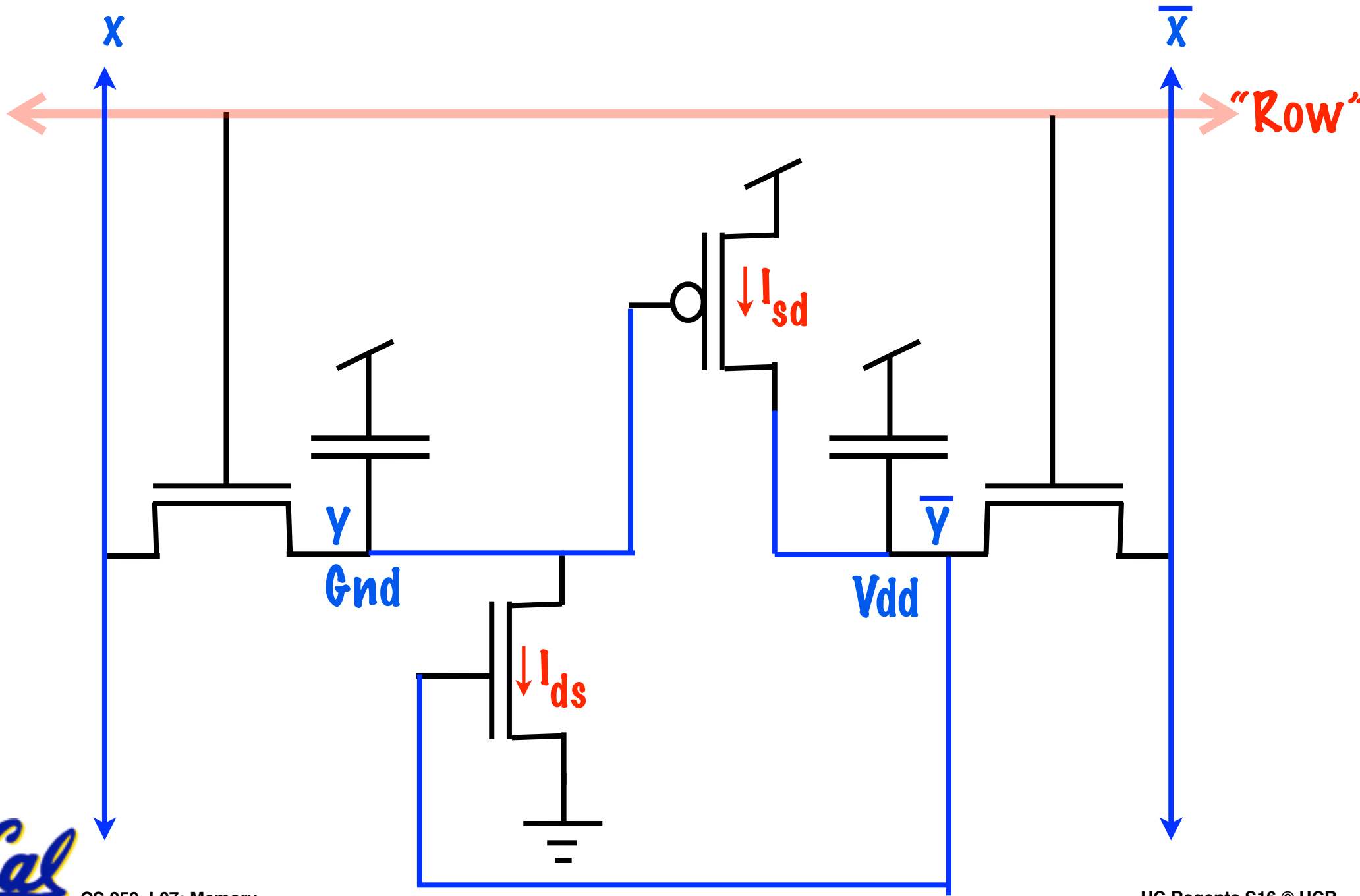


Idea: Store each bit with its complement

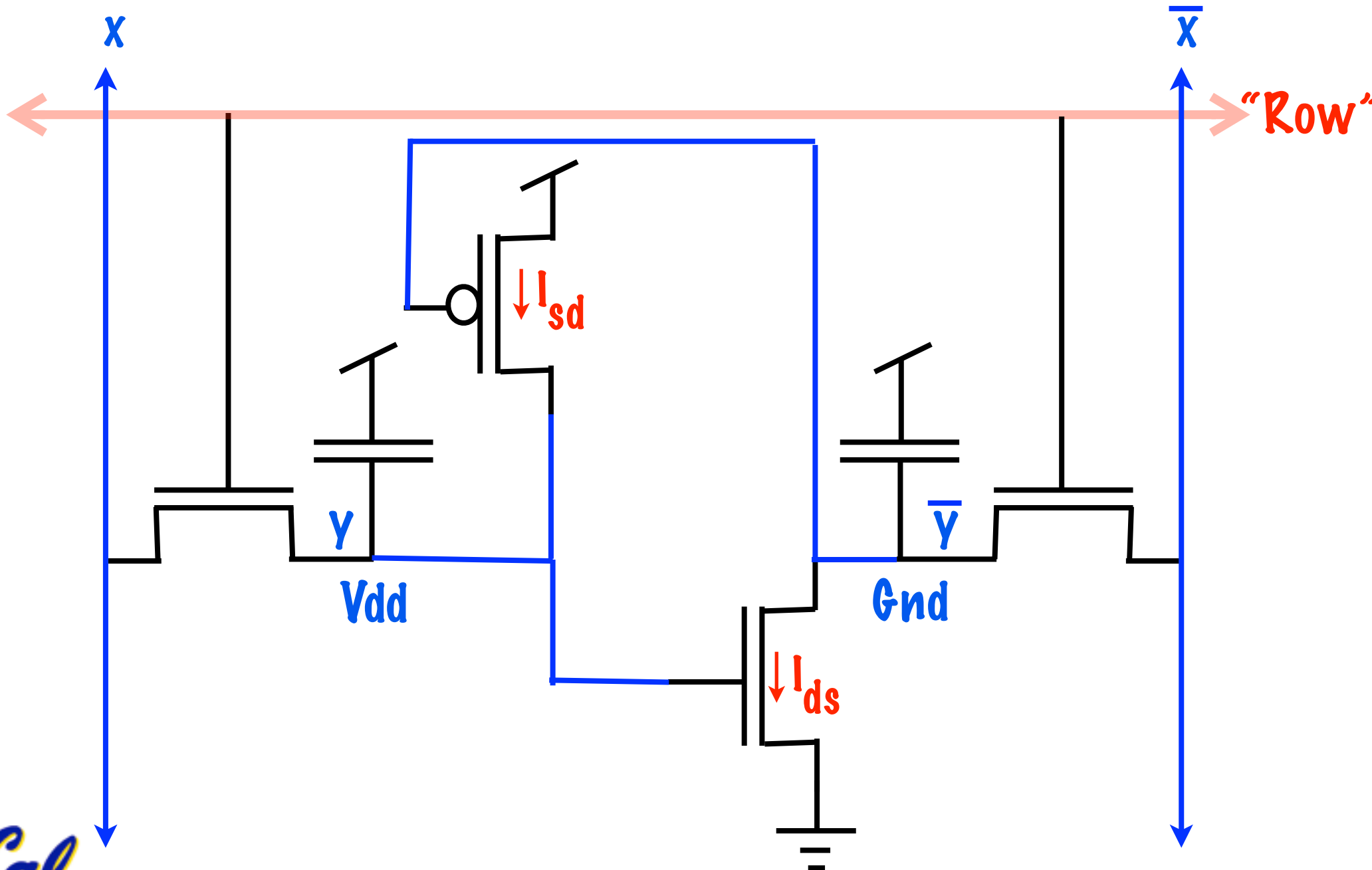


We can use the redundant representation to compensate for noise and leakage.

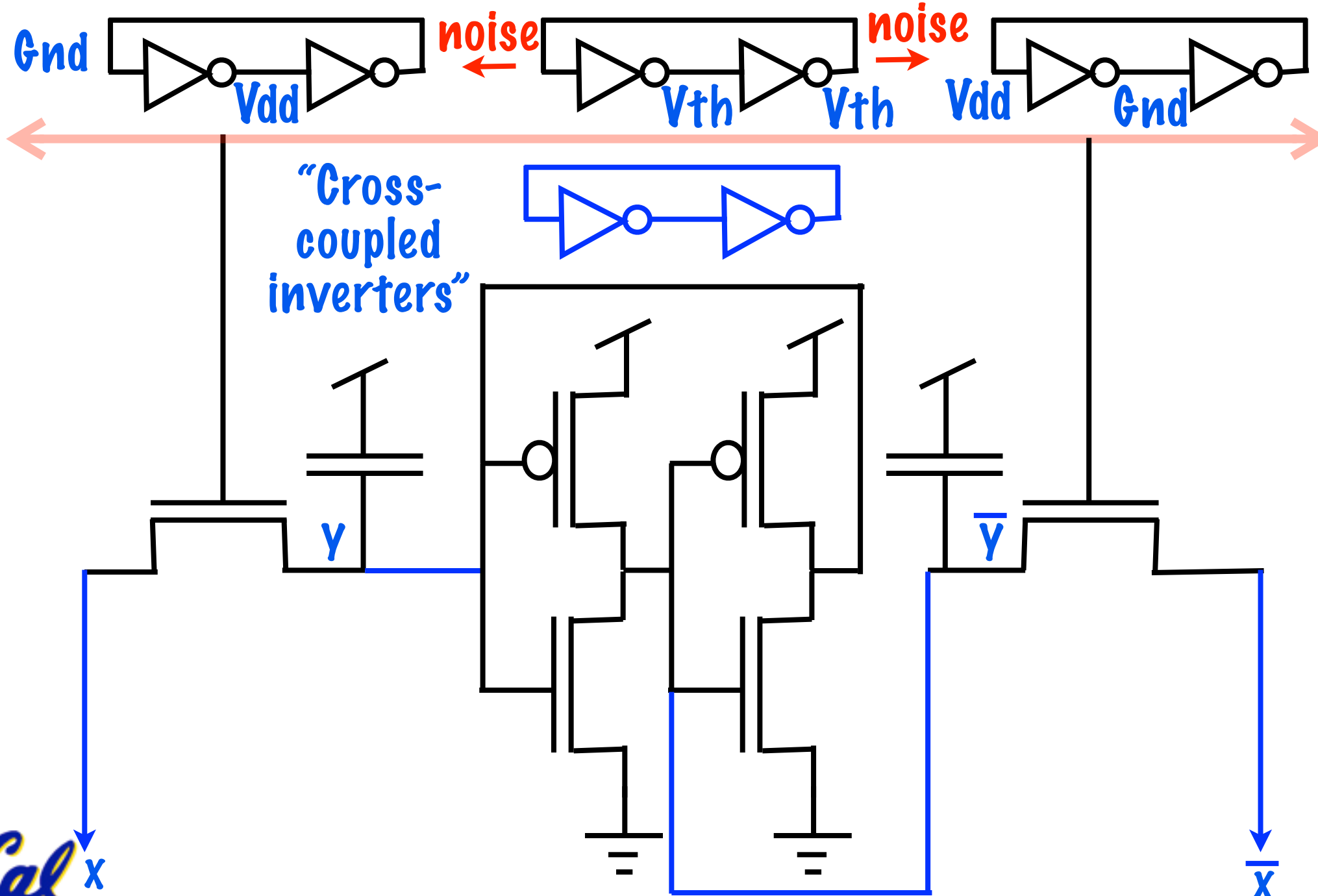
Case #1: y = Gnd, $\overline{y}$ = Vdd ...



Case #2: y = Vdd, $\overline{y}$ = Gnd ...

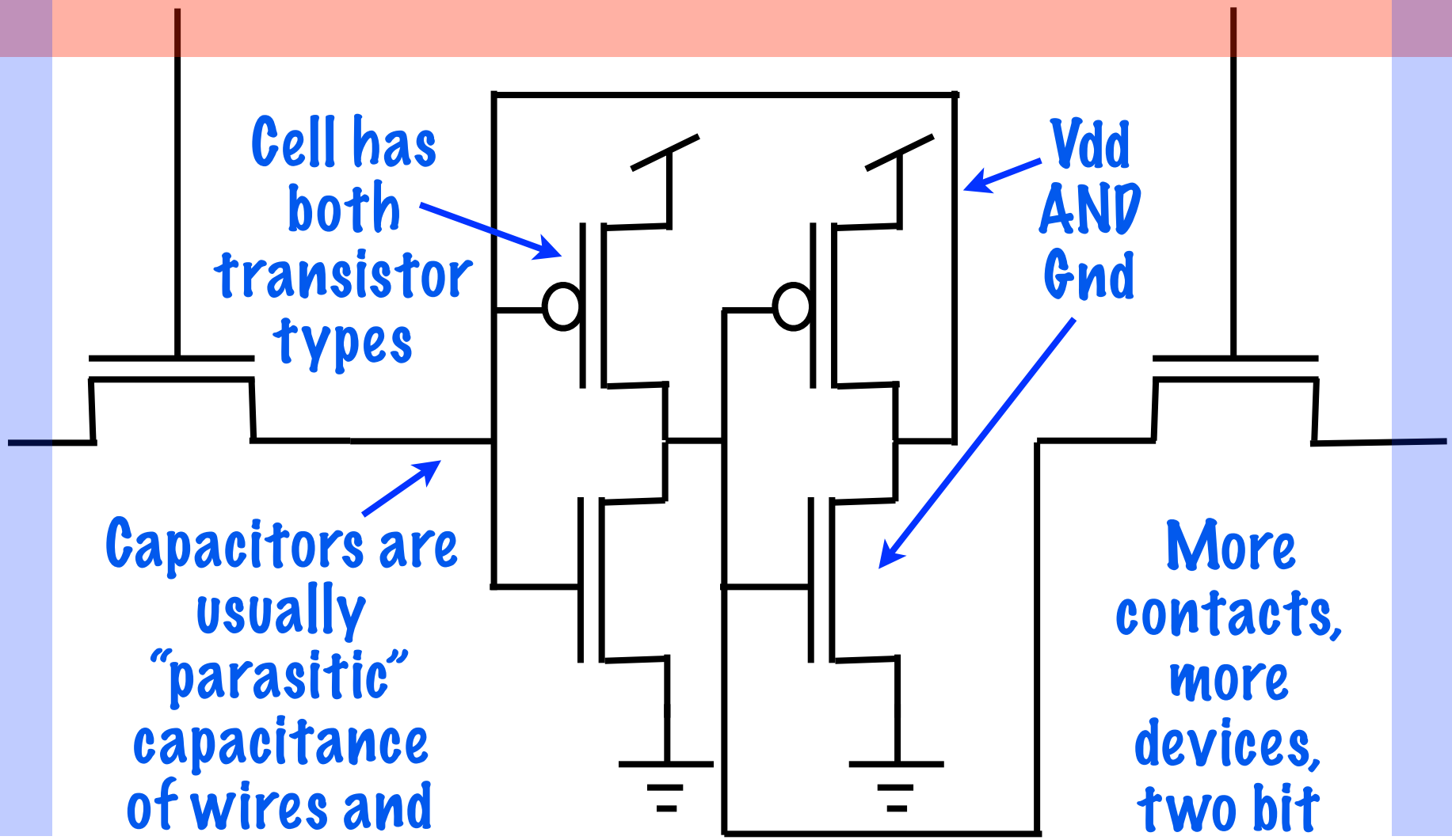


Combine both cases to complete circuit



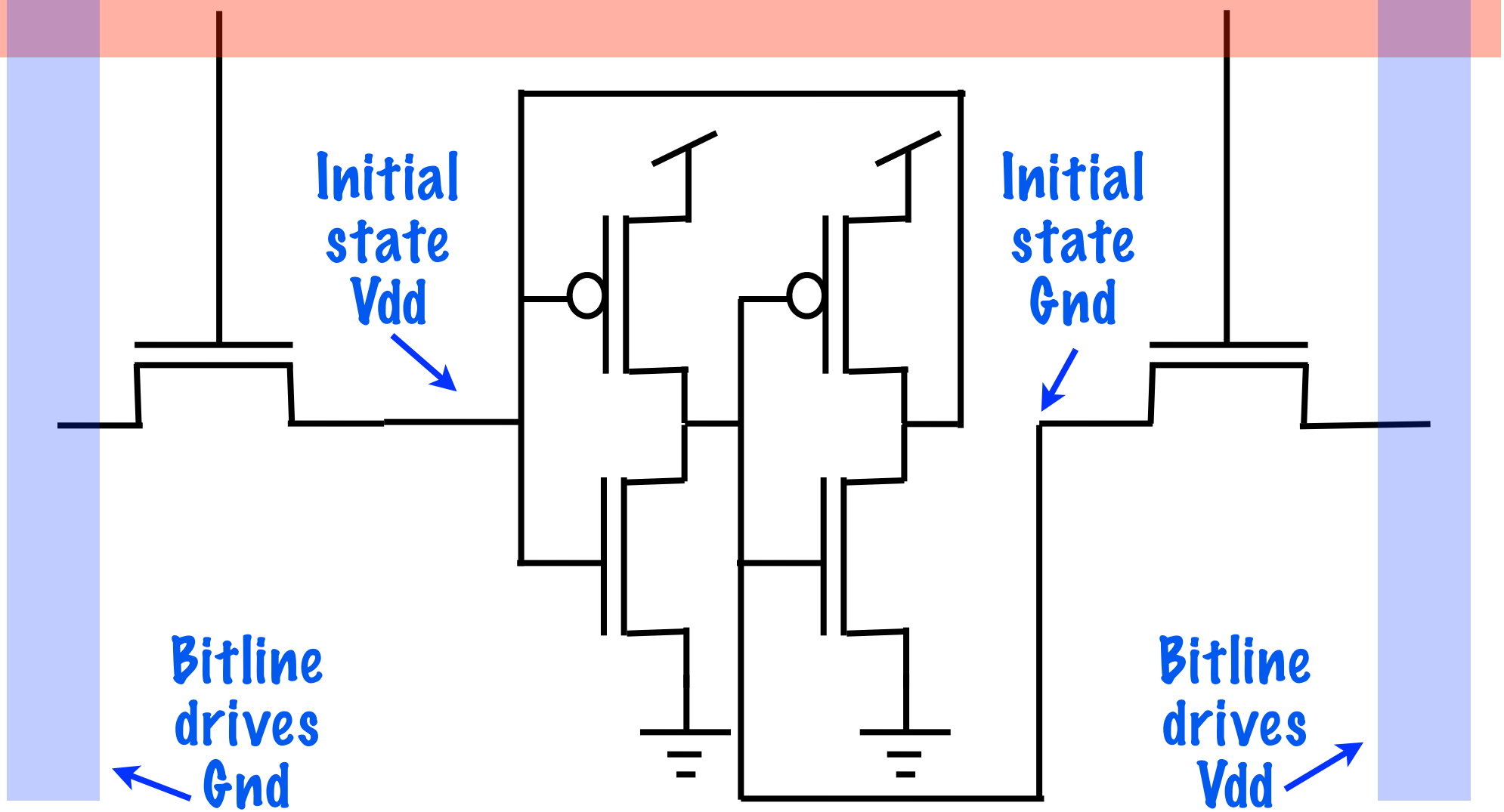
SRAM Challenge #1: It's so big!

SRAM area is 6X-10X DRAM area, same generation ...



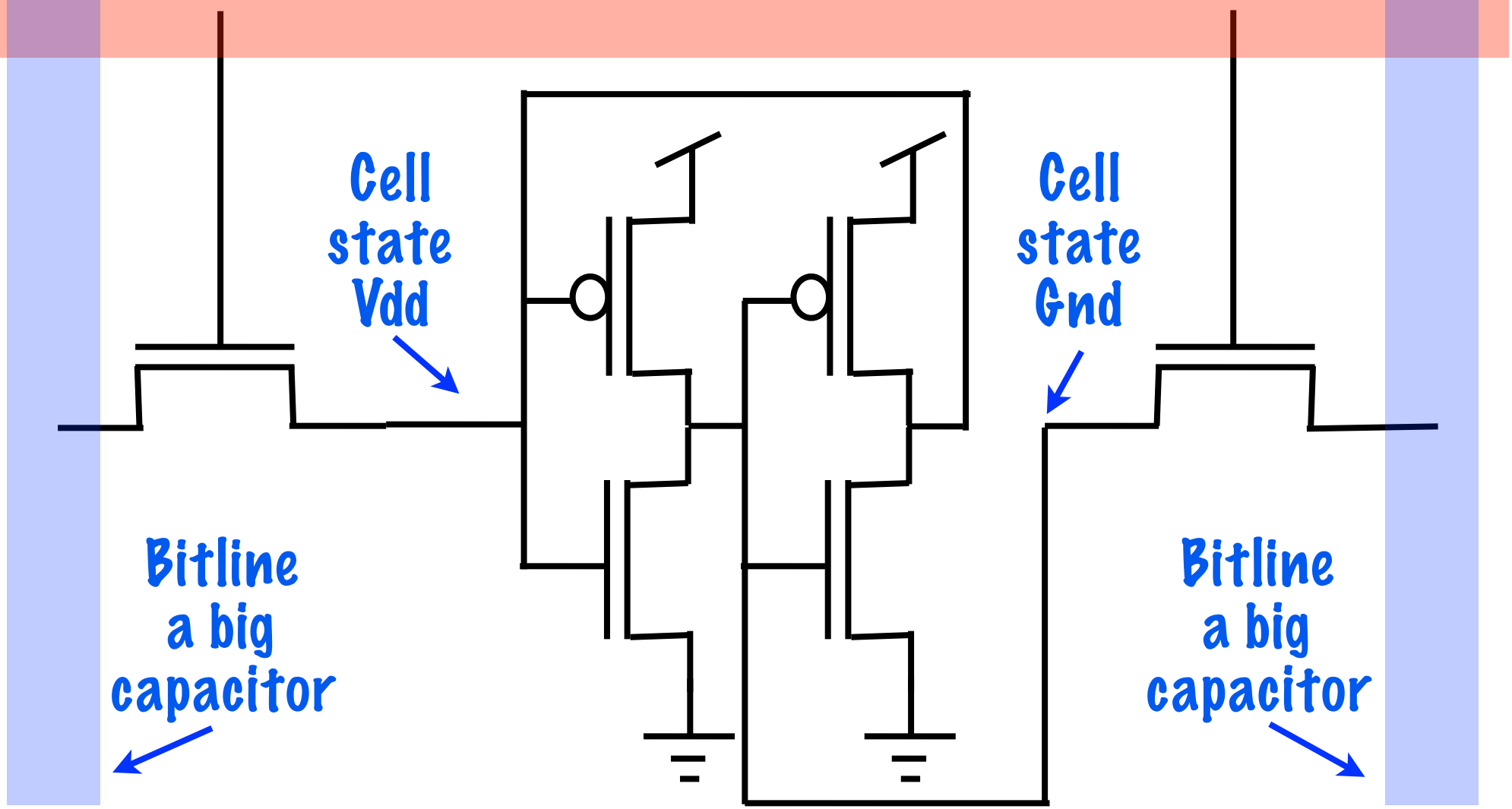
Challenge #2: Writing is a "fight"

When word line goes high, bitlines "fight" with cell inverters to "flip the bit" -- must win quickly!
Solution: tune W/L of cell & driver transistors

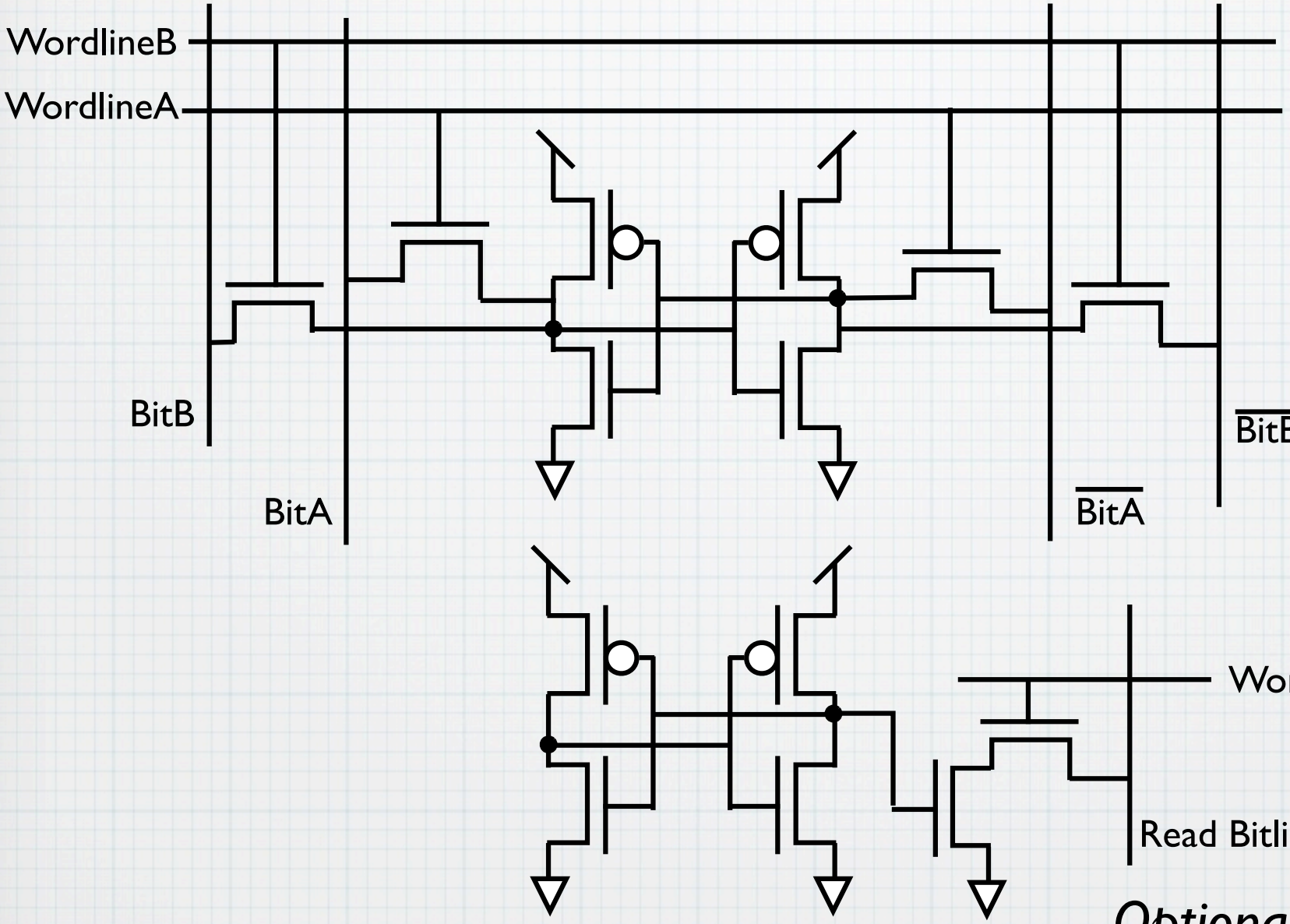


Challenge #3: Preserving state on read

When word line goes high on read, cell inverters must drive large bitline capacitance quickly, to preserve state on its small cell capacitances



Adding More Ports

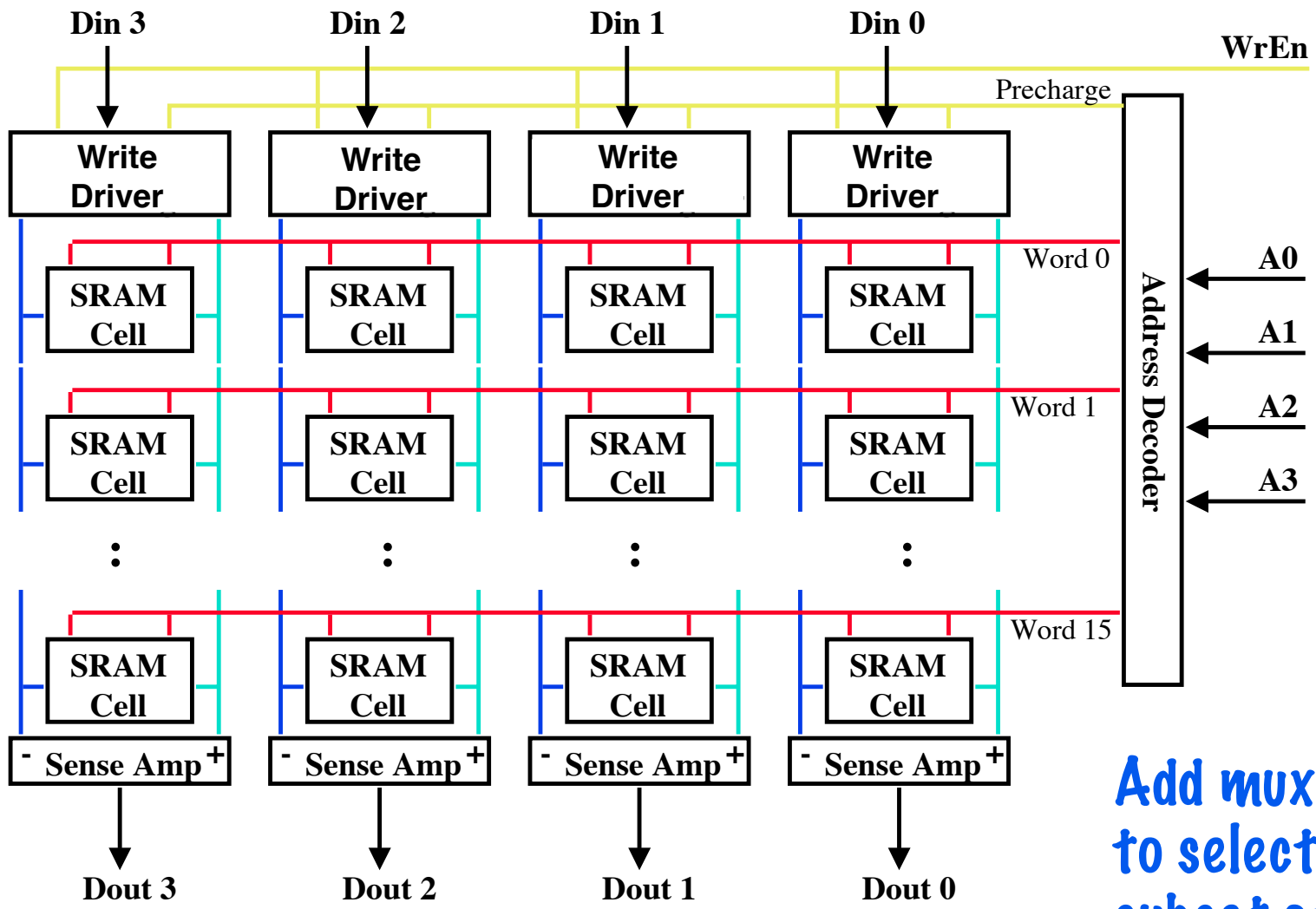


Differential Read or Write ports

Optional Single-ended Read port

SRAM array: like DRAM, but non-destructive

Architects specify number of rows and columns.
Word and bit lines slow down as array grows larger!

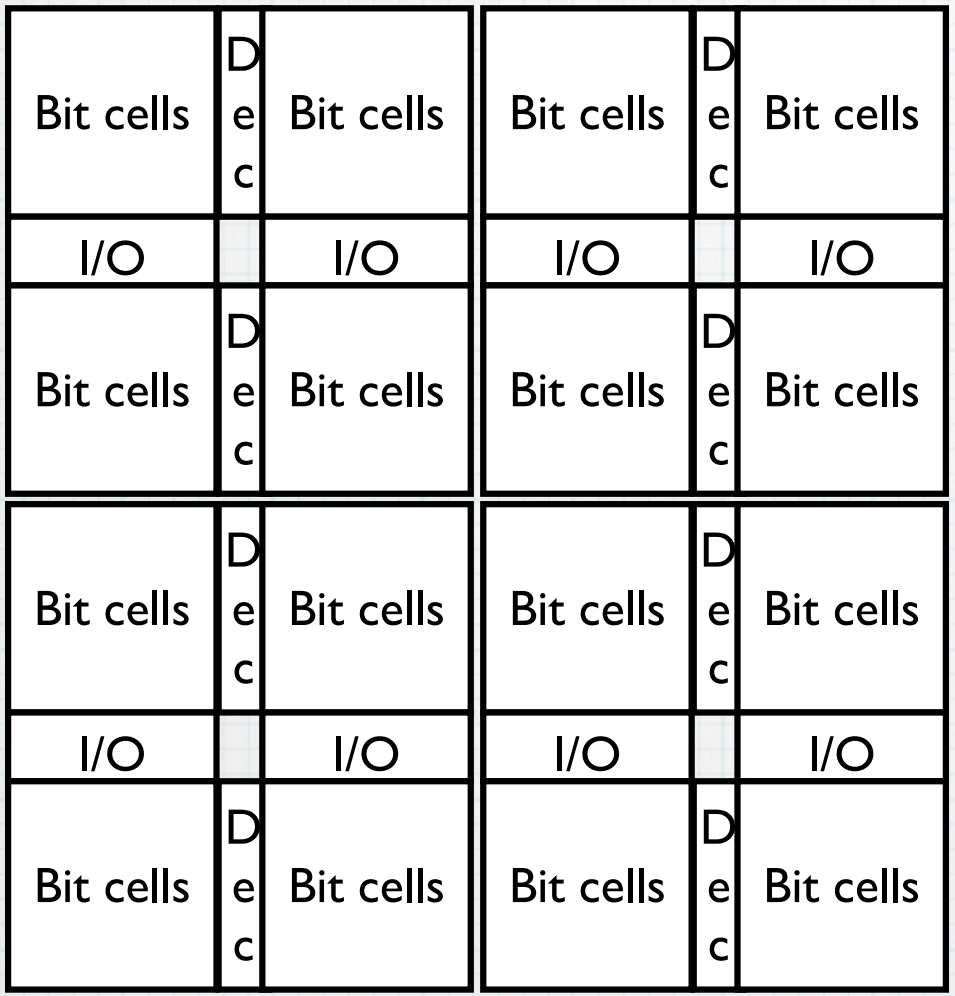


Parallel Data I/O Lines

Add muxes to select subset of bits

How could we pipeline this memory?

Building Larger Memories



- Large arrays constructed by tiling multiple leaf arrays, sharing decoders and I/O circuitry
  - e.g., sense amp attached to arrays above and below
- Leaf array limited in size to 128-256 bits in row/column due to RC delay of wordlines and bitlines
- Also to reduce power by only activating selected sub-bank
- In larger memories, delay and energy dominated by I/O wiring

SRAM vs DRAM, pros and cons

Big win for DRAM

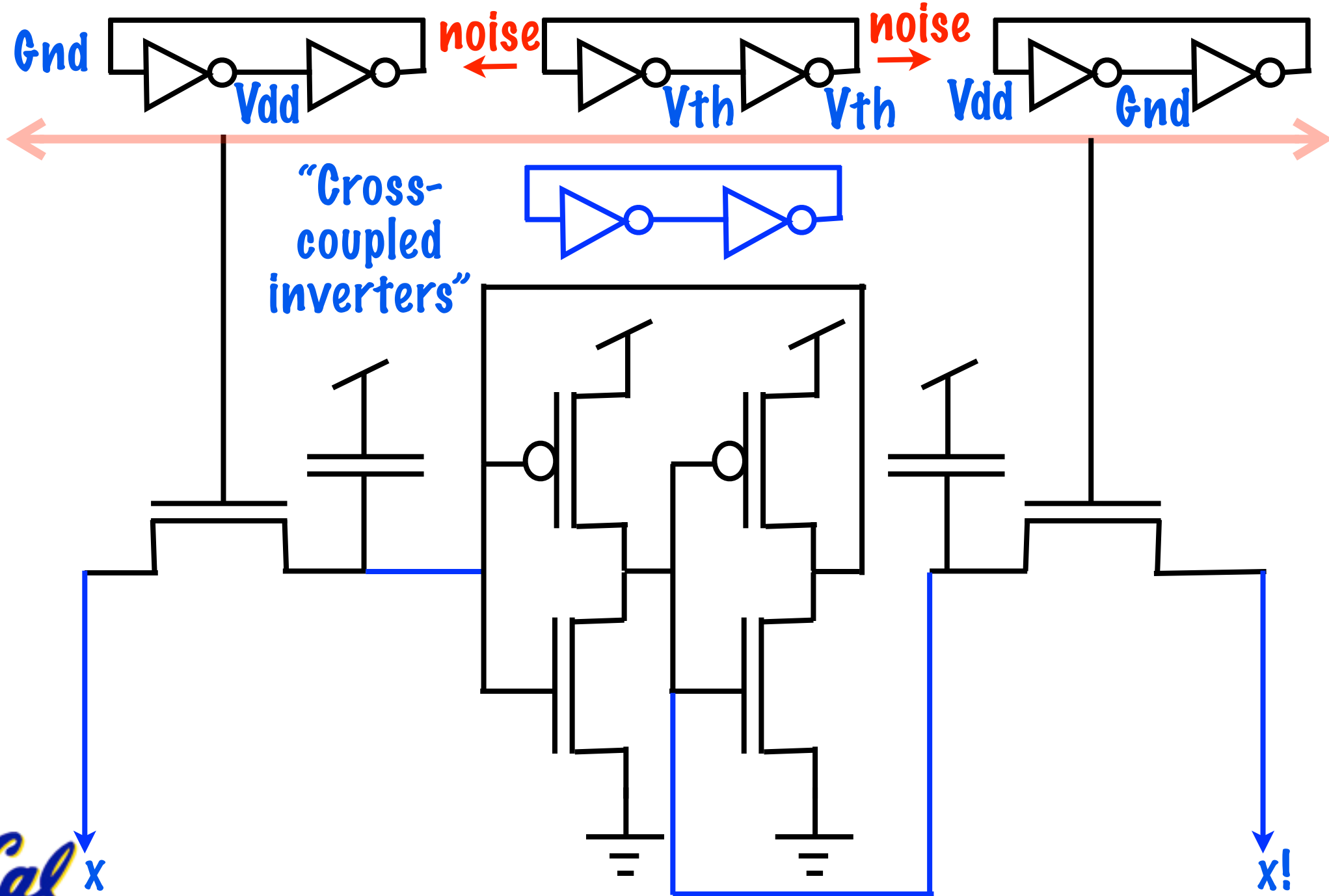
- **DRAM** has a 6-10X density advantage at the same technology generation.

SRAM advantages

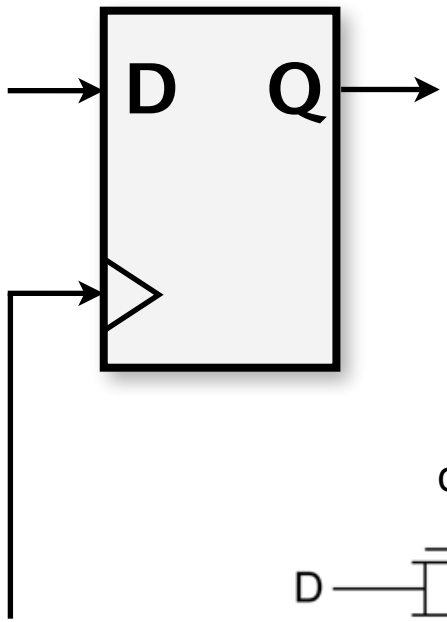
- **SRAM** has deterministic latency: its cells do not need to be refreshed.
- **SRAM** is much faster: transistors drive bitlines on reads.
- **SRAM** easy to design in logic fabrication process (and premium logic processes have SRAM add-ons)

Flip Flops Revisited

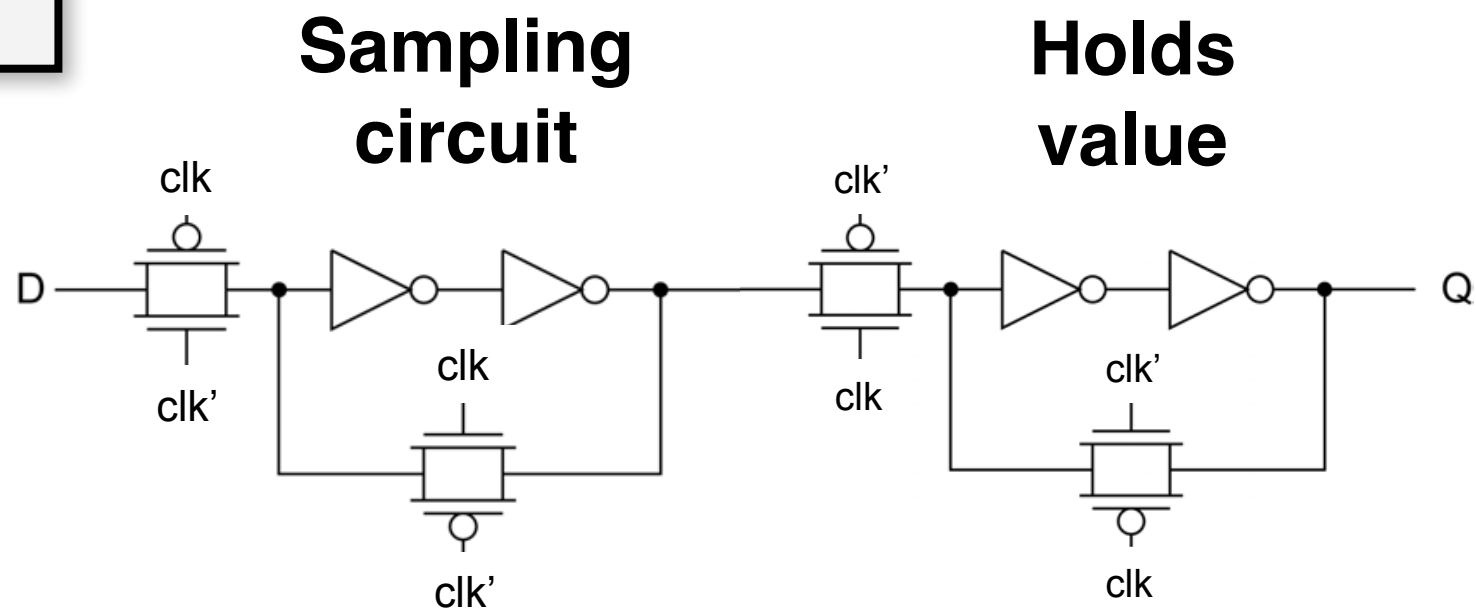
Recall: Static RAM cell (6 Transistors)



Recall: Positive edge-triggered flip-flop



A flip-flop "samples" right before the edge, and then "holds" value.

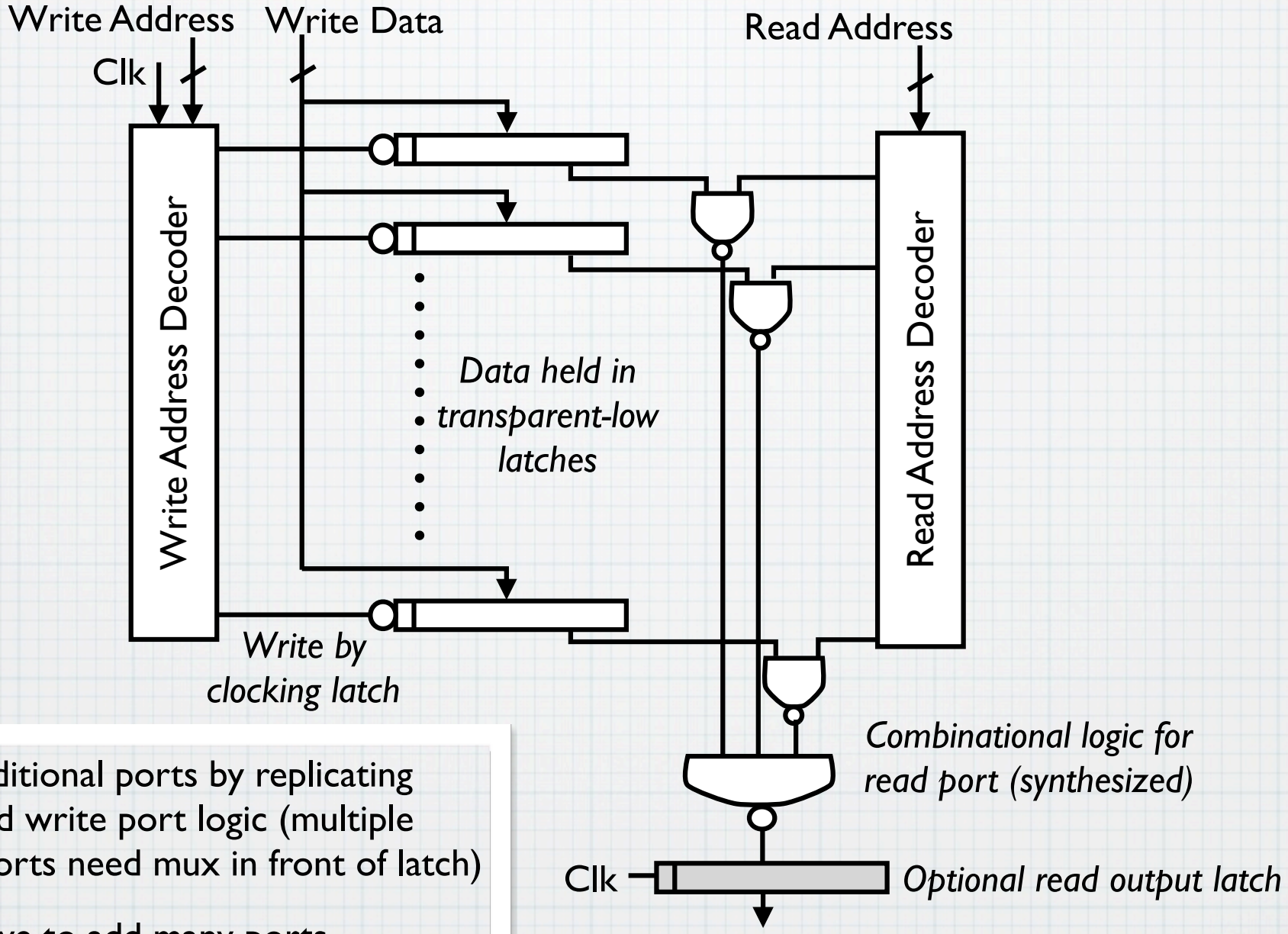


16 Transistors: Makes an SRAM look compact!

What do we get for the 10 extra transistors?

Clocked logic semantics.

Small Memories from Stdcell Latches



- Add additional ports by replicating read and write port logic (multiple write ports need mux in front of latch)
- Expensive to add many ports

Synthesized, custom, and SRAM-based register files, 40nm

For small register files, logic synthesis is competitive.

Not clear if the SRAM data points include area for register control, etc.

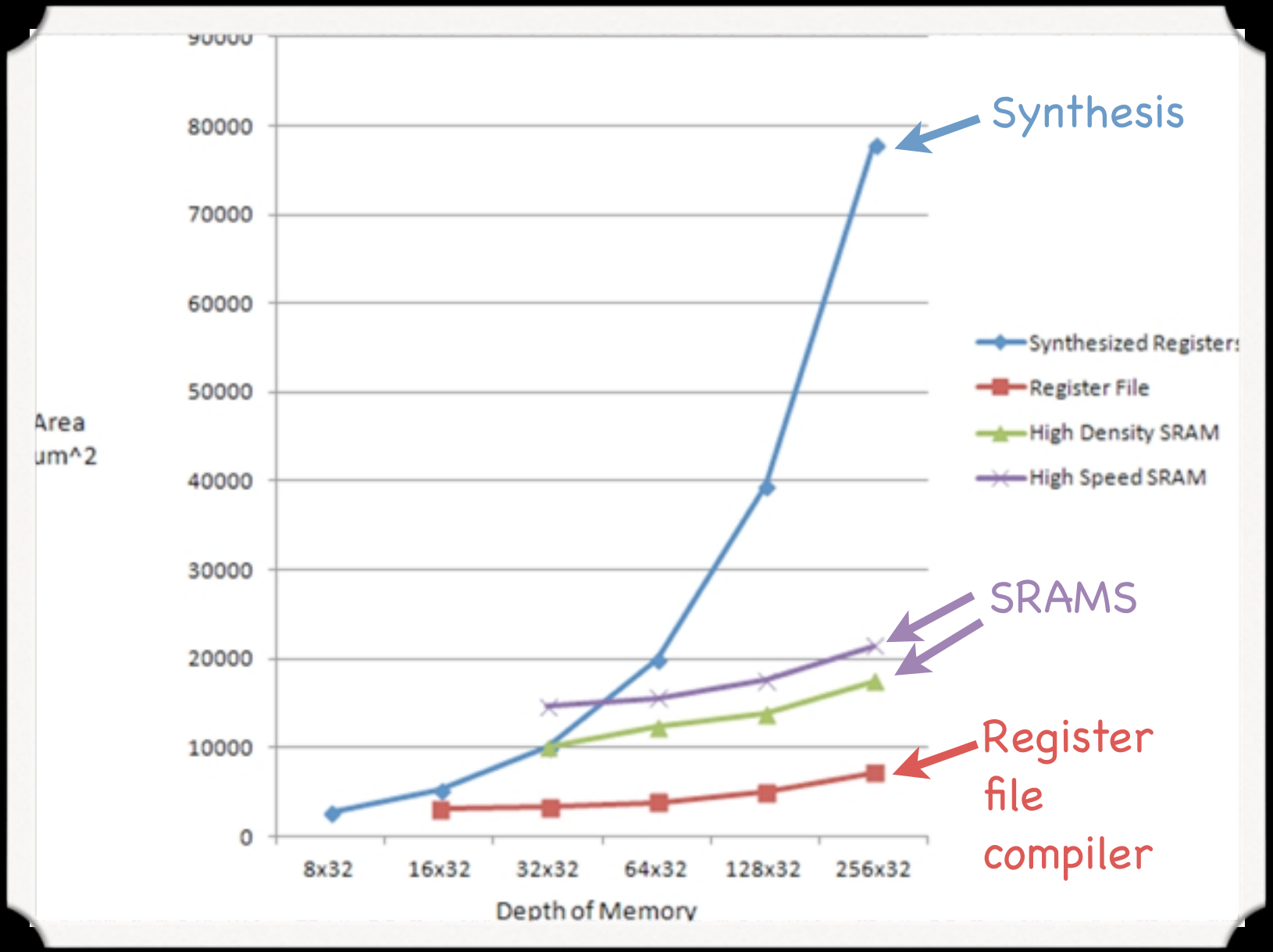
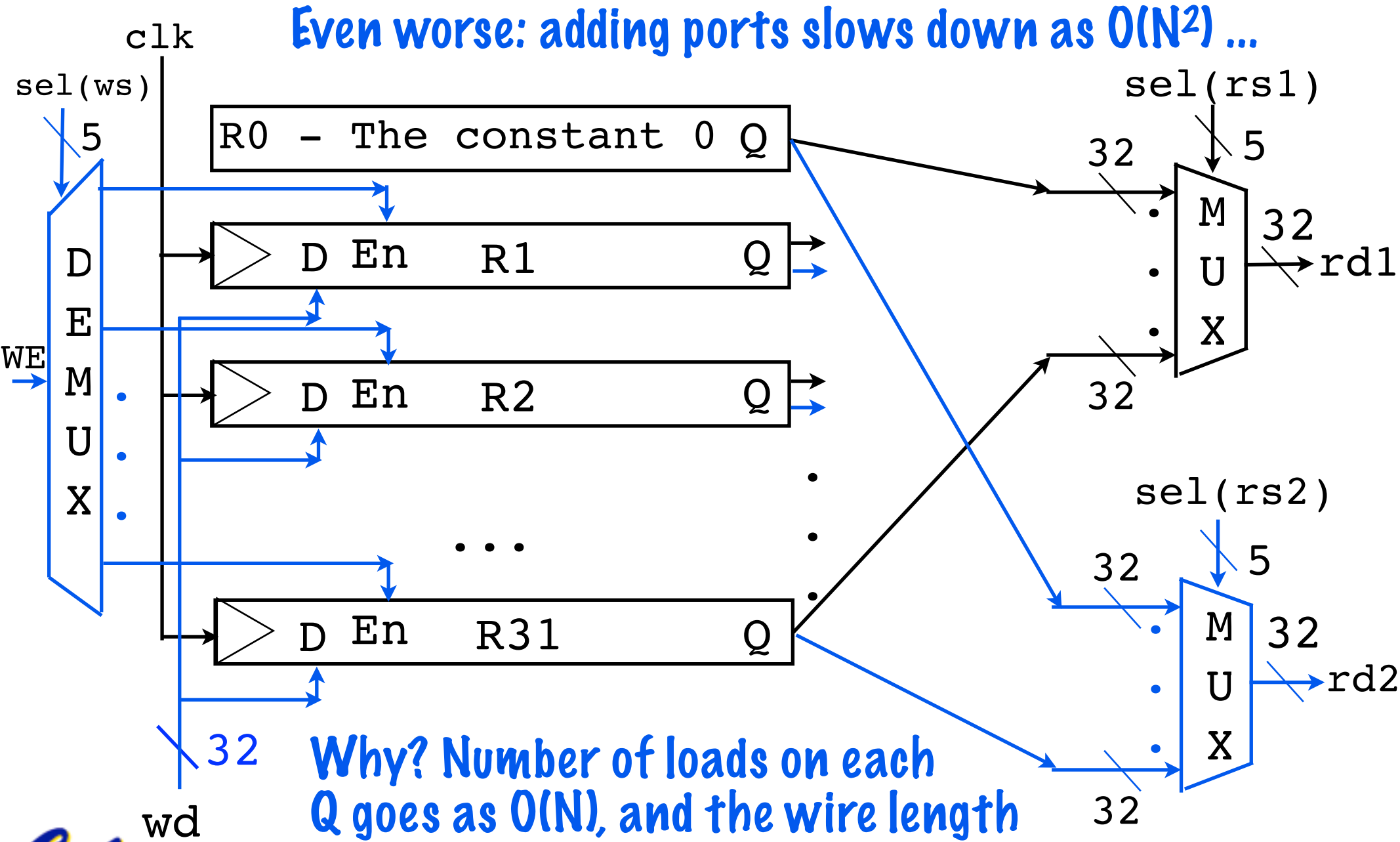


Figure 3: Using the raw area data, the physical implementation team can get a more accurate area estimation early in the RTL development stage for floorplanning purposes. This shows an example of this graph for a 1-port, 32-bit-wide SRAM.

Bhupesh Dasila

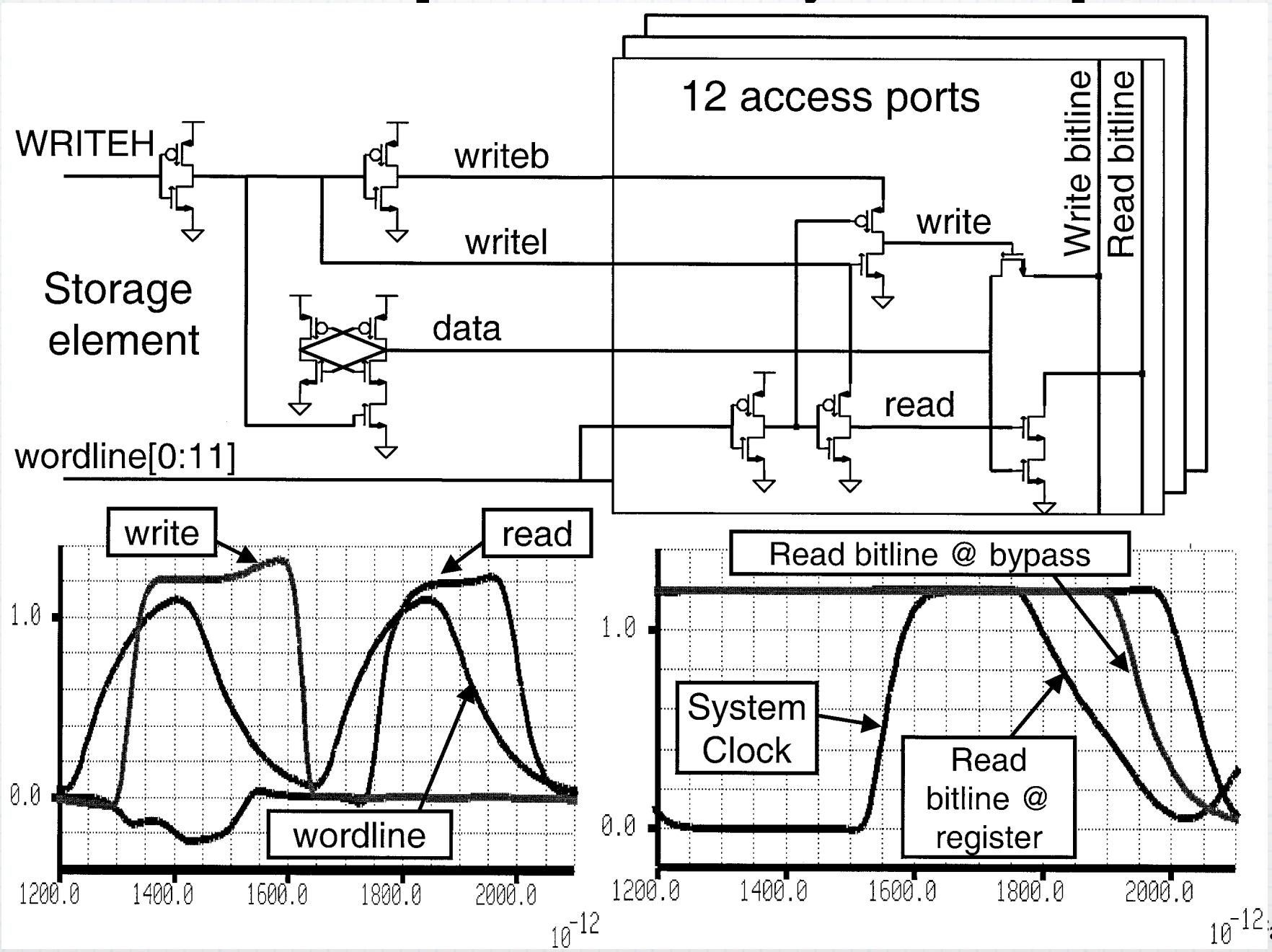
Memory Design Patterns

When register files get big, they get slow.



True Multiport Example: Itanium-2 Regfile

- Intel Itanium-2 [Fetzer et al, IEEE JSSCC 2002]



True Multiport Memory

Problem: Require simultaneous read and write access by multiple independent agents to a shared common memory.

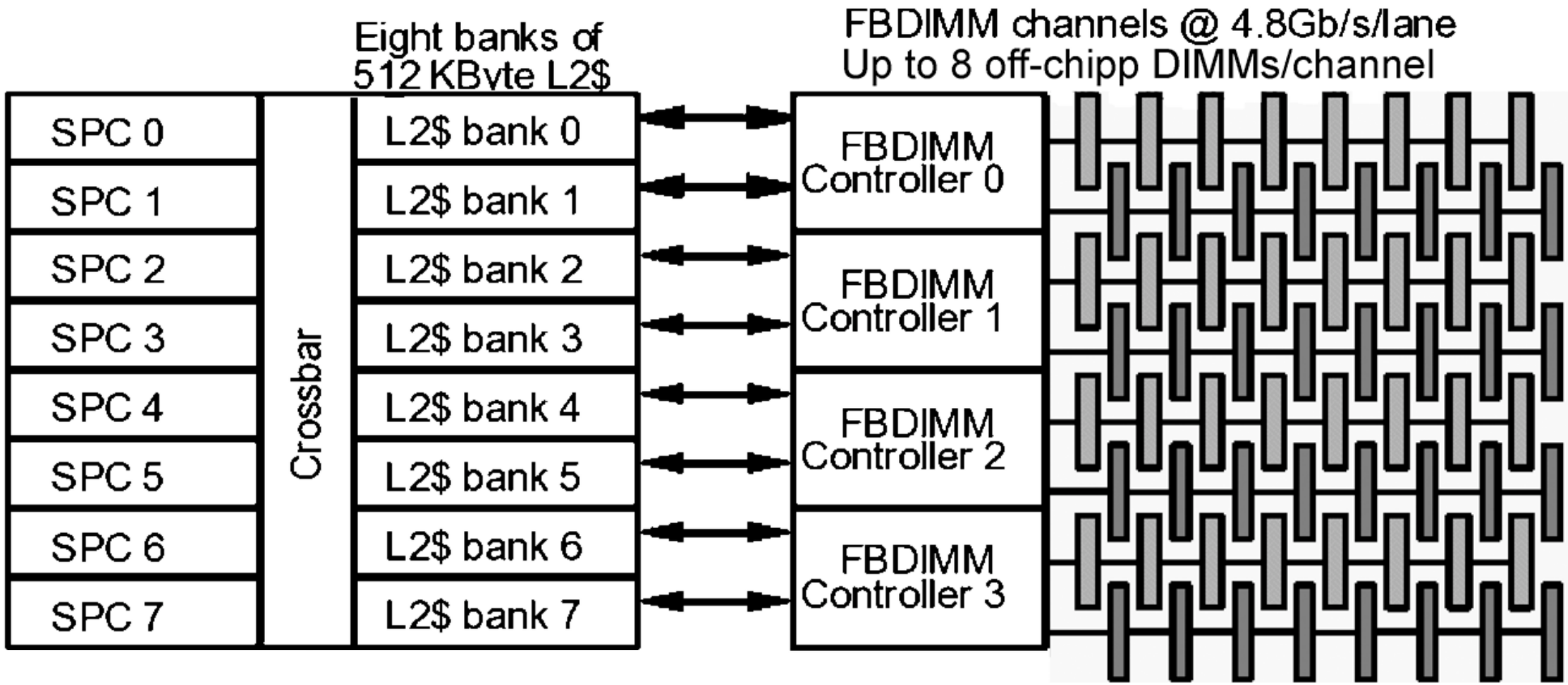
Solution: Provide separate read and write ports to each bit cell for each requester

Applicability: Where unpredictable access latency to the shared memory cannot be tolerated.

Consequences: High area, energy, and delay cost for large number of ports. Must define behavior when multiple writes on same cycle to same word (e.g., prohibit, provide priority, or combine writes).

Crossbar networks: many CPUs sharing cache banks

Sun Niagara II: 8 cores, 4MB L2, 4 DRAM channels



Each DRAM channel: 50 GB/s Read, 25 GB/s Write BW.

Crossbar BW: 270 GB/s total (Read + Write).
(Also shared by an I/O port, not shown)

Banked Multiport Memory

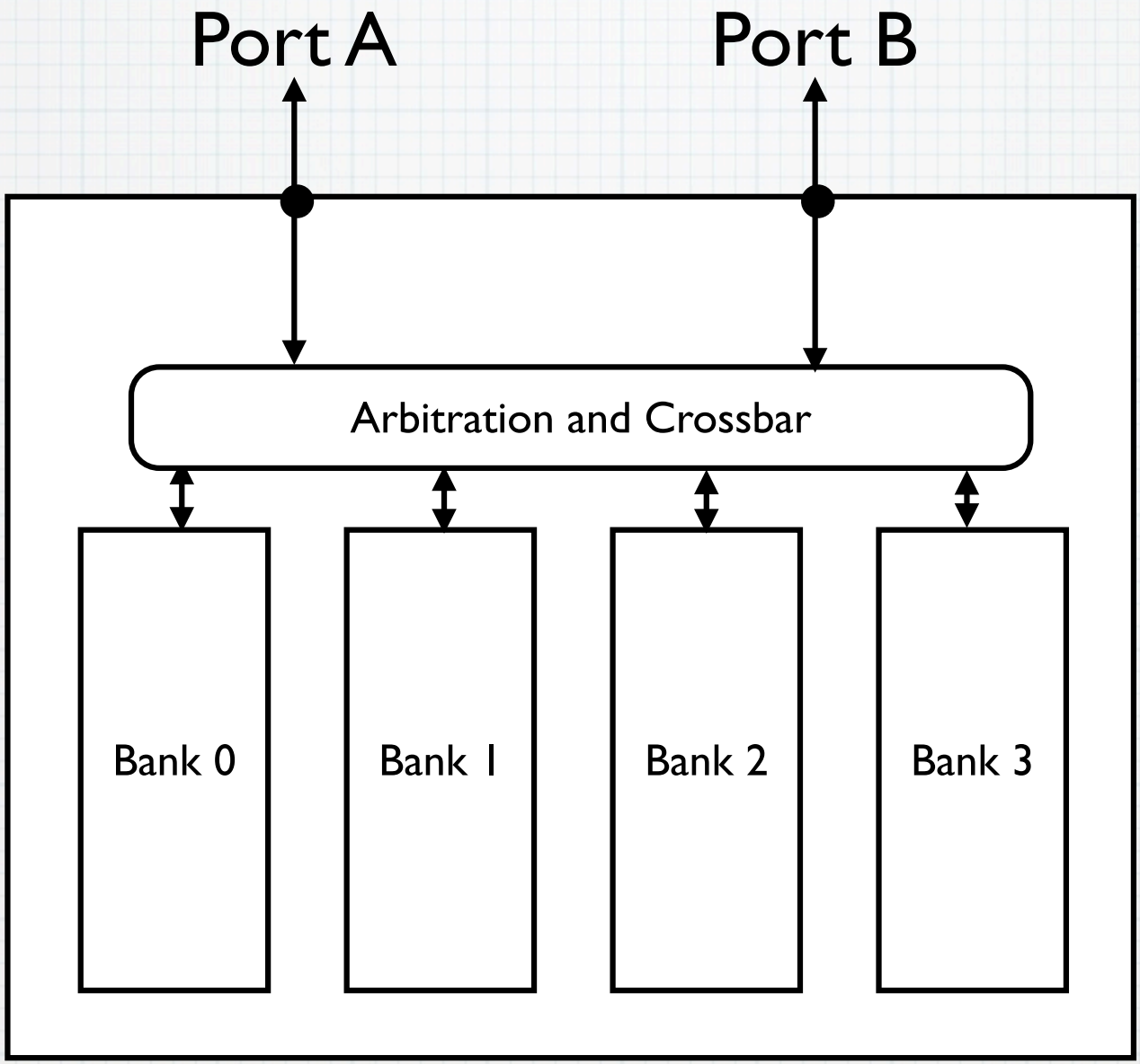
Problem: Require simultaneous read and write access by multiple independent agents to a large shared common memory.

Solution: Divide memory capacity into smaller banks, each of which has fewer ports. Requests are distributed across banks using a fixed hashing scheme. Multiple requesters arbitrate for access to same bank/port.

Applicability: Requesters can tolerate variable latency for accesses. Accesses are distributed across address space so as to avoid "hotspots".

Consequences: Requesters must wait arbitration delay to determine if request will complete. Have to provide interconnect between each requester and each bank/port. Can have greater, equal, or lesser number of banks*ports/bank compared to total number of external access ports.

Banked Multiport Memory



Stream-Buffered Multiport Memory

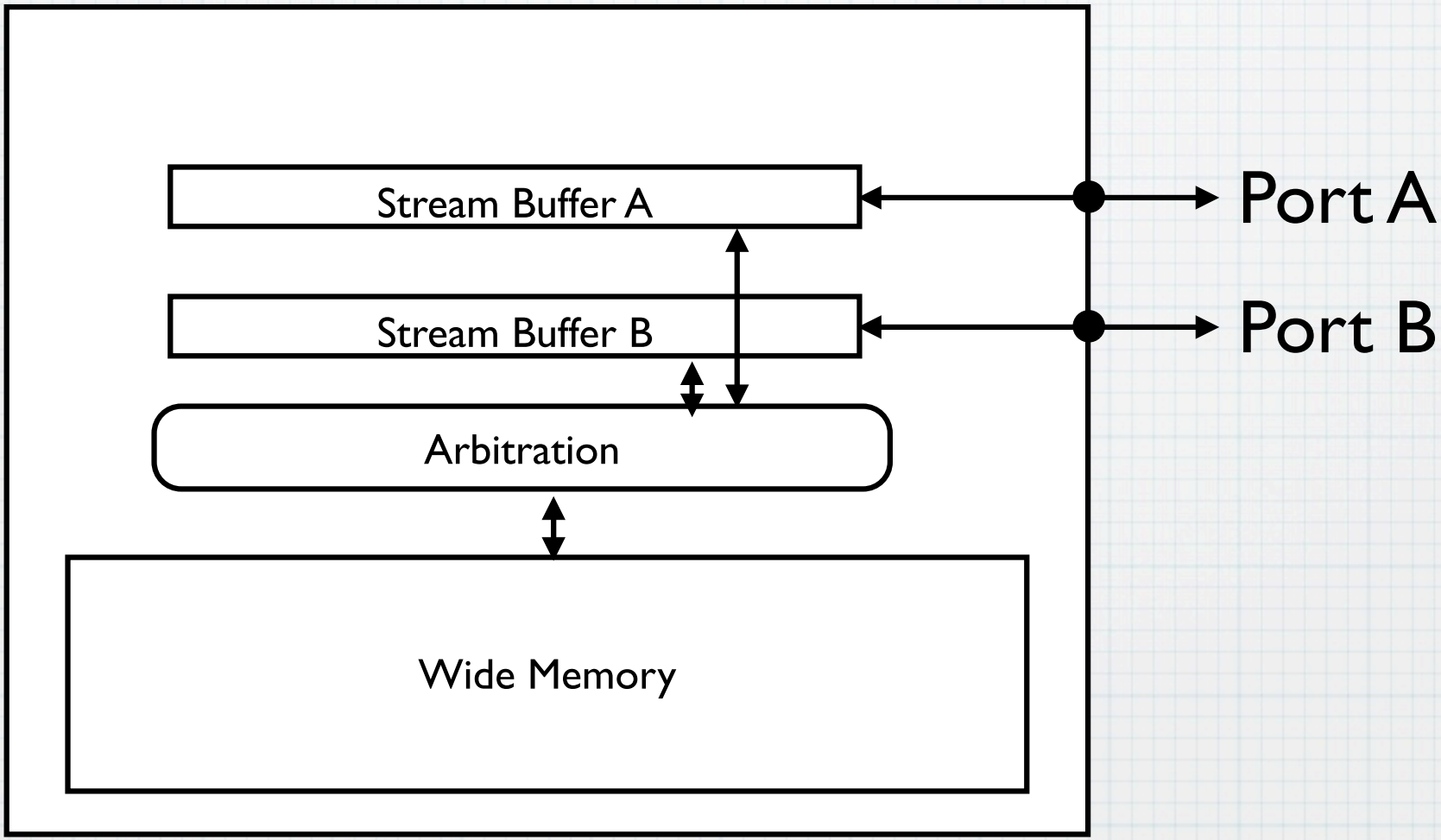
Problem: Require simultaneous read and write access by multiple independent agents to a large shared common memory, where each requester usually makes multiple sequential accesses.

Solution: Organize memory to have a single wide port. Provide each requester with an internal stream buffer that holds width of data returned/consumed by each memory access. Each requester can access own stream buffer without contention, but arbitrates with others to read/write stream buffer from memory.

Applicability: Requesters make mostly sequential requests and can tolerate variable latency for accesses.

Consequences: Requesters must wait arbitration delay to determine if request will complete. Have to provide stream buffers for each requester. Need sufficient access width to serve aggregate bandwidth demands of all requesters, but wide data access can be wasted if not all used by requester. Have to specify memory consistency model between ports (e.g., provide stream flush operations).

Stream-Buffered Multiport Memory



Cached Multiport Memory

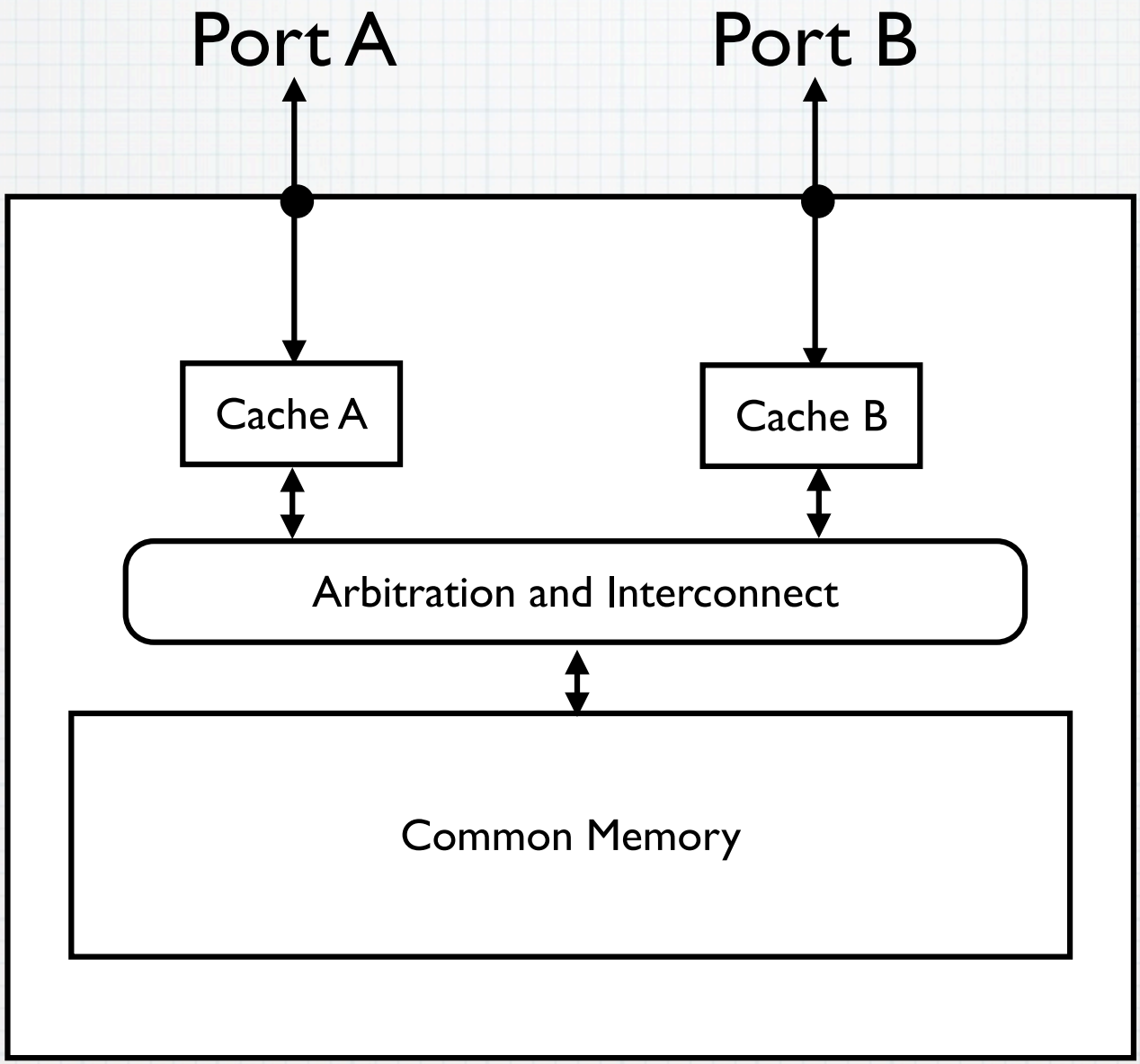
Problem: Require simultaneous read and write access by multiple independent agents to a large shared common memory.

Solution: Provide each access port with a local cache of recently touched addresses from common memory, and use a cache coherence protocol to keep the cache contents in sync.

Applicability: Request streams have significant temporal locality, and limited communication between different ports.

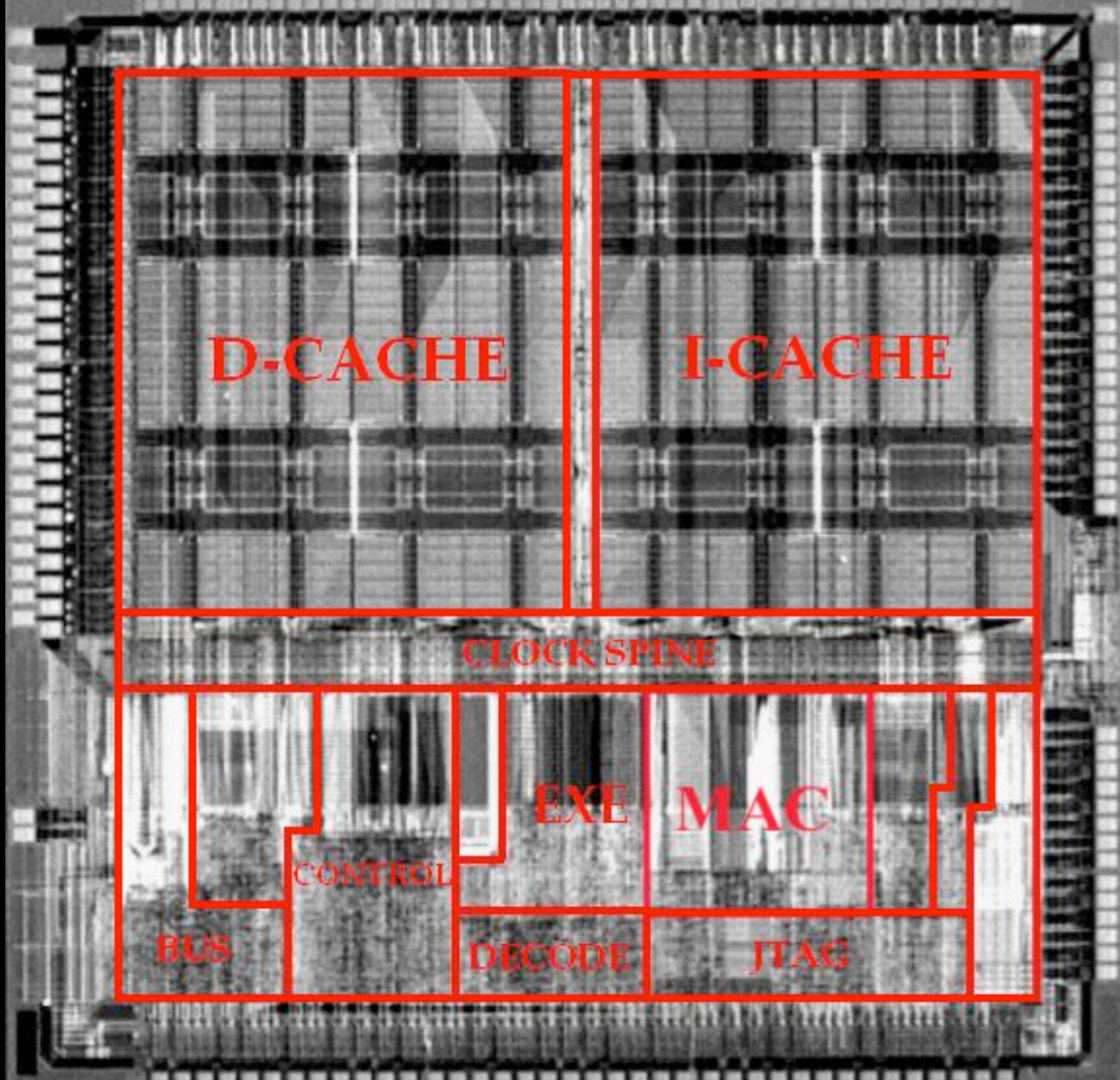
Consequences: Requesters will experience variable delay depending on access pattern and operation of cache coherence protocol. Tag overhead in both area, delay, and energy/access. Complexity of cache coherence protocol.

Cached Multiport Memory



ARM CPU

The arbiter and interconnect on the last slide is how the two caches on this chip share access to DRAM.



Replicated-State Multiport Memory

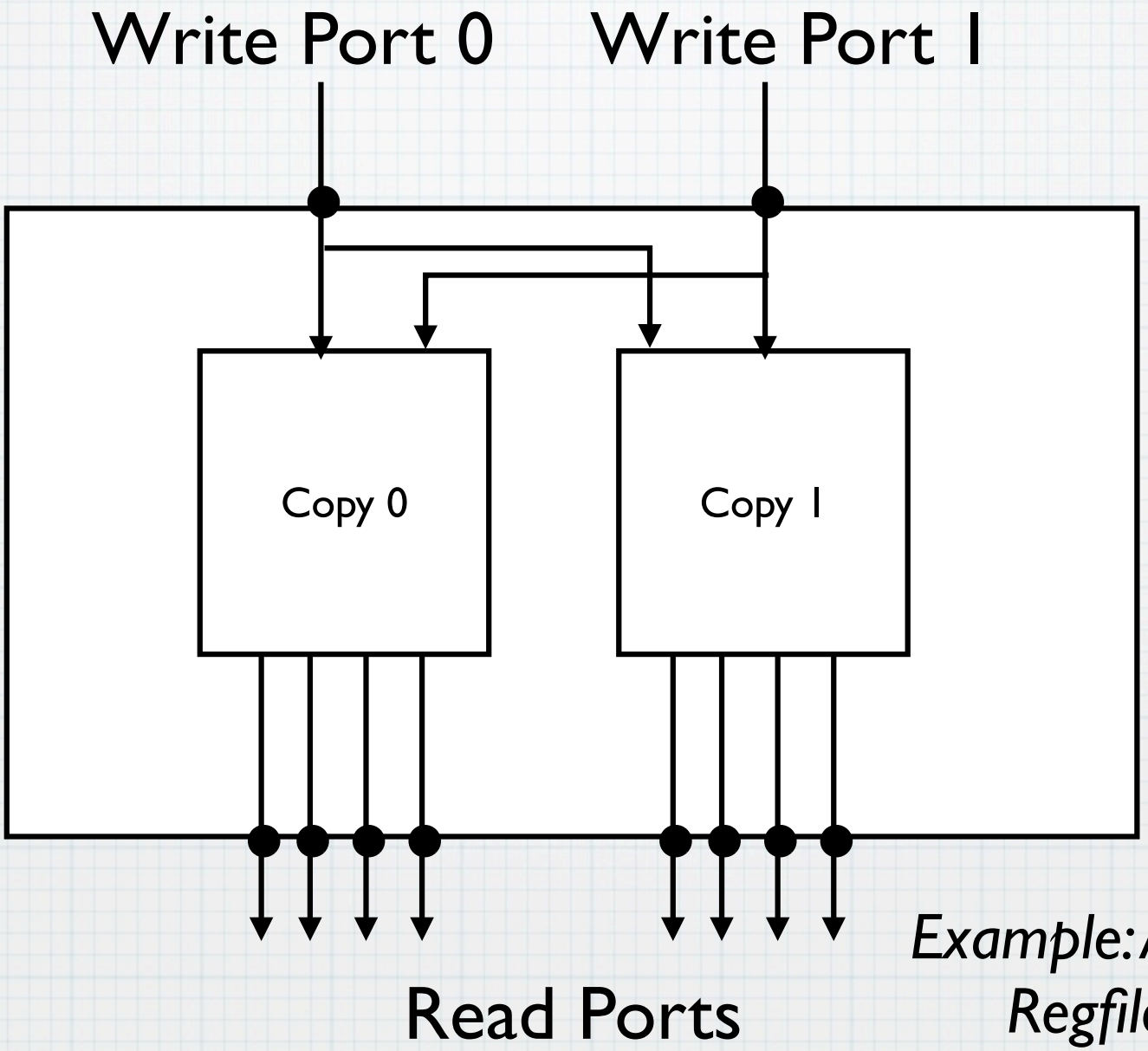
Problem: Require simultaneous read and write access by multiple independent agents to a small shared common memory. Cannot tolerate variable latency of access.

Solution: Replicate storage and divide read ports among replicas. Each replica has enough write ports to keep all replicas in sync.

Applicability: Many read ports required, and variable latency cannot be tolerated.

Consequences: Potential increase in latency between some writers and some readers.

Replicated-State Multiport Memory



Example: Alpha 21264
Regfile clusters

Memory: Technology and Patterns

- **Memory, the 10,000 ft view.** Latency, from Steve Wozniak to the power wall.
- **How DRAM works.** Memory design when low cost per bit is the priority.

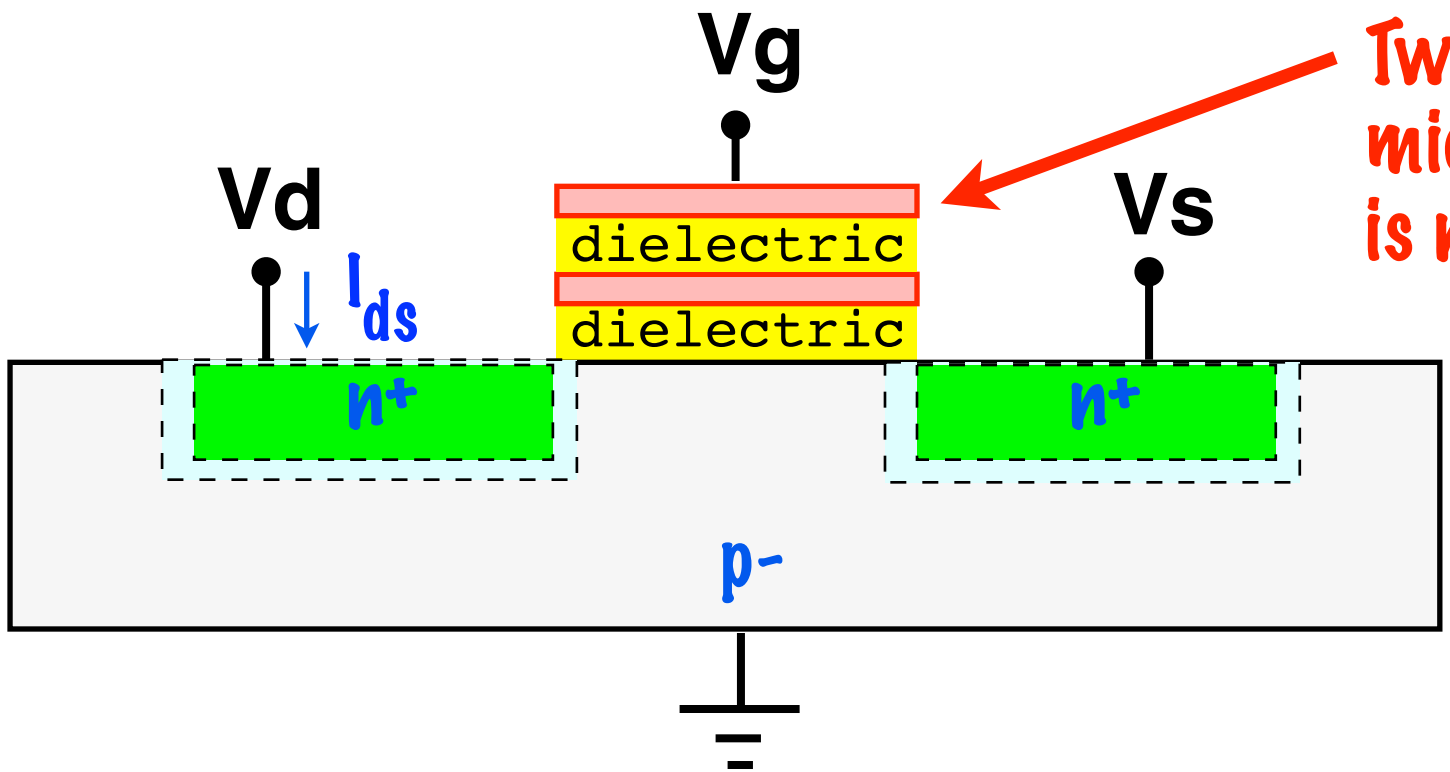
Break

- **How SRAM works.** The memory technology available on logic dies.
- **Memory design patterns.** Ways to use SRAM in your project designs.

Device physics ...

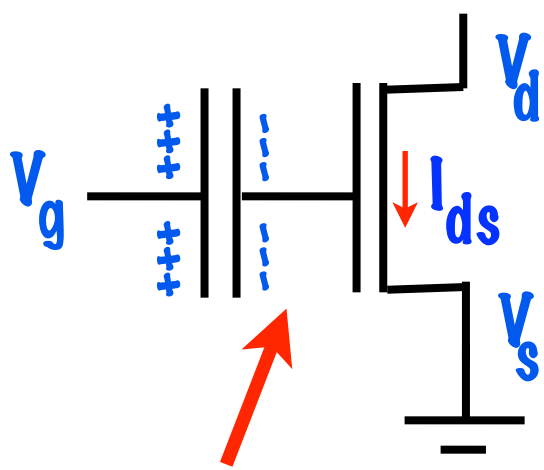
NAND Flash Memory

The physics of non-volatile memory



Two gates. But the middle one is not connected.

1. Electrons "placed" on floating gate stay there for many years (ideally).
2. 10,000 electrons on floating gate shift transistor threshold by 2V.
3. In a memory array, shifted transistors hold "0", unshifted hold "1".

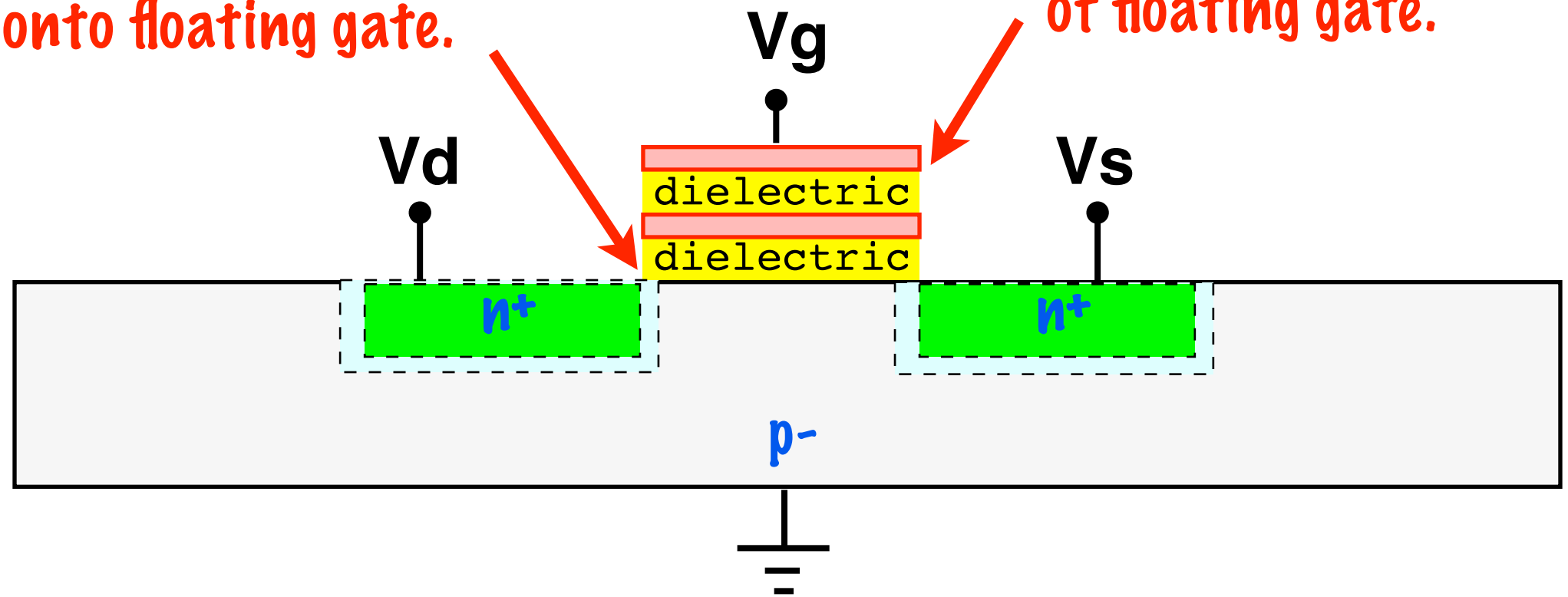


"Floating gate".

Moving electrons on/off floating gate

A high drain voltage injects "hot electrons" onto floating gate.

A high gate voltage "tunnels" electrons off of floating gate.



1. Hot electron injection and tunneling produce tiny currents, thus writes are slow.

2. High voltages damage the floating gate. Too many writes and a bit goes "bad".

Architecture ...

NAND Flash Memory

Flash: Disk Replacement

Chip "remembers" for 10 years.

Presents memory to the CPU as a set of pages.

Page format:

2048 Bytes	+	64 Bytes
(user data)		(meta data)

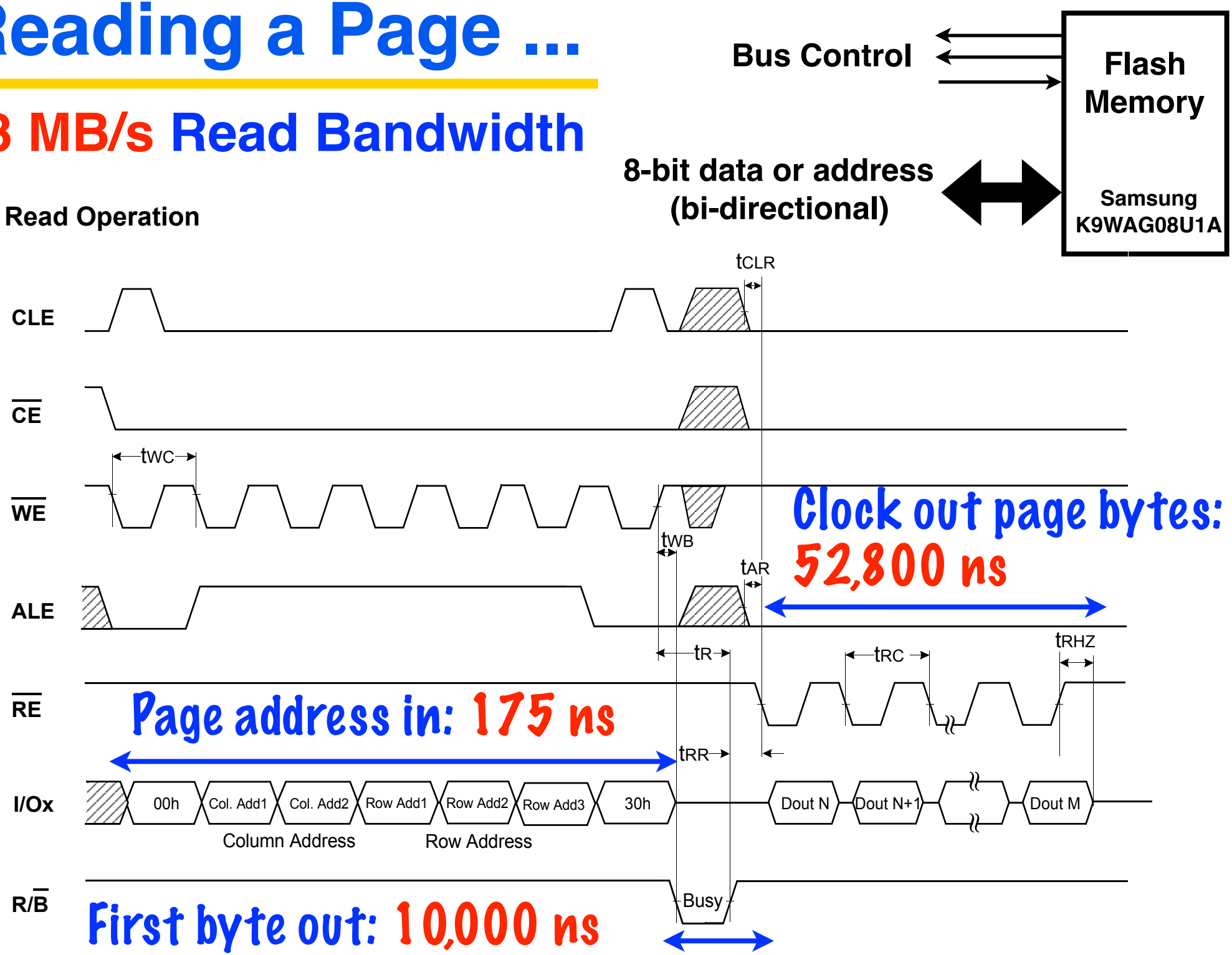
1GB Flash: 512K pages

2GB Flash: 1M pages

4GB Flash: 2M pages

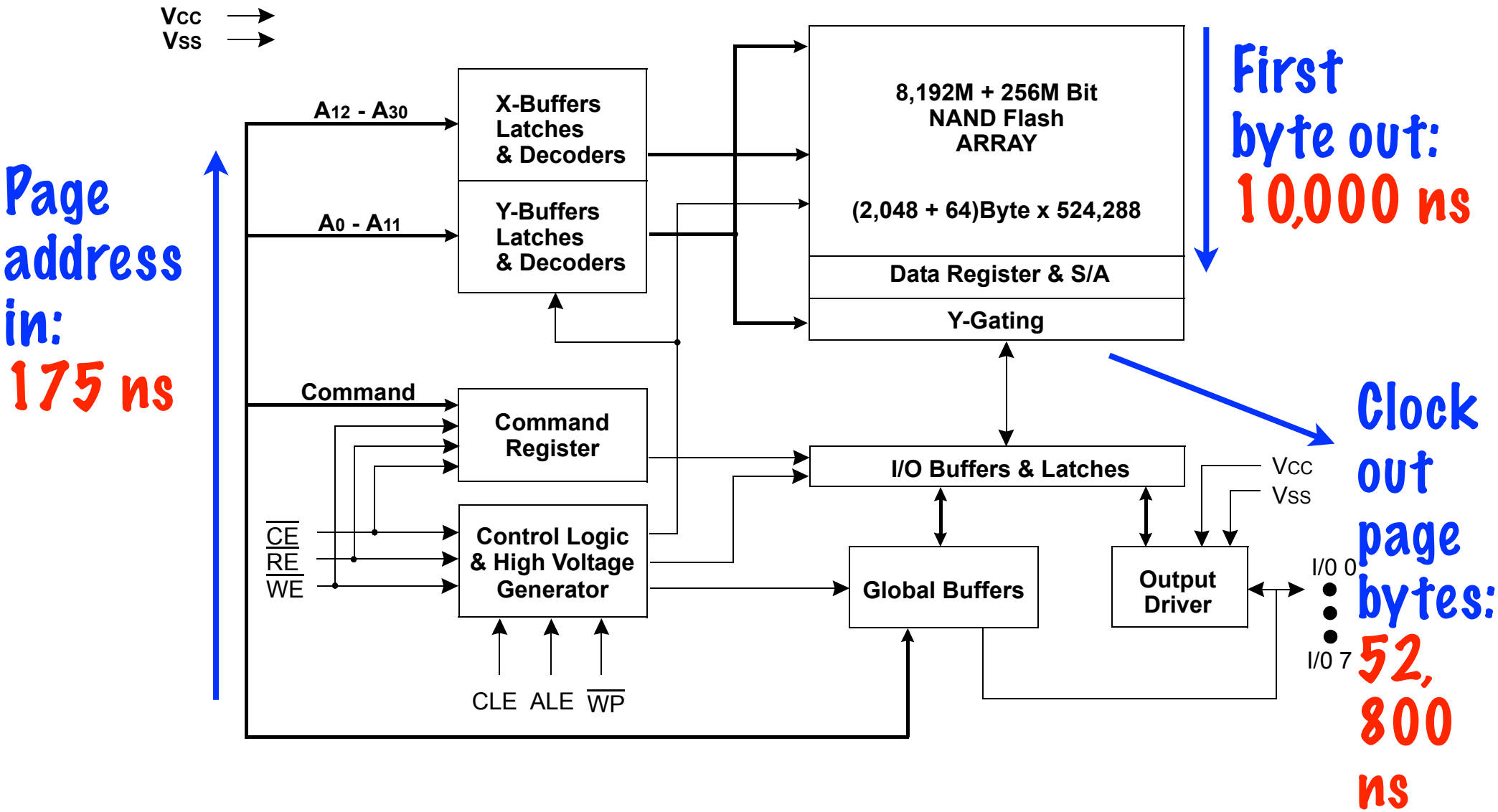
Reading a Page ...

33 MB/s Read Bandwidth



Where Time Goes

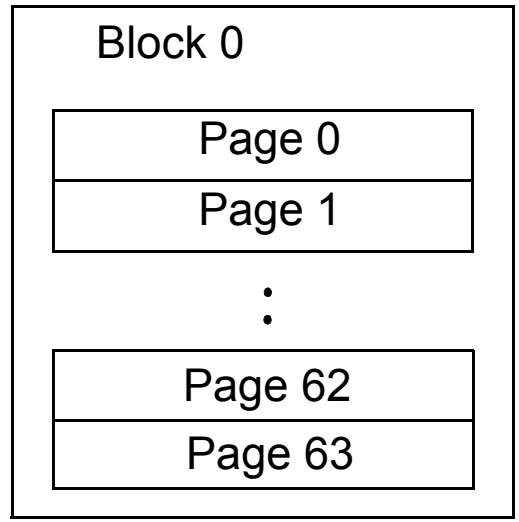
Figure 1. K9K8G08U0A Functional Block Diagram



Writing a Page ...

A page lives in a block of 64 pages:

- 1GB Flash: 8K blocks
- 2GB Flash: 16K blocks
- 4GB Flash: 32K blocks



To write a page:

1. Erase all pages in the block (cannot erase just one page). Time: 1,500,000 ns

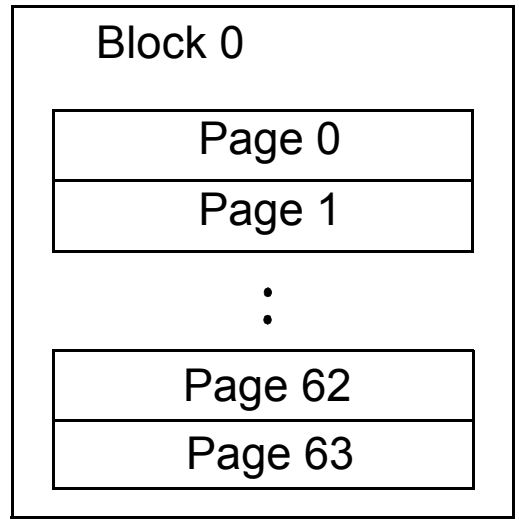
2. May program each page individually, exactly once. Time: 200,000 ns per page.

Block lifetime: 100,000 erase/program cycles.

Block Failure

Even when new, not all blocks work!

- **1GB: 8K blocks, 160 may be bad.**
- **2GB: 16K blocks, 220 may be bad.**
- **4GB: 32K blocks, 640 may be bad.**



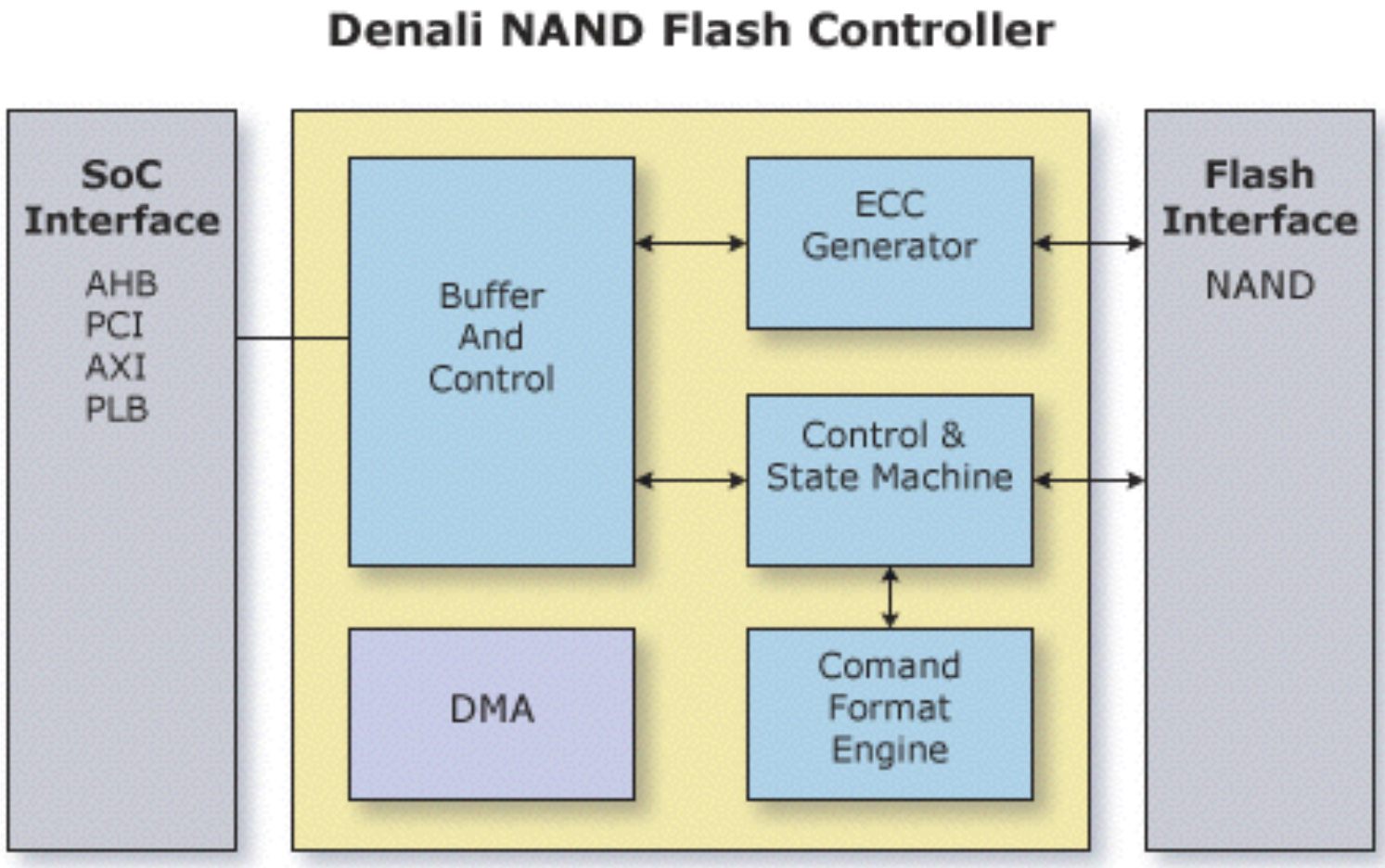
During factory testing, Samsung writes good/bad info for each block in the meta data bytes.

2048 Bytes	+	64 Bytes
(user data)		(meta data)

After an erase/program, chip can say "write failed", and block is now "bad". OS must recover (migrate bad block data to a new block). Bits can also go bad "silently" (!!!).

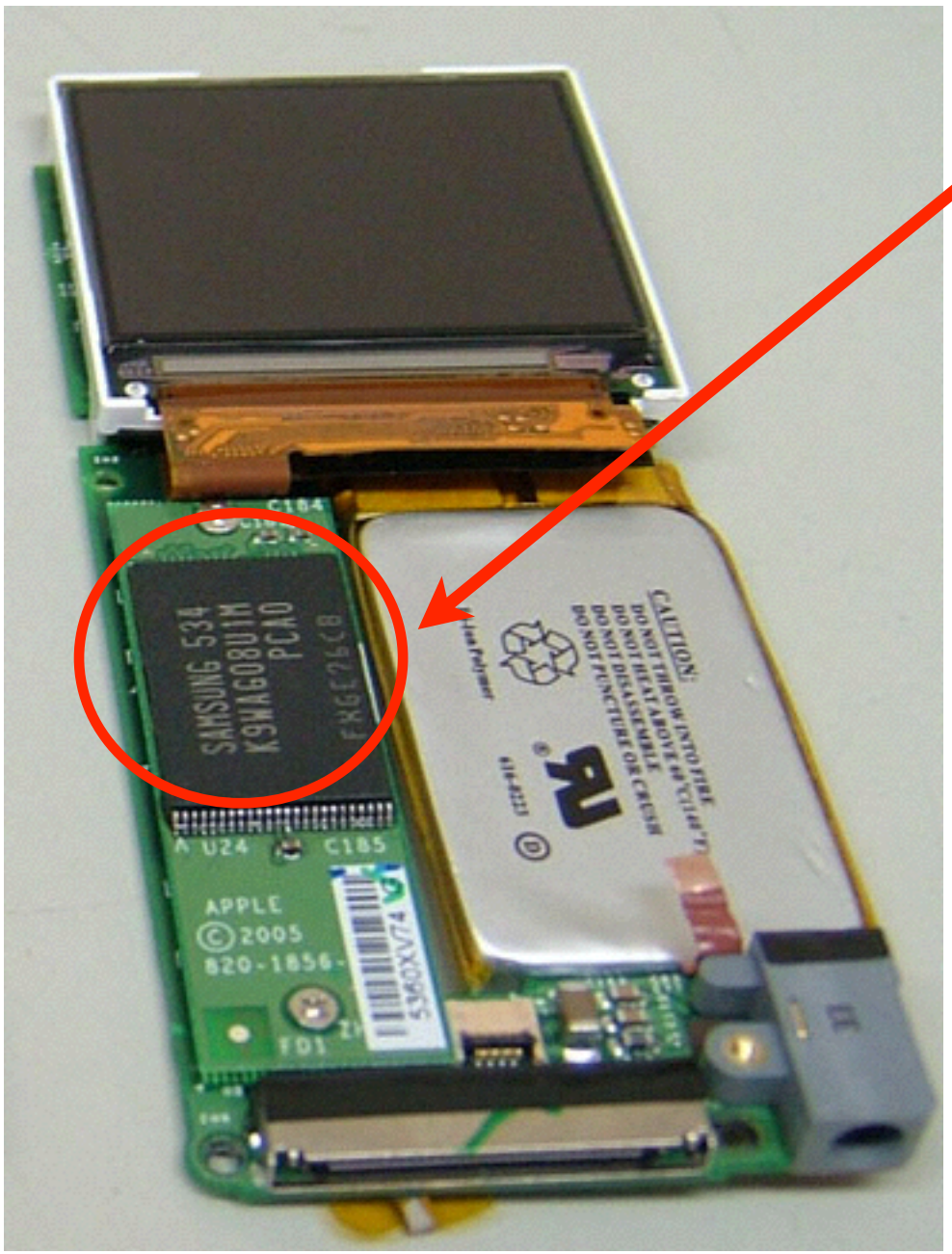
Flash controllers: Chips or Verilog IP ...

Flash memory controller manages write lifetime management, block failures, silent bit errors ...



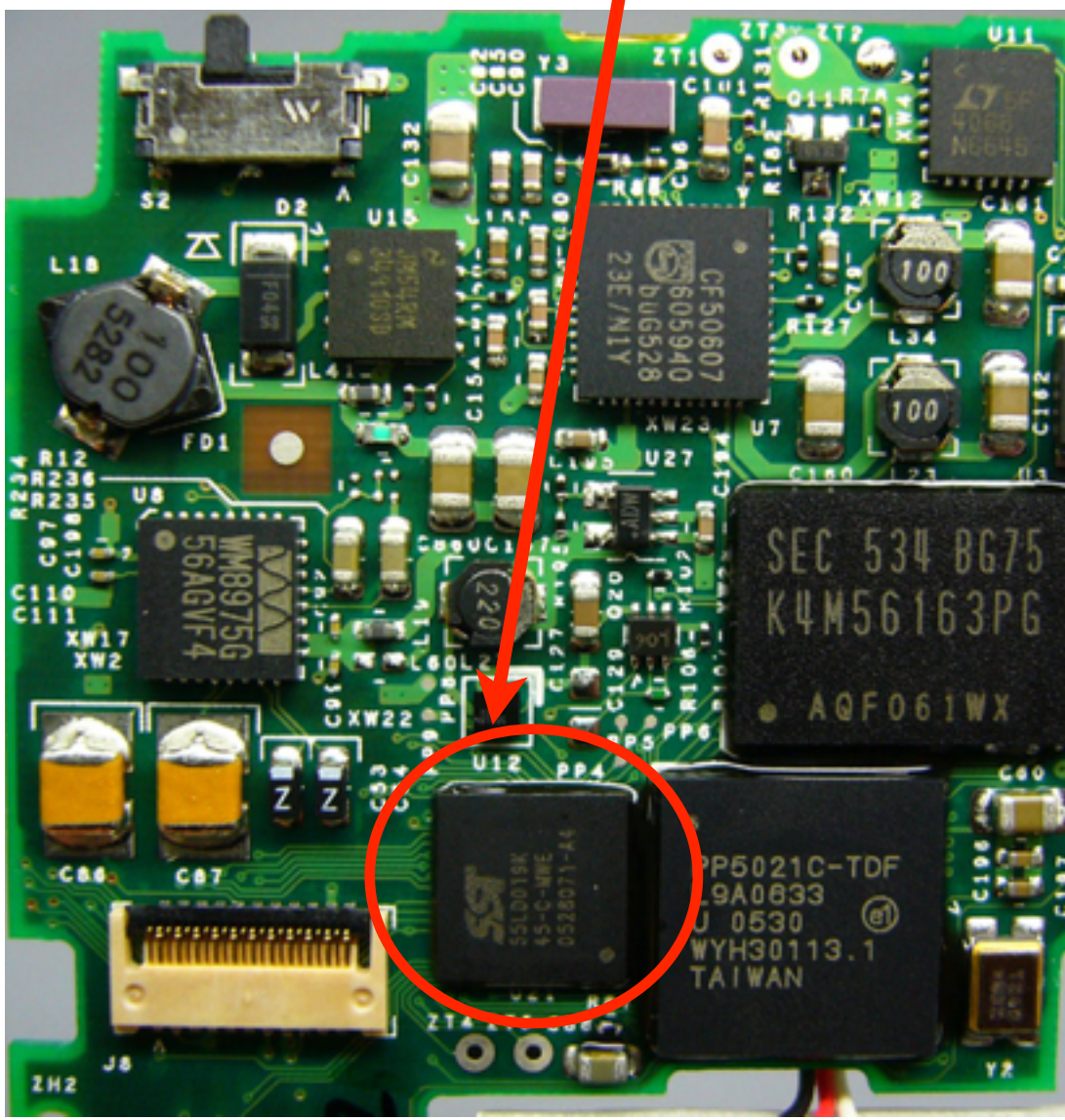
Software sees a "perfect" disk-like storage device.

Recall: iPod 2005 ...



Flash memory

Flash controller



Appendix: Krste's version of this talk (2012)

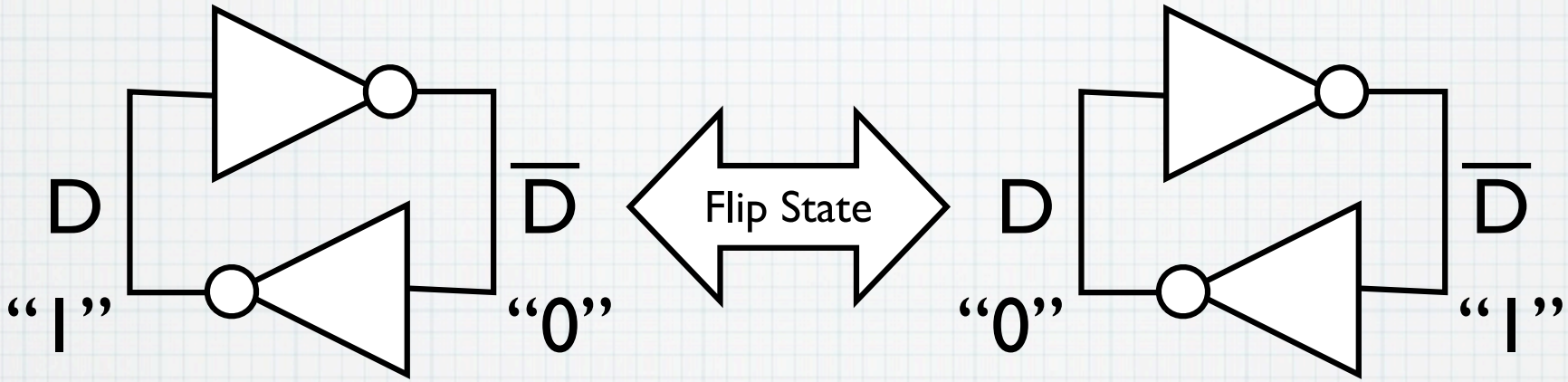
CS250 VLSI Systems Design

Lecture 9: Memory

John Wawrzynek, Jonathan Bachrach,
with
Krste Asanovic, John Lazzaro
and
Rimas Avizienis (TA)

UC Berkeley
Fall 2012

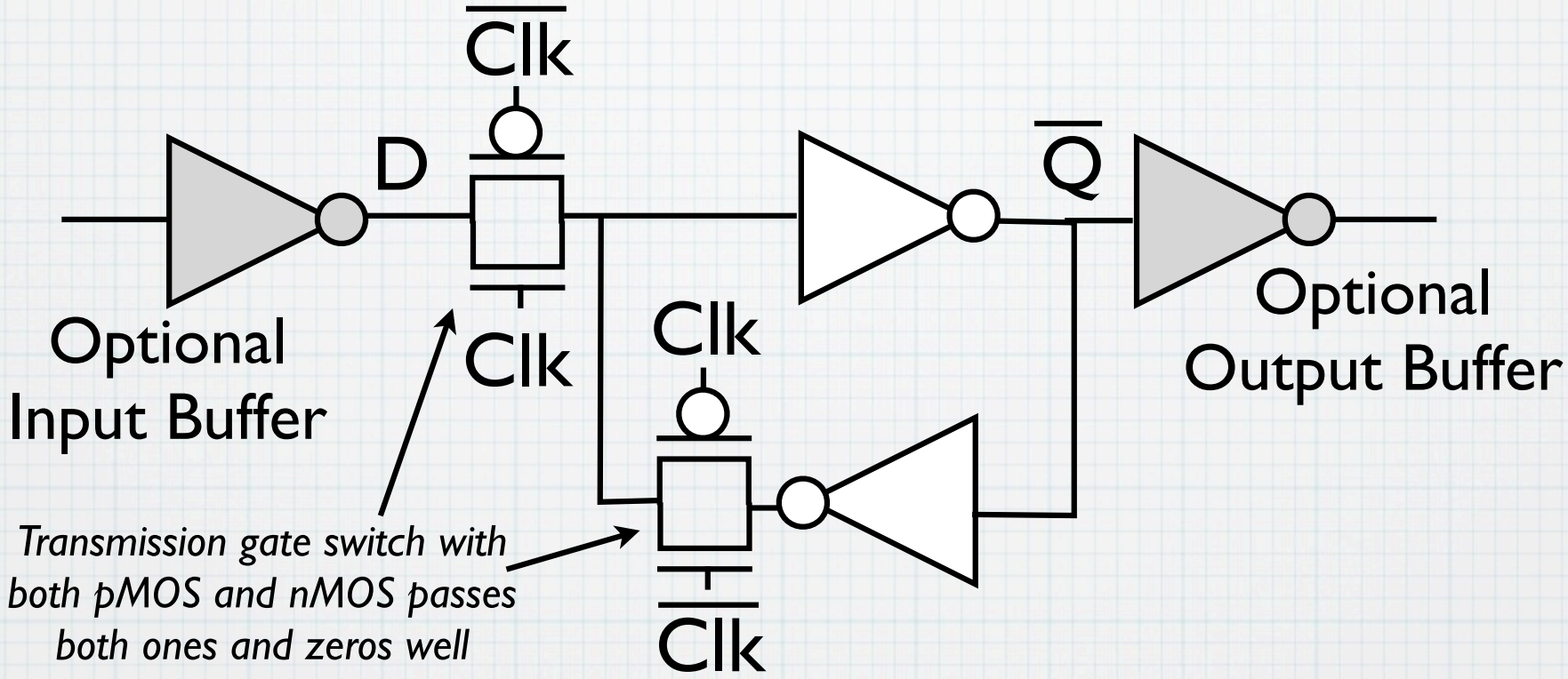
CMOS Bistable



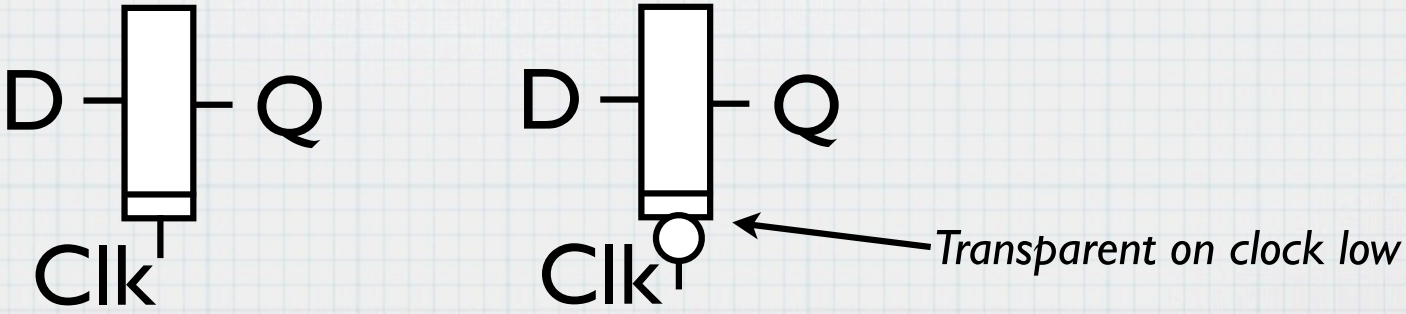
- Cross-coupled inverters used to hold state in CMOS
- "Static" storage in powered cell, no refresh needed
  - If a storage node leaks or is pushed slightly away from correct value, non-linear transfer function of high-gain inverter removes noise and recirculates correct value
- To write new state, have to force nodes to opposite state

CMOS Transparent Latch

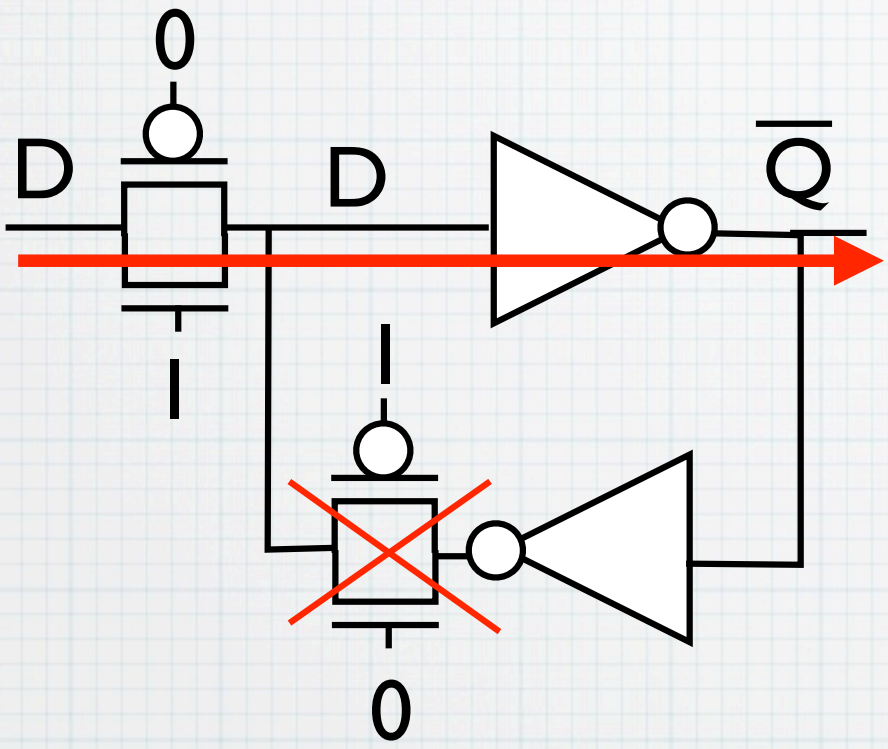
Latch transparent (output follows input) when clock is high, holds last value when clock is low



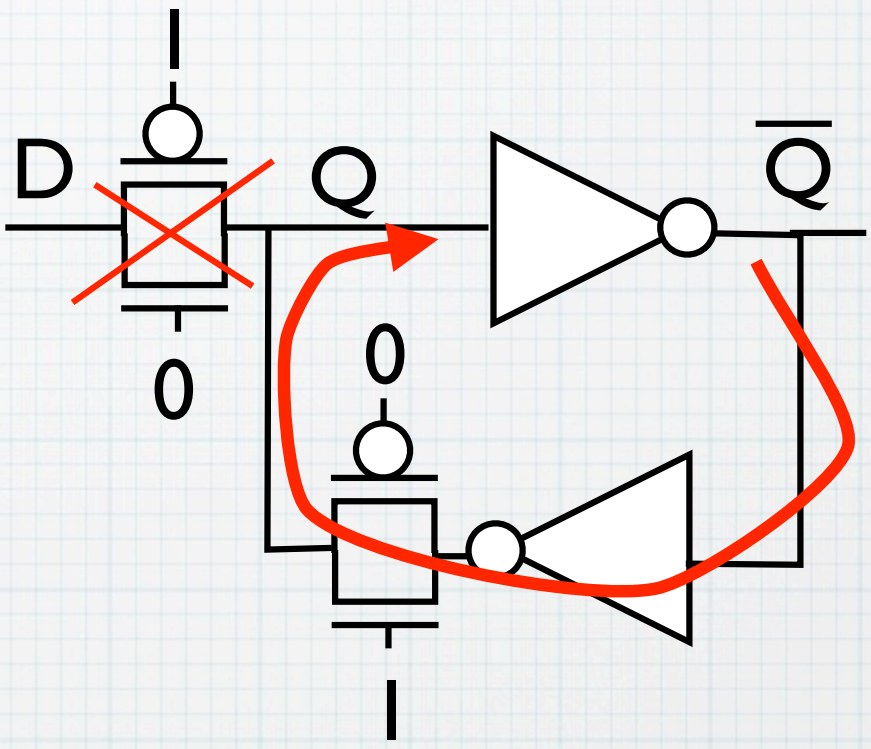
Schematic Symbols



Latch Operation

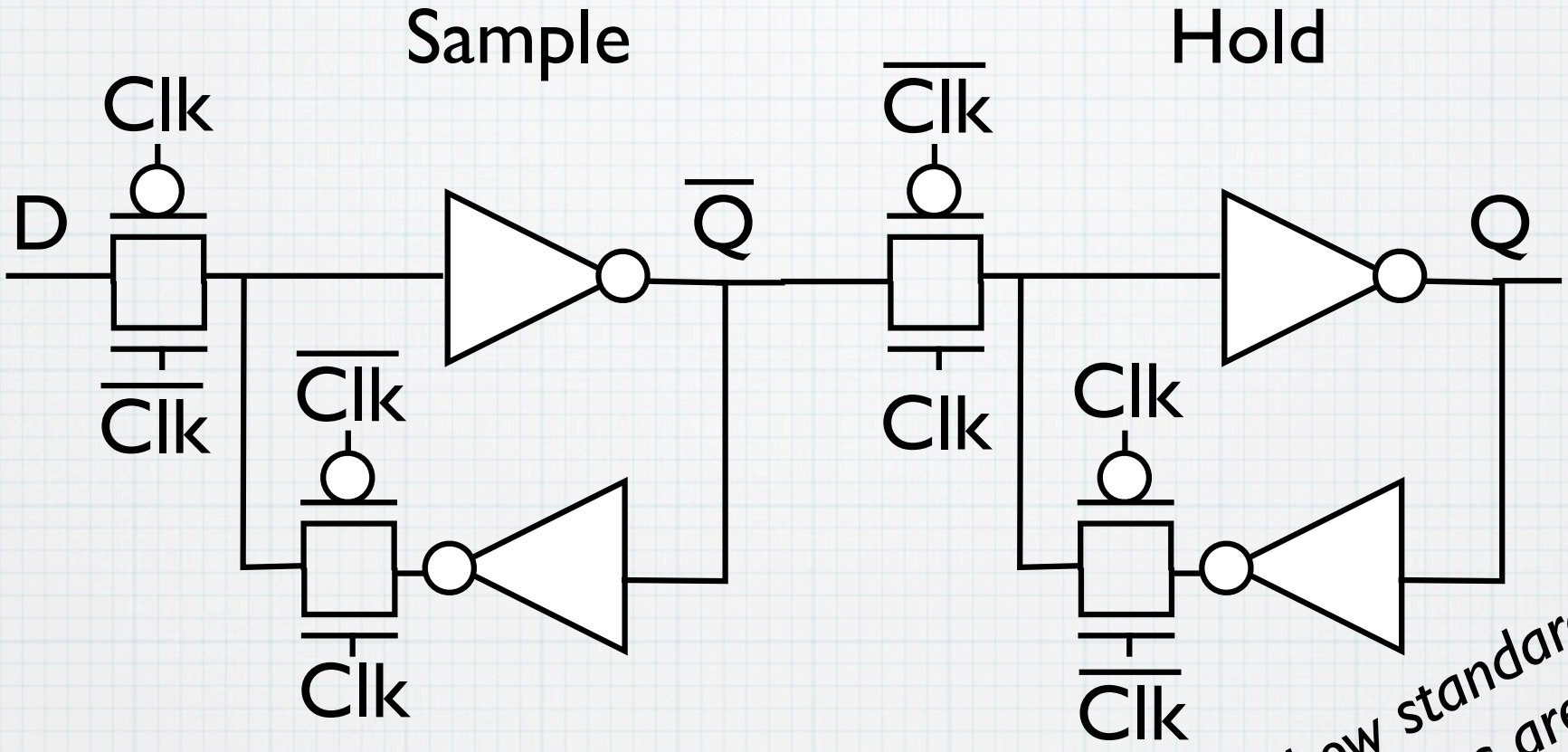


Clock High
Latch Transparent



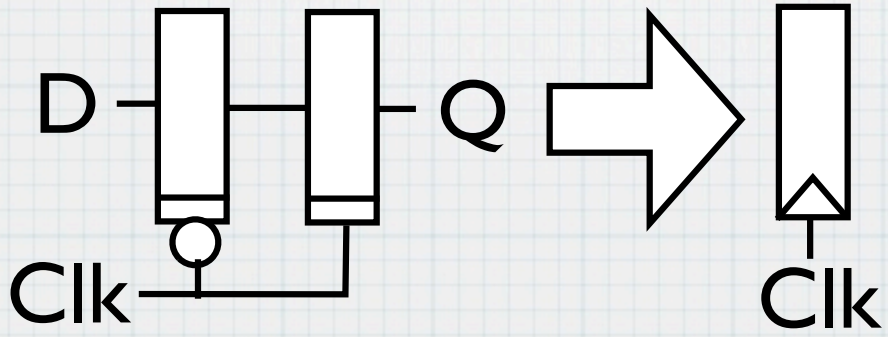
Clock Low
Latch Holding

Flip-Flop as Two Latches

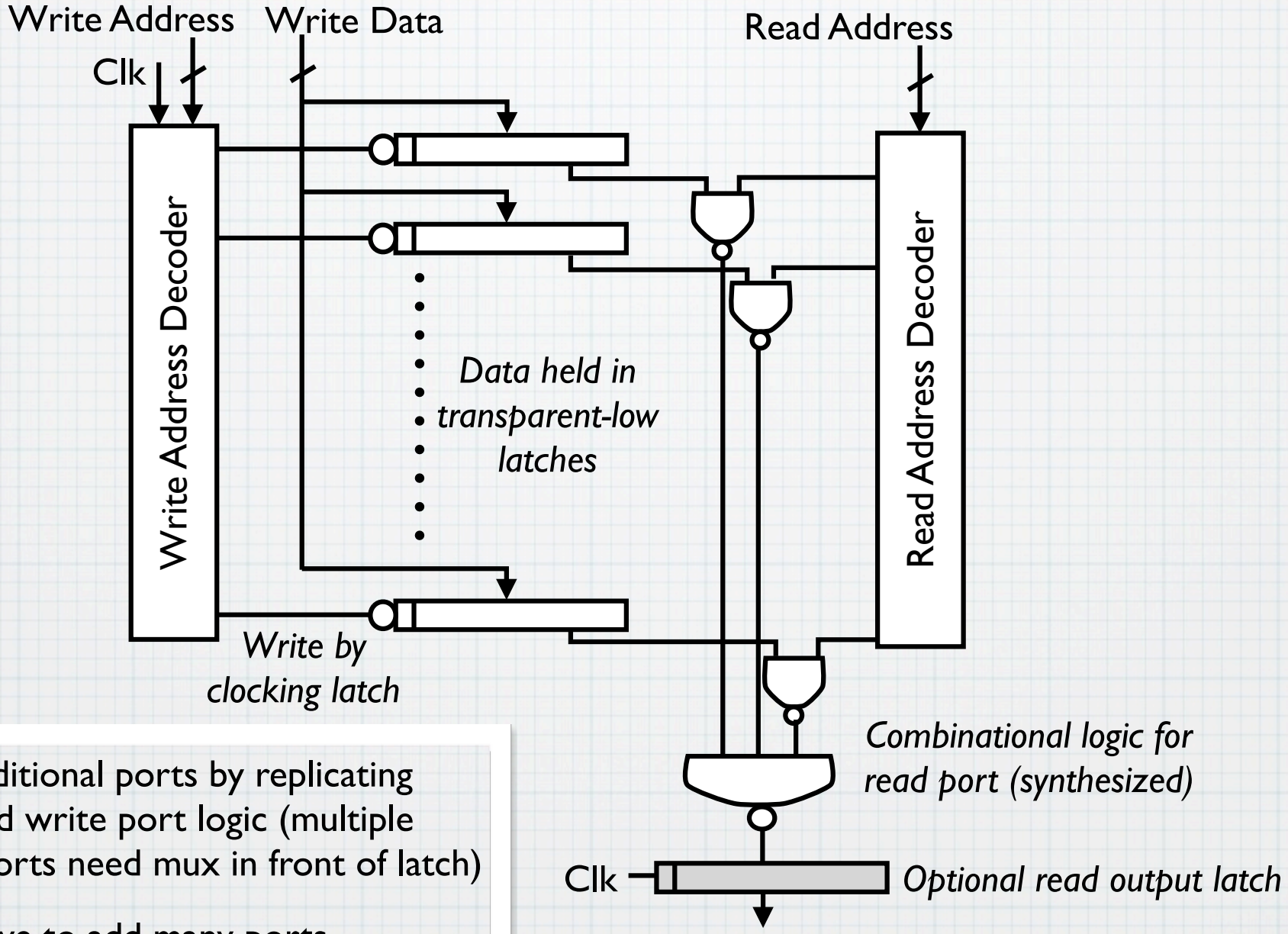


This is how standard cell flip-flops are built(usually with extra in/out buffers)

Schematic Symbols

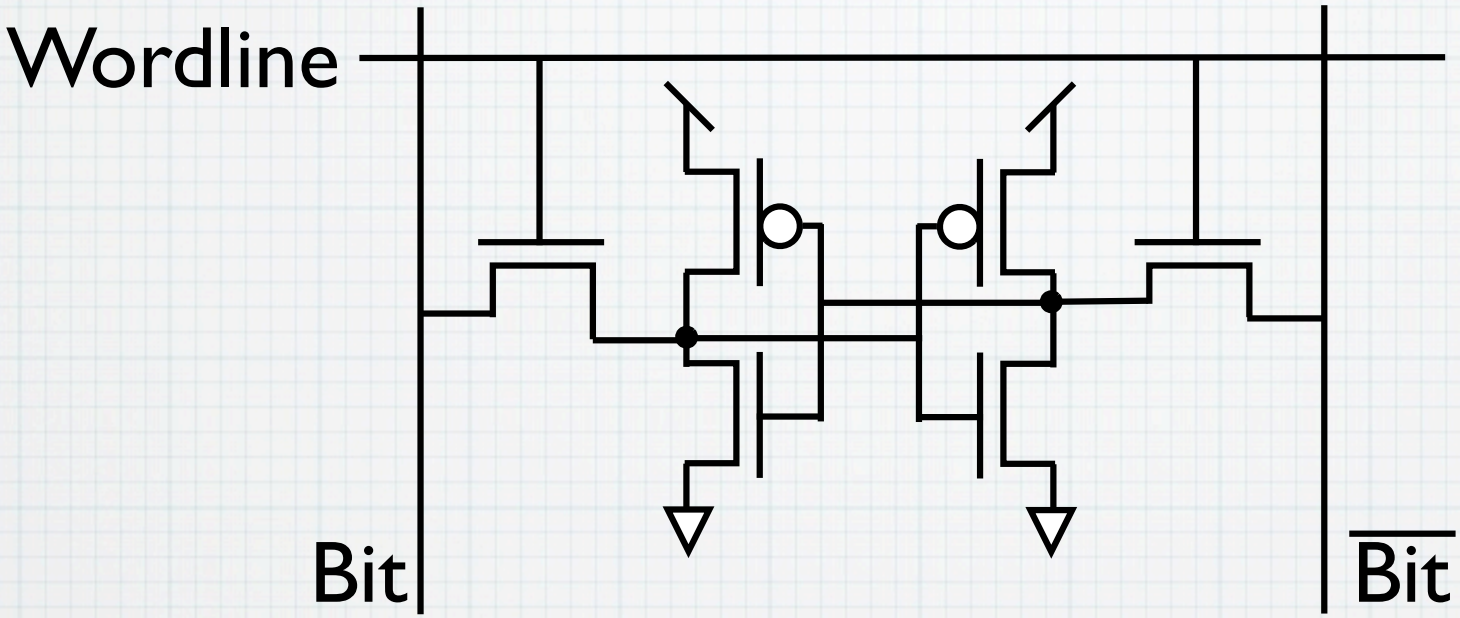


Small Memories from Stdcell Latches



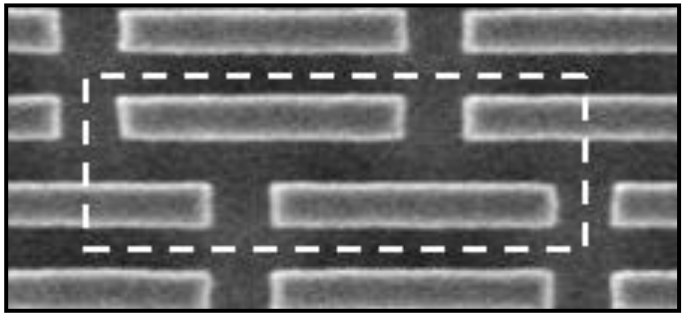
- Add additional ports by replicating read and write port logic (multiple write ports need mux in front of latch)
- Expensive to add many ports

6-Transistor SRAM (Static RAM)

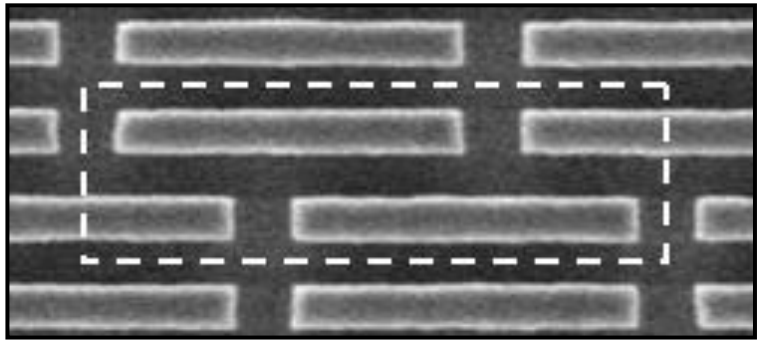


- Large on-chip memories built from arrays of static RAM bitcells, where each bit cell holds a bistable (cross-coupled inverters) and two access transistors.
- Other clocking and access logic factored out into periphery

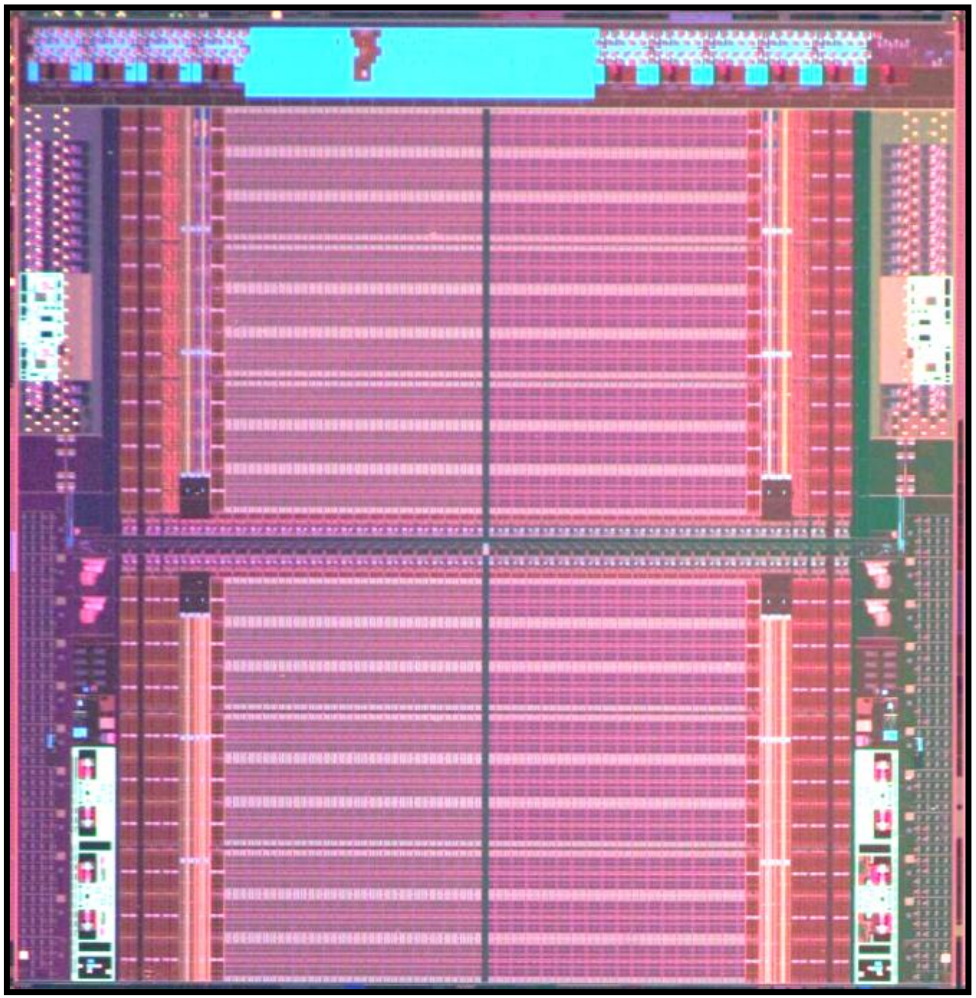
Intel's 22nm SRAM cell



0.092 um^2 SRAM cell
for high density applications

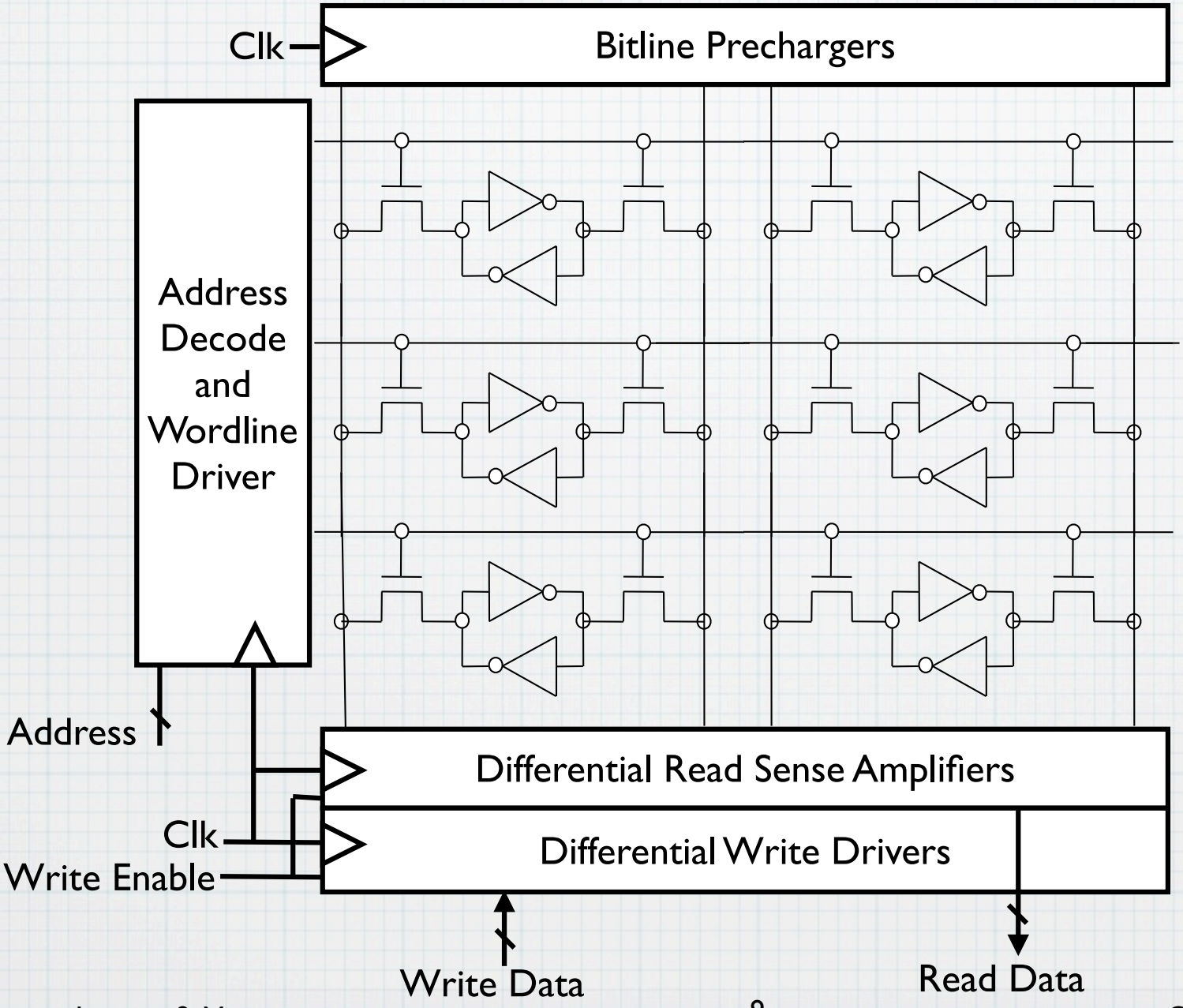


0.108 um^2 SRAM cell
for low voltage applications



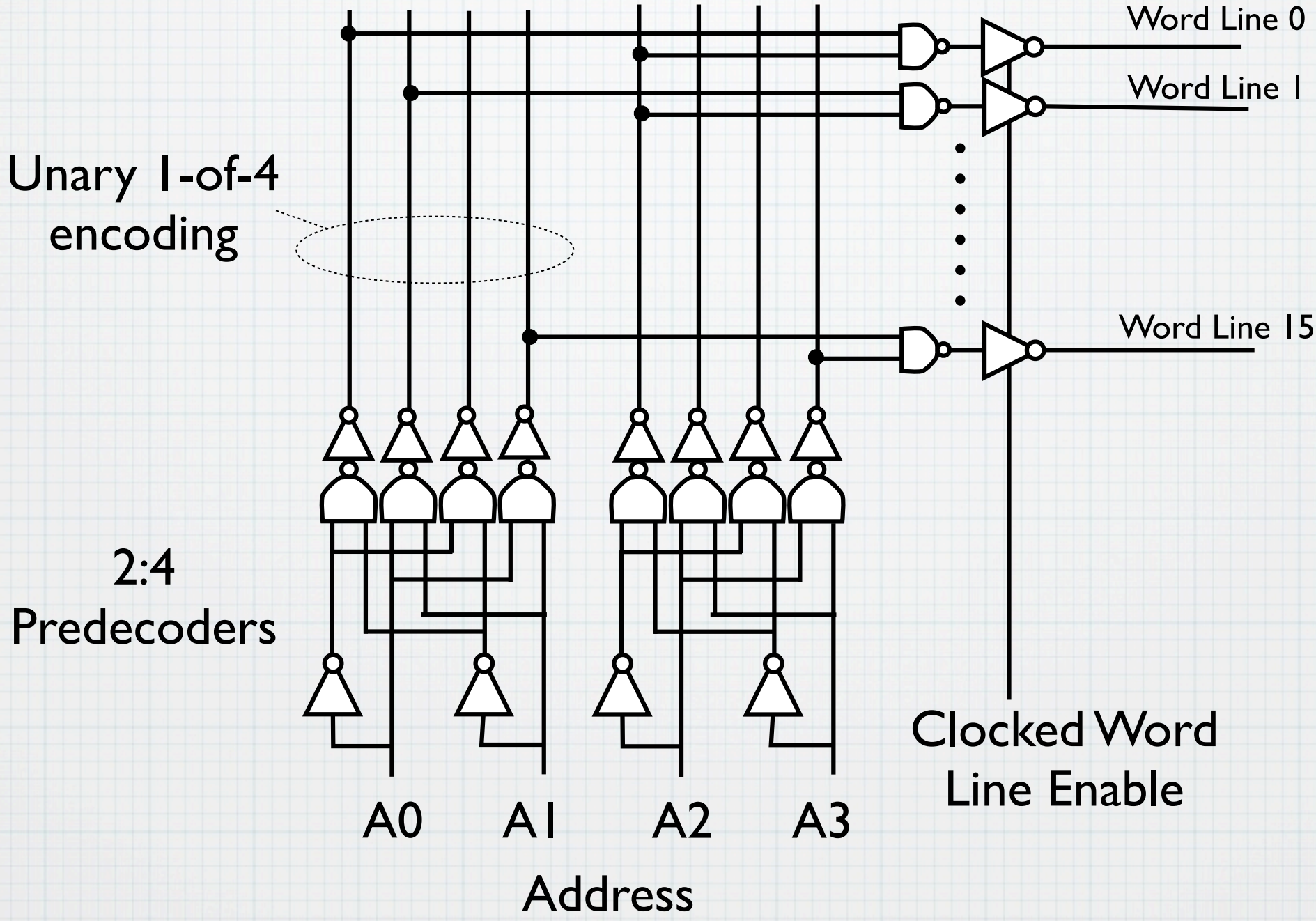
[Bohr, Intel, Sept 2009]

General SRAM Structure

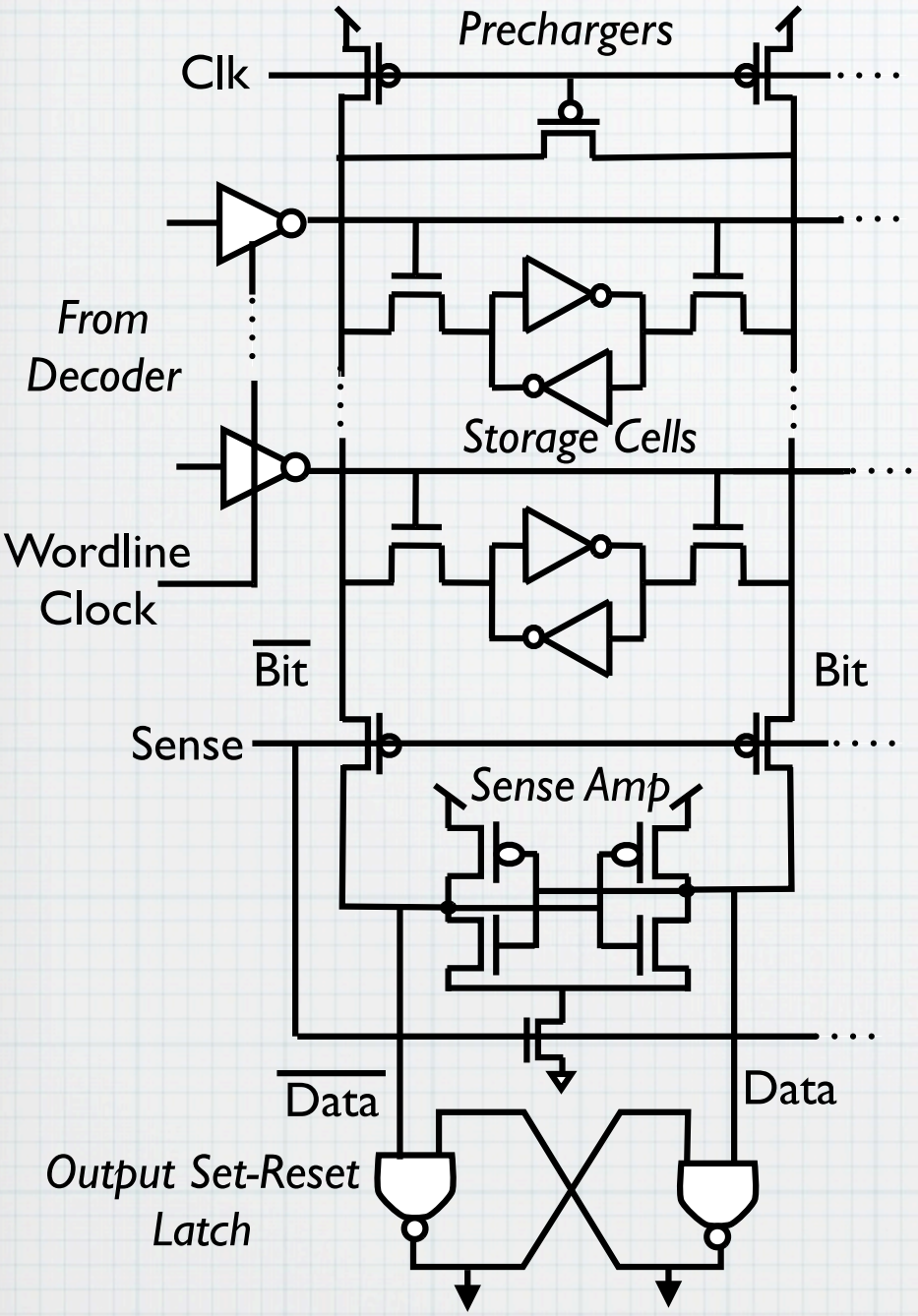


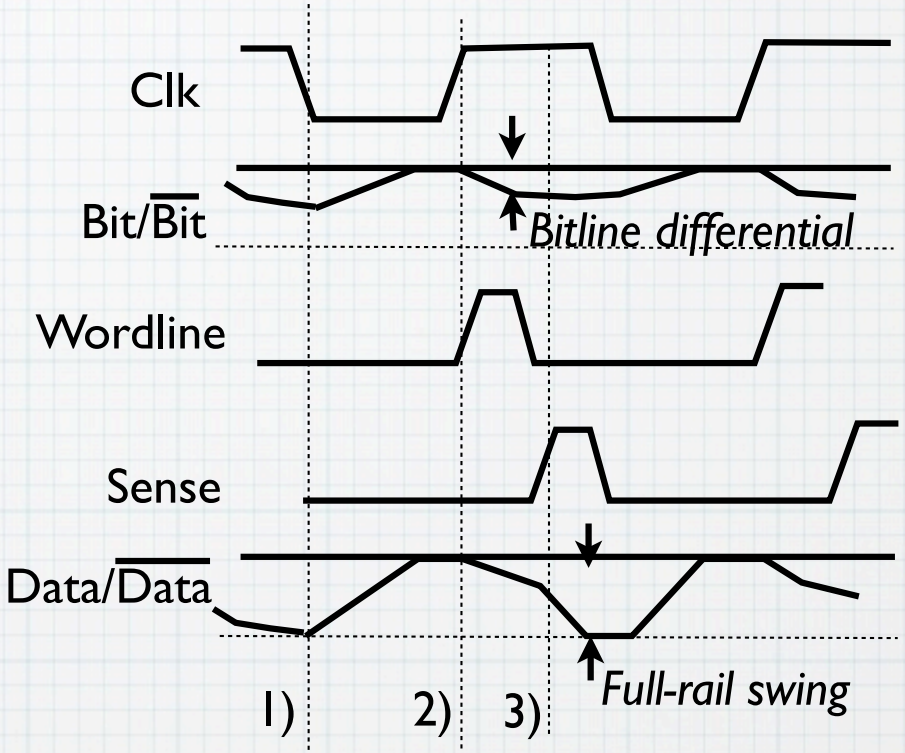
Usually maximum of 128-256 bits per row or column

Address Decoder Structure



Read Cycle





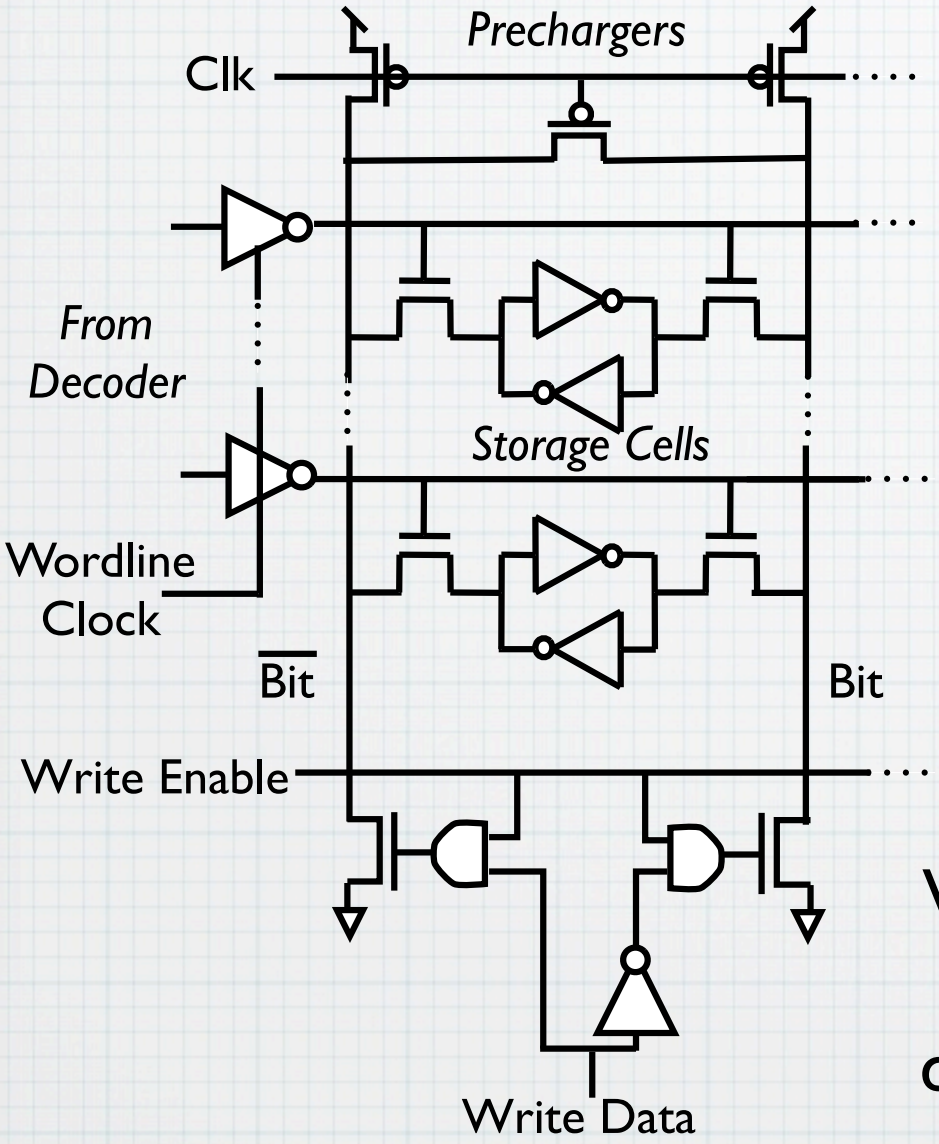
1) Precharge bitlines and senseamp

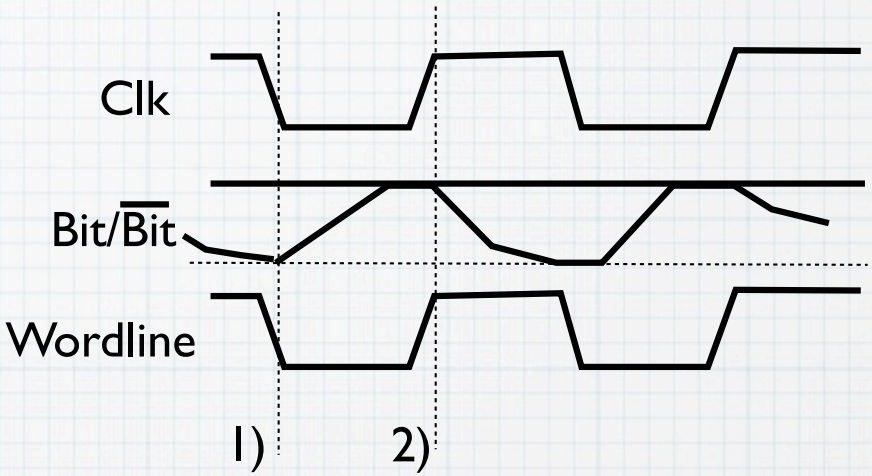
2) Pulse wordlines, develop bitline differential voltage

3) Disconnect bitlines from senseamp, activate sense pulldown, develop full-rail data signals

Pulses generated by internal self-timed signals, often using "replica" circuits representing critical paths

Write Cycle



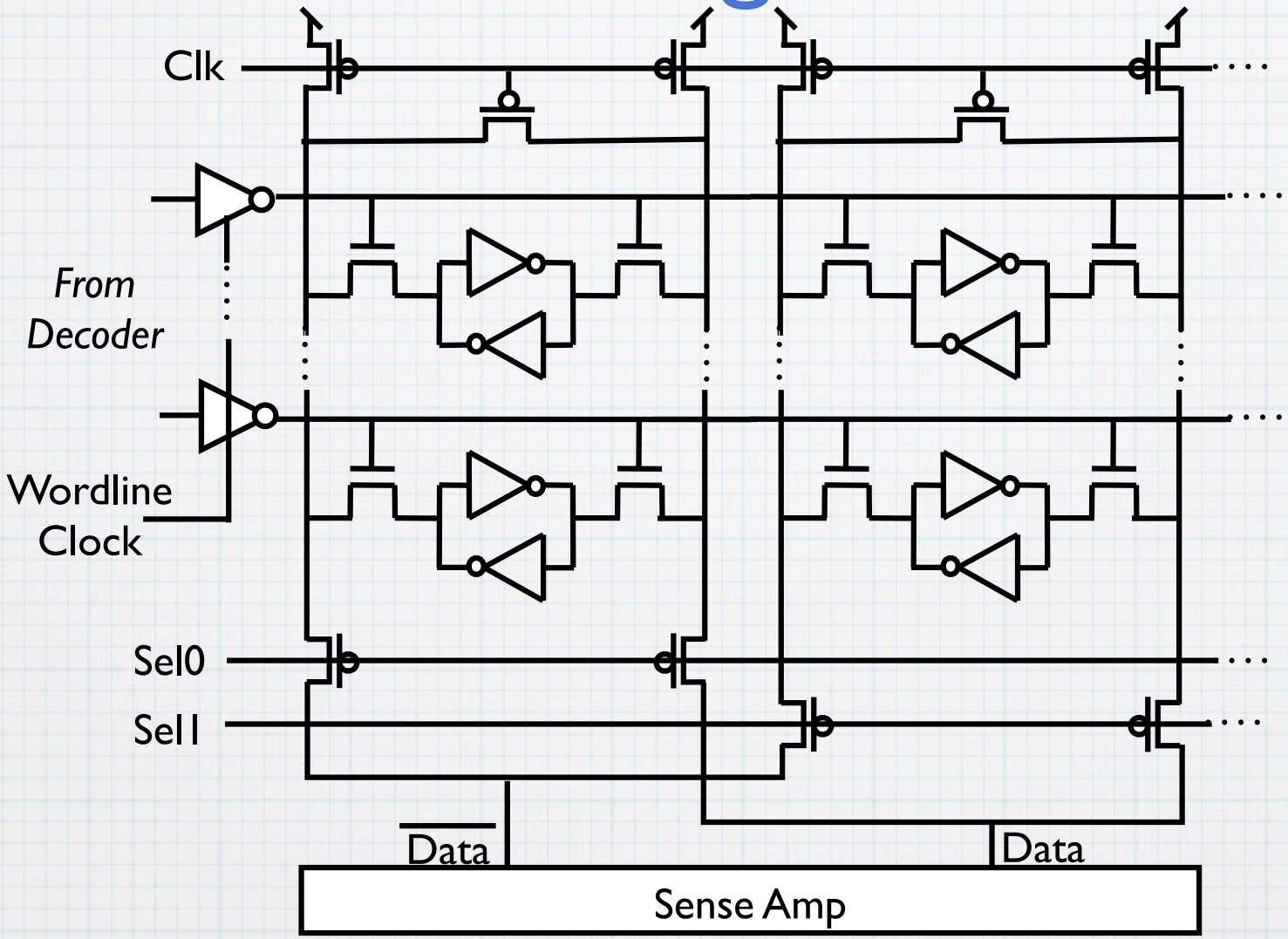


1) Precharge bitlines

2) Open wordline, pull down one bitline full rail

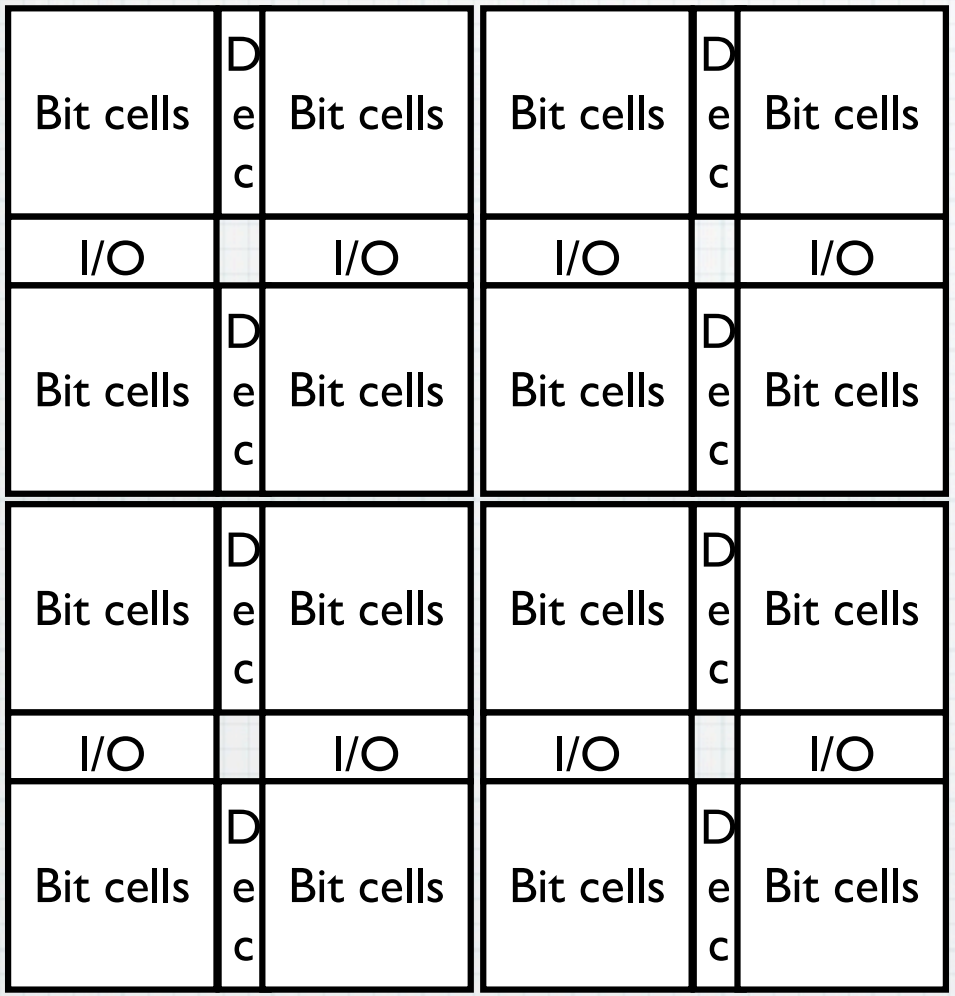
Write-enable can be controlled on a per-bit level. If bit lines not driven during write, cell retains value (looks like a read to the cell).

Column-Muxing at Sense Amps



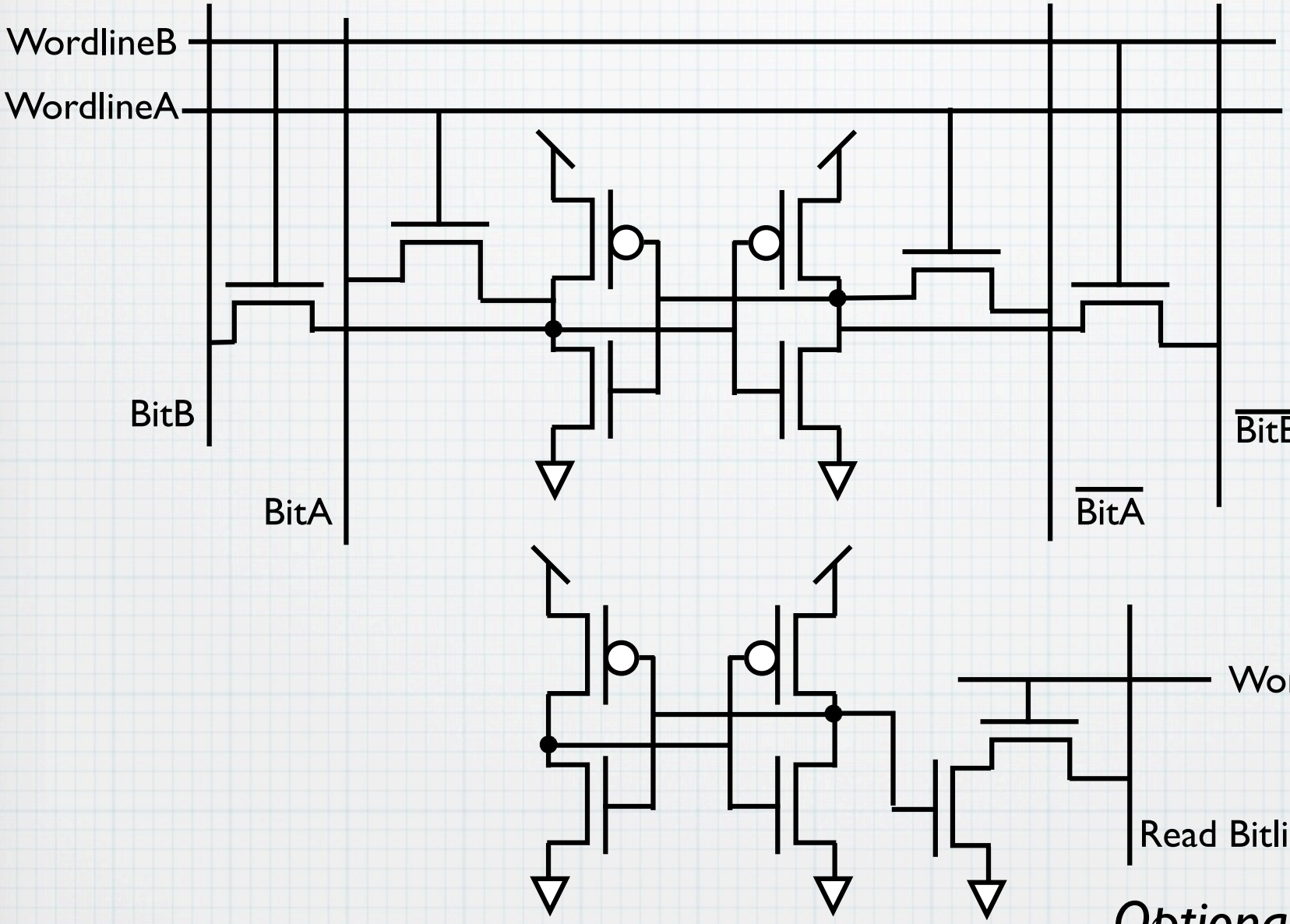
Difficult to pitch match sense amp to tight SRAM bit cell spacing so often 2-8 columns share one sense amp. Impacts power dissipation as multiple bitline pairs swing for each bit read.

Building Larger Memories



- Large arrays constructed by tiling multiple leaf arrays, sharing decoders and I/O circuitry
  - e.g., sense amp attached to arrays above and below
- Leaf array limited in size to 128-256 bits in row/column due to RC delay of wordlines and bitlines
- Also to reduce power by only activating selected sub-bank
- In larger memories, delay and energy dominated by I/O wiring

Adding More Ports



Differential Read or Write ports

Optional Single-ended Read port

Memory Compilers

- In ASIC flow, memory compilers used to generate layout for SRAM blocks in design
  - Often hundreds of memory instances in a modern SoC
  - Memory generators can also produce built-in self-test (BIST) logic, to speed manufacturing testing, and redundant rows/ columns to improve yield
- Compiler can be parameterized by number of words, number of bits per word, desired aspect ratio, number of sub banks, degree of column muxing, etc.
  - Area, delay, and energy consumption complex function of design parameters and generation algorithm
  - Worth experimenting with design space
- Usually only single read *or* write port SRAM and one read *and* one write SRAM generators in ASIC library

Small Memories

- Compiled SRAM arrays usually have a high overhead due to peripheral circuits, BIST, redundancy.
- Small memories are usually built from latches and/or flip-flops in a stdcell flow
- Cross-over point is usually around 1K bits of storage
  - Should try design both ways

Memory Design Patterns

Multiport Memory Design Patterns

Often we require multiple access ports to a common memory

- True Multiport Memory
  - As describe earlier in lecture, completely independent read and write port circuitry
- Banked Multiport Memory
  - Interleave lesser-ported banks to provide higher bandwidth
- Stream-Buffered Multiport Memory
  - Use single wider access port to provide multiple narrower streaming ports
- Cached Multiport Memory
  - Use large single-port main memory, but add cache to service

True Multiport Memory

Problem: Require simultaneous read and write access by multiple independent agents to a shared common memory.

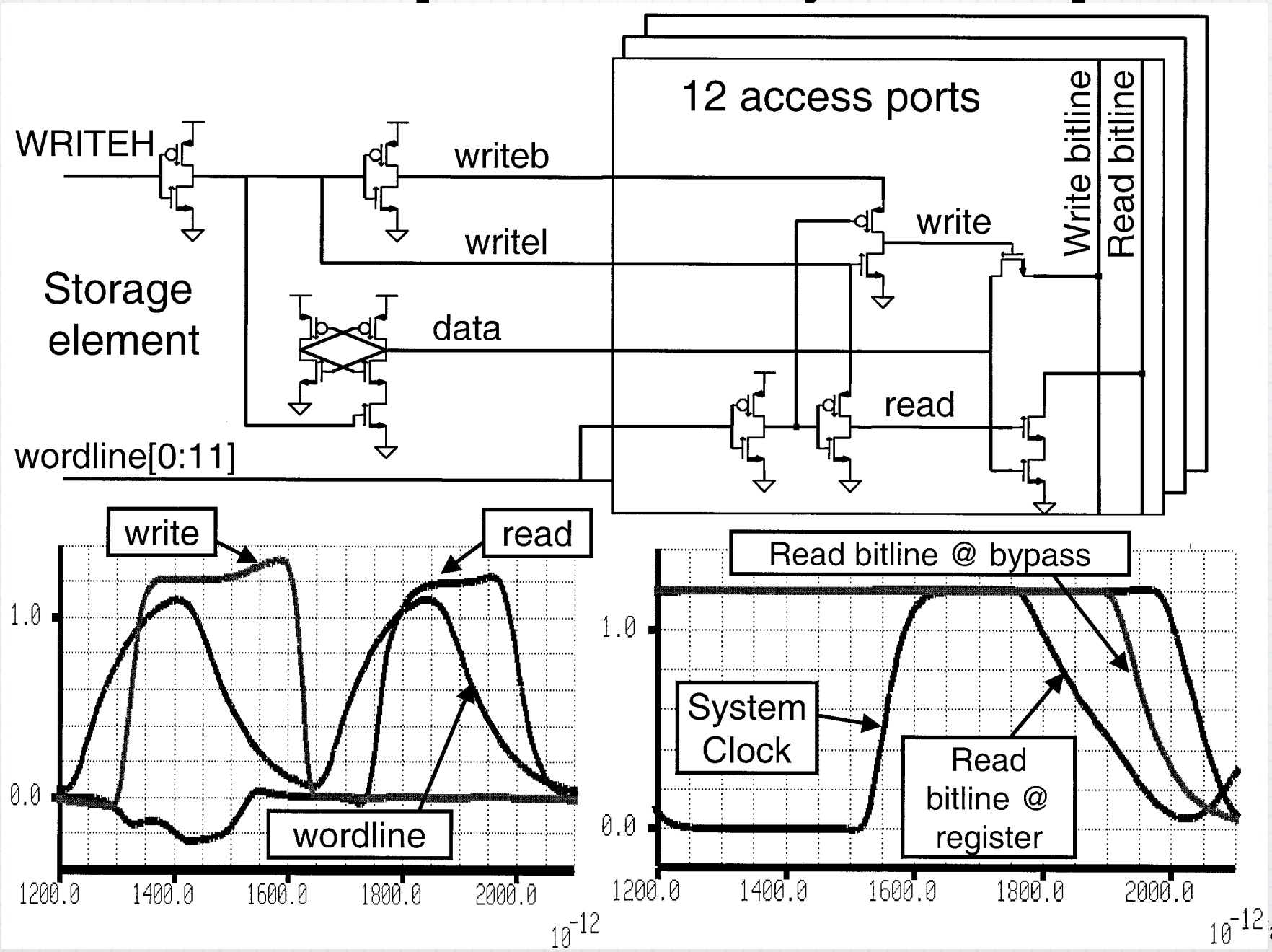
Solution: Provide separate read and write ports to each bit cell for each requester

Applicability: Where unpredictable access latency to the shared memory cannot be tolerated.

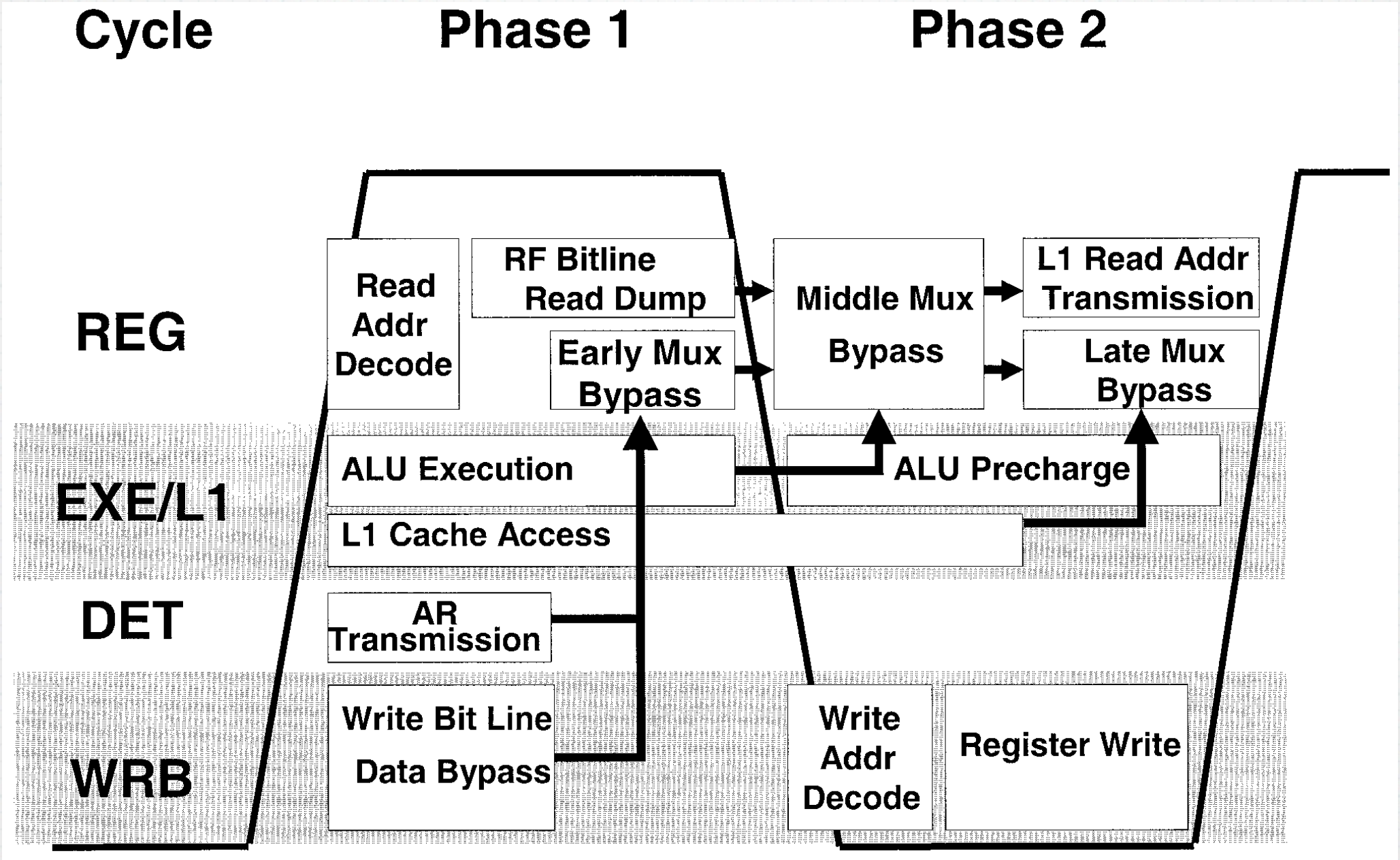
Consequences: High area, energy, and delay cost for large number of ports. Must define behavior when multiple writes on same cycle to same word (e.g., prohibit, provide priority, or combine writes).

True Multiport Example: Itanium-2 Regfile

- Intel Itanium-2 [Fetzer et al, IEEE JSSCC 2002]



Itanium-2 Regfile Timing



Banked Multiport Memory

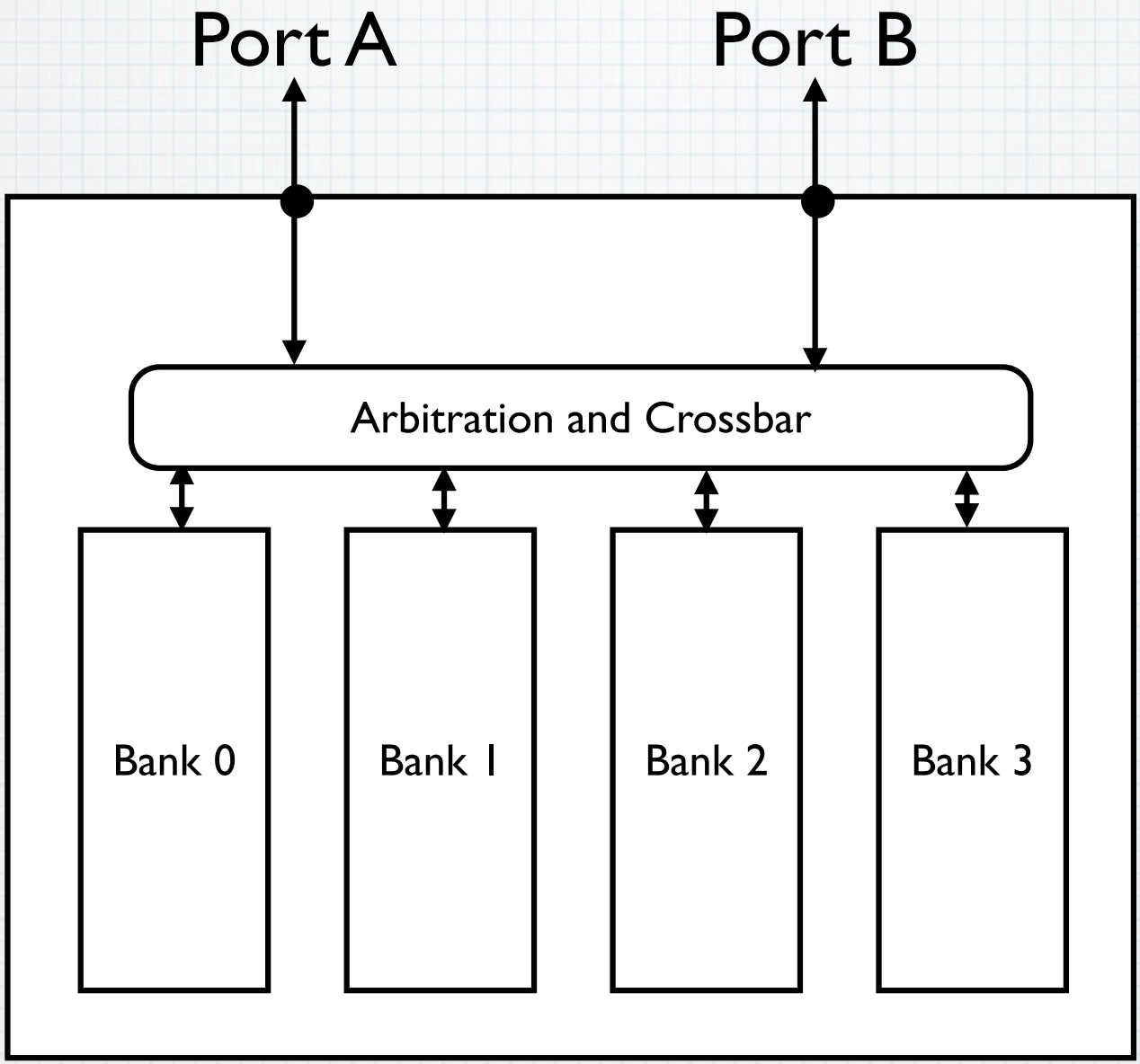
Problem: Require simultaneous read and write access by multiple independent agents to a large shared common memory.

Solution: Divide memory capacity into smaller banks, each of which has fewer ports. Requests are distributed across banks using a fixed hashing scheme. Multiple requesters arbitrate for access to same bank/port.

Applicability: Requesters can tolerate variable latency for accesses. Accesses are distributed across address space so as to avoid "hotspots".

Consequences: Requesters must wait arbitration delay to determine if request will complete. Have to provide interconnect between each requester and each bank/port. Can have greater, equal, or lesser number of banks*ports/bank compared to total number of external access ports.

Banked Multiport Memory



Banked Multiport Memory Example

Pentium (P5) 8-way interleaved data cache, with two ports

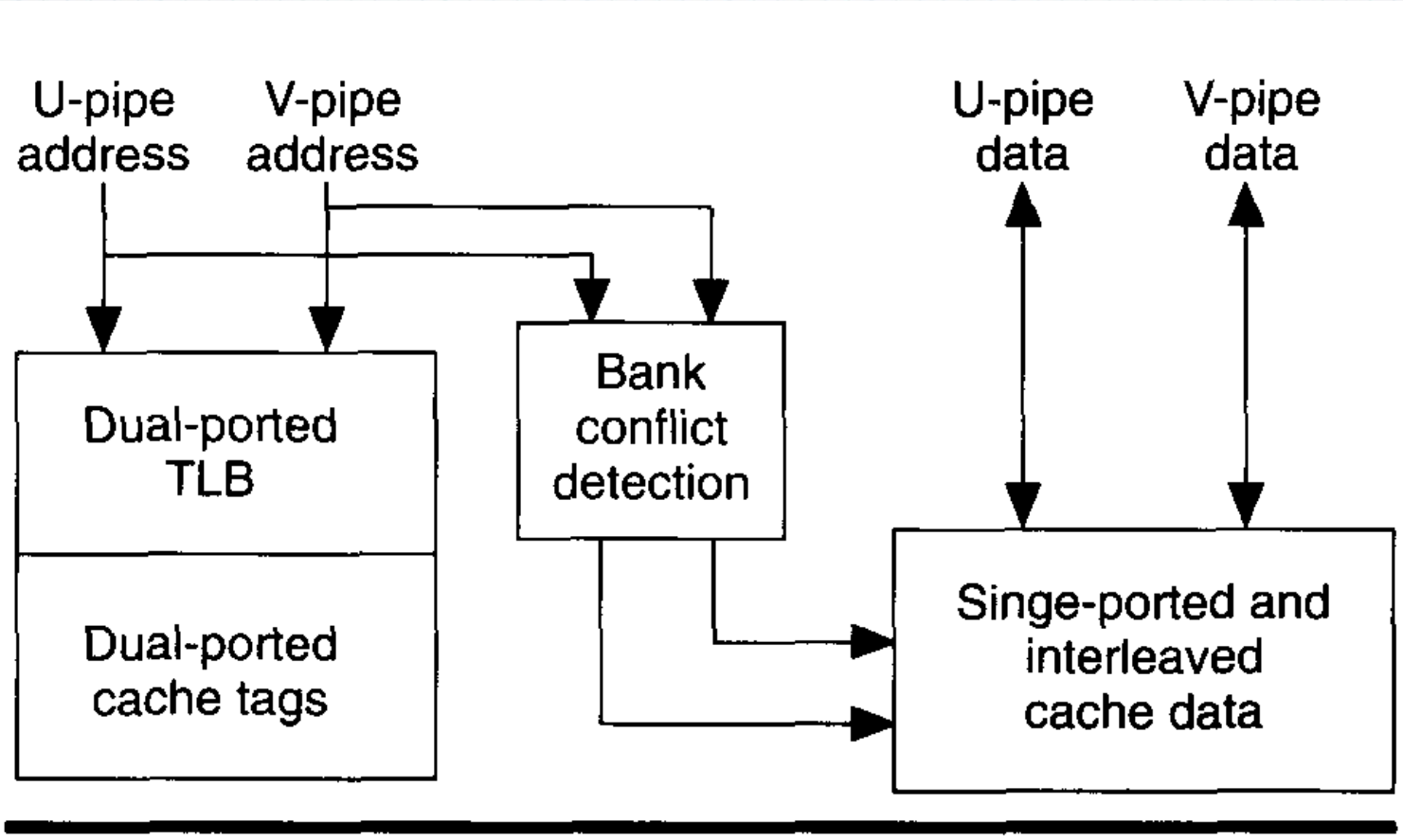


Figure 7. Dual-access data cache.

[Alpert et al, IEEE Micro, May 1993]

Stream-Buffered Multiport Memory

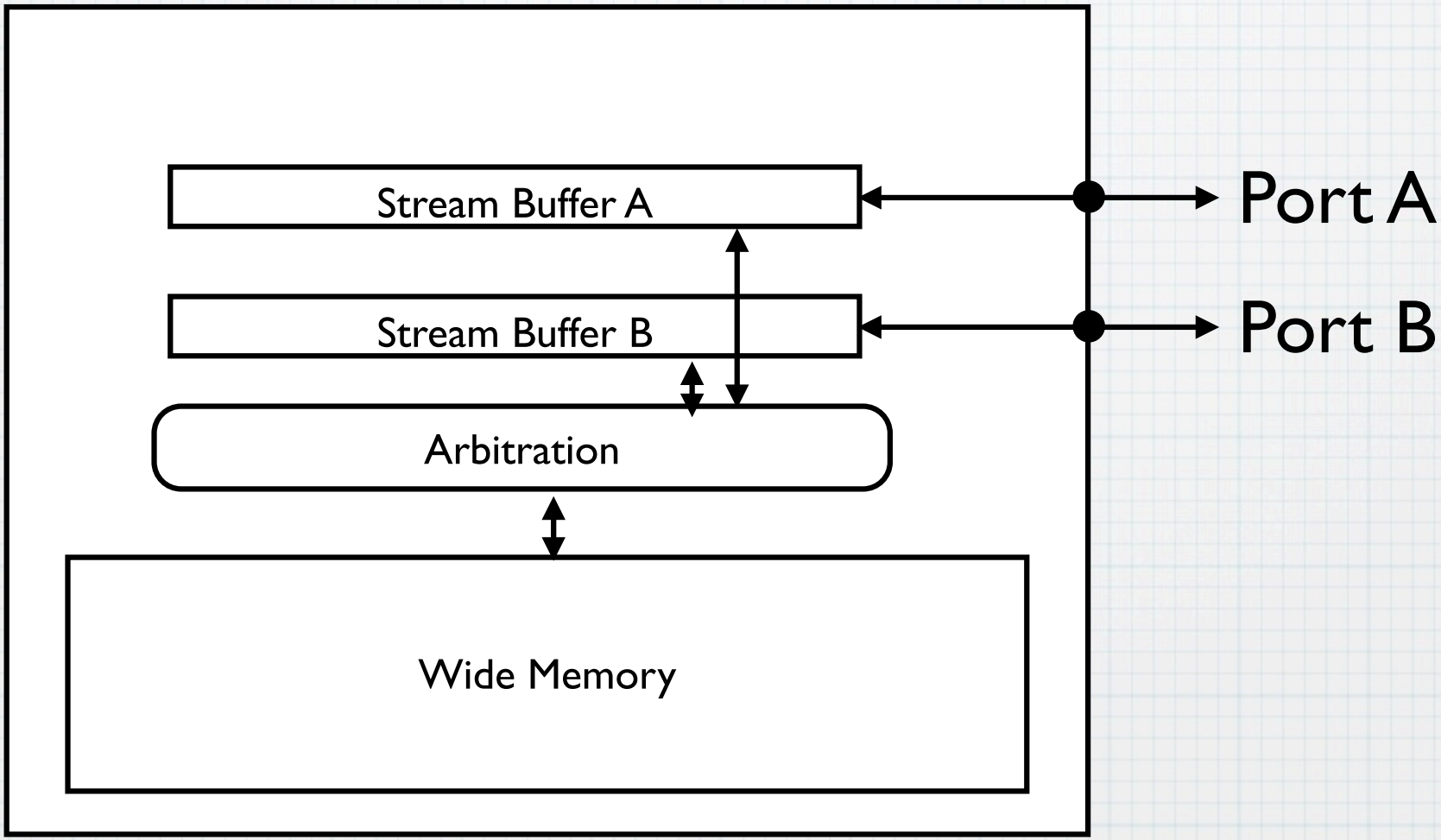
Problem: Require simultaneous read and write access by multiple independent agents to a large shared common memory, where each requester usually makes multiple sequential accesses.

Solution: Organize memory to have a single wide port. Provide each requester with an internal stream buffer that holds width of data returned/consumed by each memory access. Each requester can access own stream buffer without contention, but arbitrates with others to read/write stream buffer from memory.

Applicability: Requesters make mostly sequential requests and can tolerate variable latency for accesses.

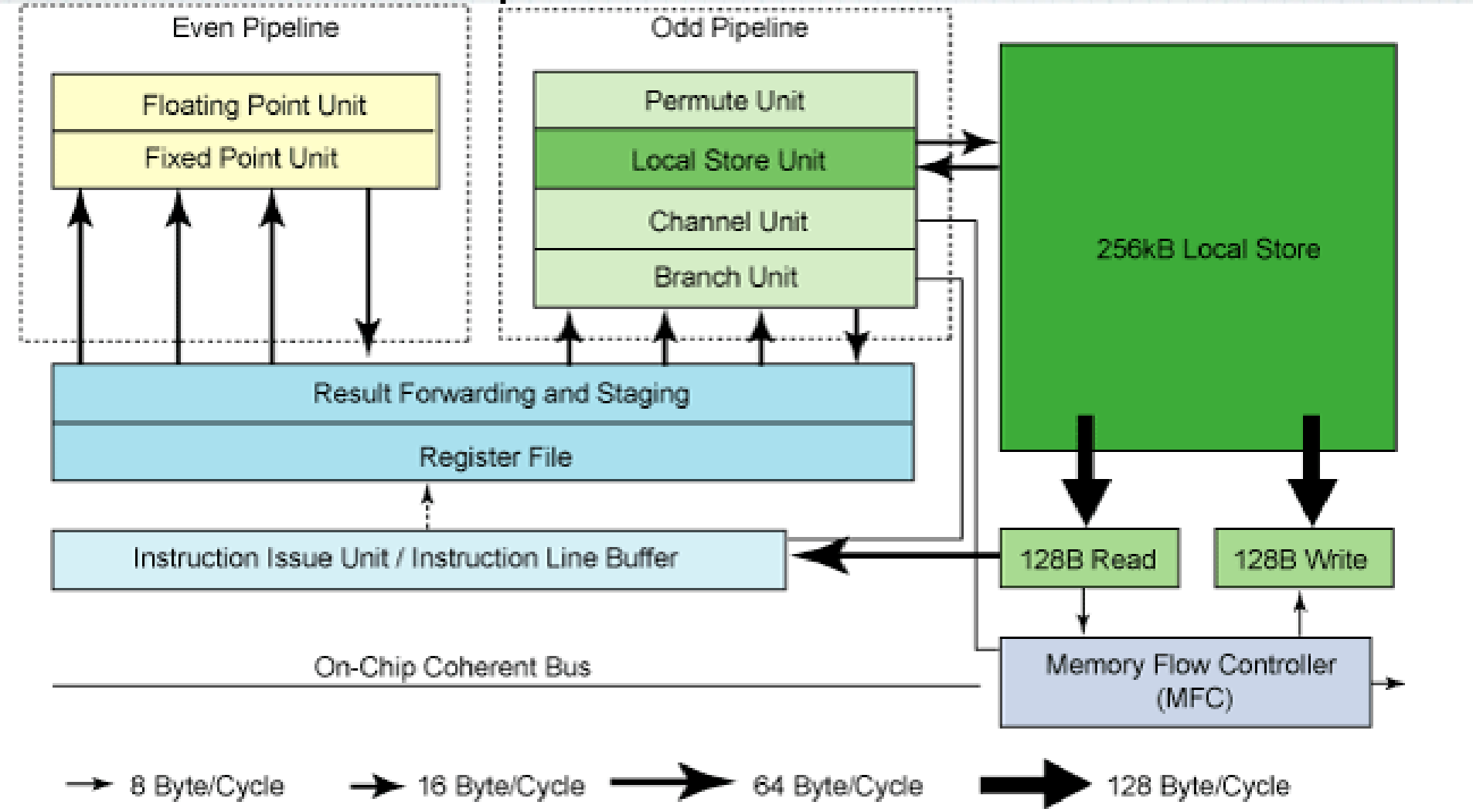
Consequences: Requesters must wait arbitration delay to determine if request will complete. Have to provide stream buffers for each requester. Need sufficient access width to serve aggregate bandwidth demands of all requesters, but wide data access can be wasted if not all used by requester. Have to specify memory consistency model between ports (e.g., provide stream flush operations).

Stream-Buffered Multiport Memory



Stream-Buffered Multiport Examples

- IBM Cell microprocessor local store



[Chen et al., IBM, 2005]

Cached Multiport Memory

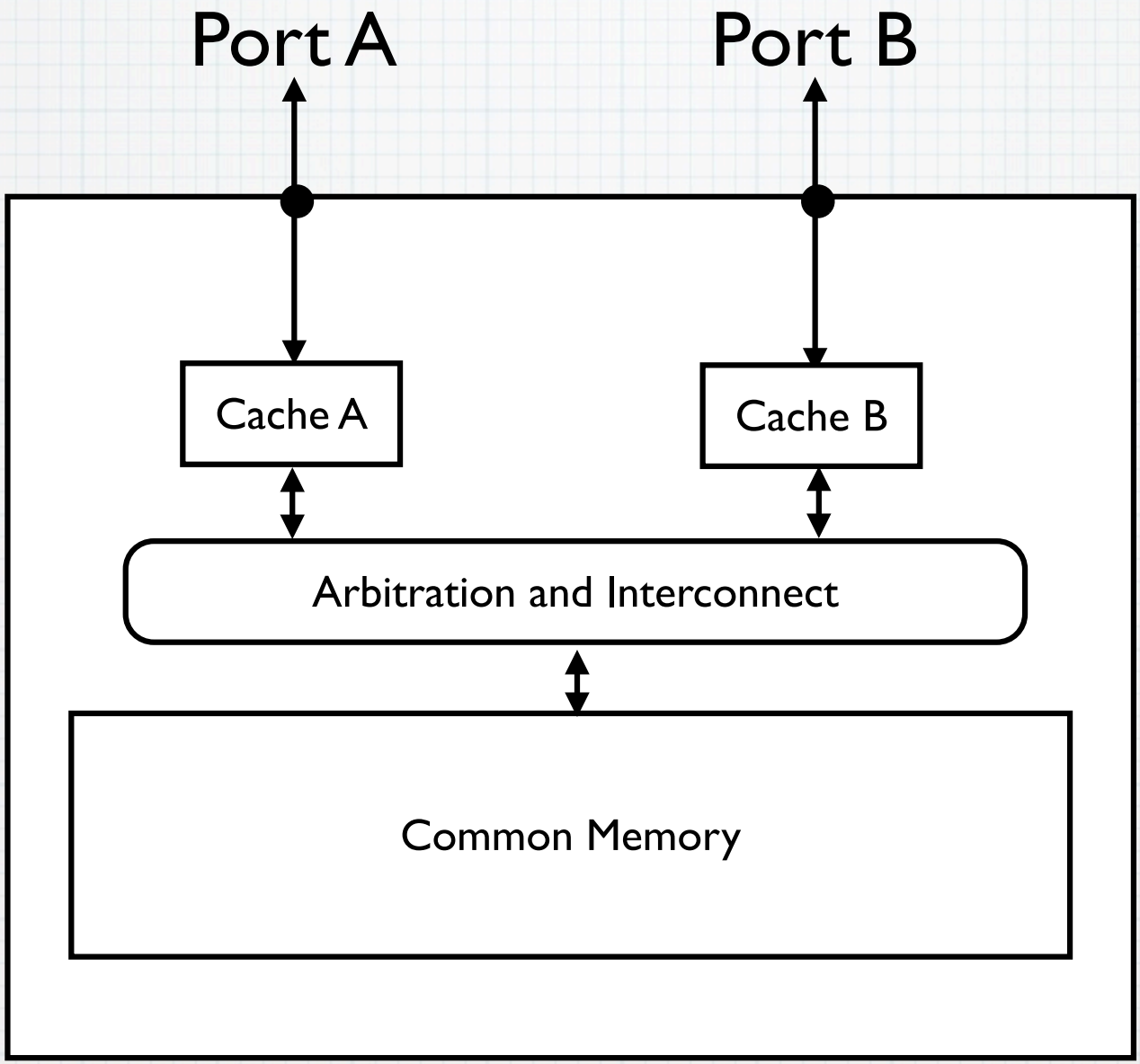
Problem: Require simultaneous read and write access by multiple independent agents to a large shared common memory.

Solution: Provide each access port with a local cache of recently touched addresses from common memory, and use a cache coherence protocol to keep the cache contents in sync.

Applicability: Request streams have significant temporal locality, and limited communication between different ports.

Consequences: Requesters will experience variable delay depending on access pattern and operation of cache coherence protocol. Tag overhead in both area, delay, and energy/access. Complexity of cache coherence protocol.

Cached Multiport Memory



Replicated-State Multiport Memory

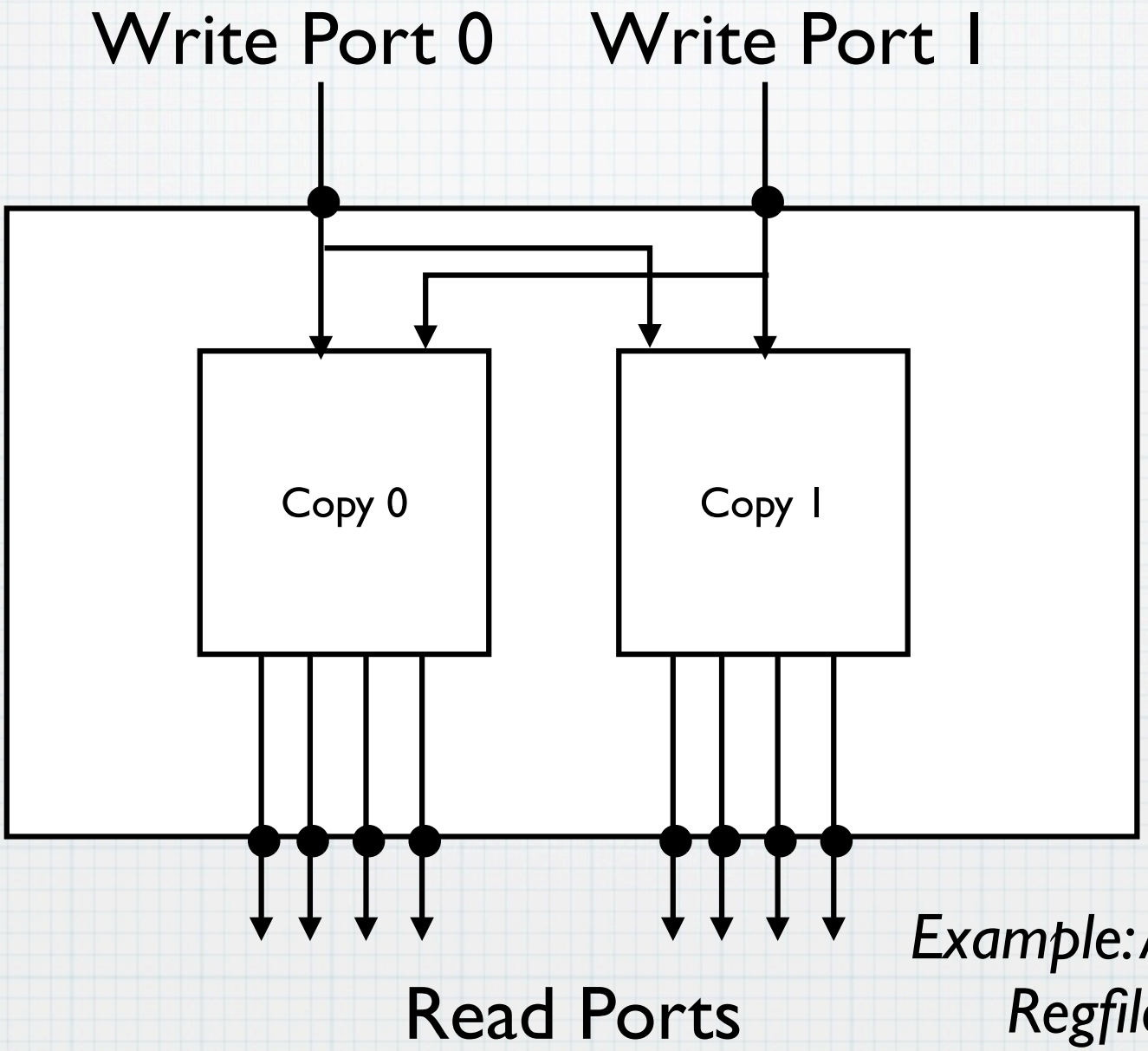
Problem: Require simultaneous read and write access by multiple independent agents to a small shared common memory. Cannot tolerate variable latency of access.

Solution: Replicate storage and divide read ports among replicas. Each replica has enough write ports to keep all replicas in sync.

Applicability: Many read ports required, and variable latency cannot be tolerated.

Consequences: Potential increase in latency between some writers and some readers.

Replicated-State Multiport Memory



Example: Alpha 21264
Regfile clusters

Memory Hierarchy Design Patterns

Use small fast memory together large slow memory to provide illusion of large fast memory.

- Explicitly managed local stores
- Automatically managed cache hierarchies