EECS 151/251A

Spring 2021

Digital Design and Integrated Circuits

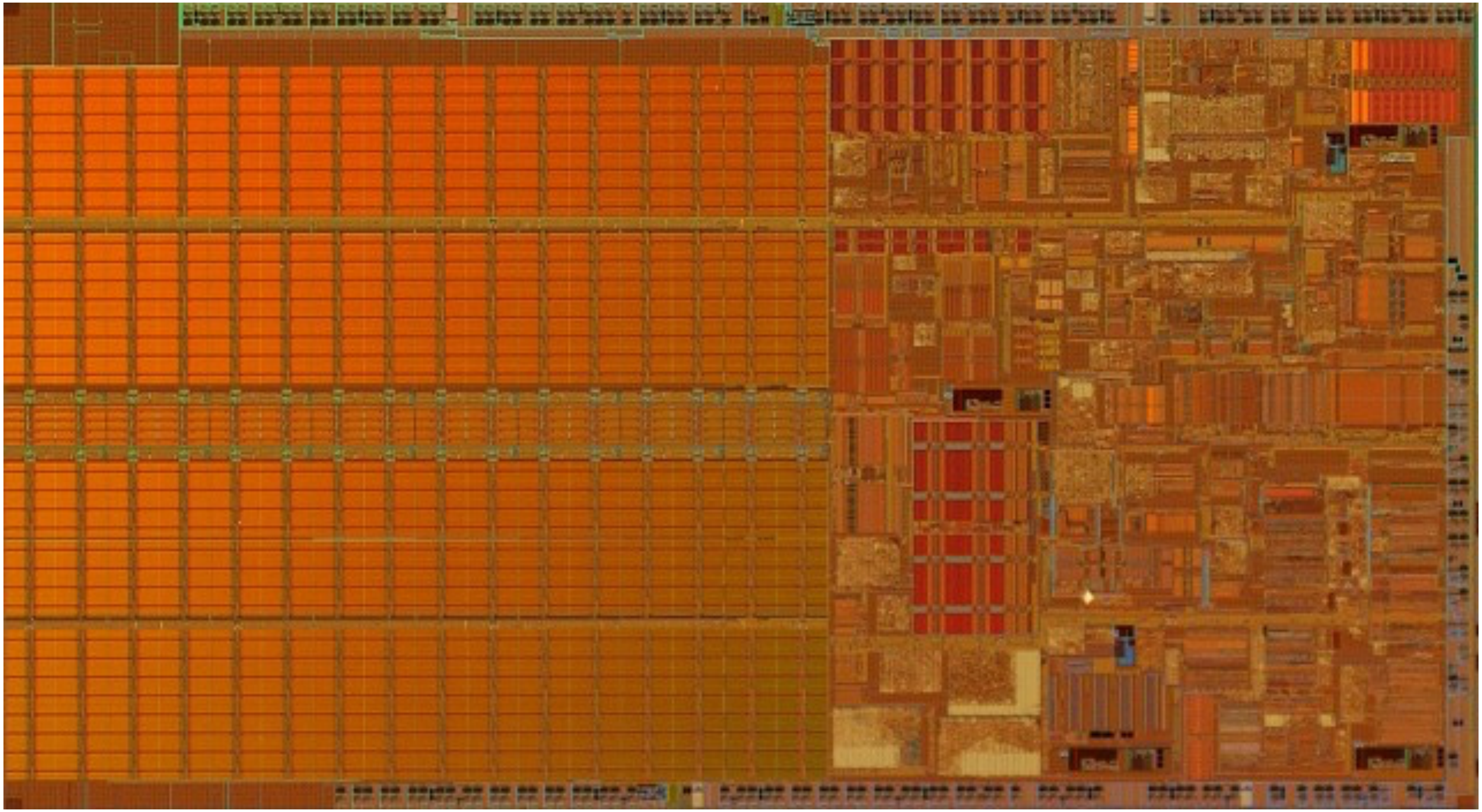
Instructor:
John Wawrzynek

Lecture 18: Memory Circuits and Blocks, Part 2

Announcements

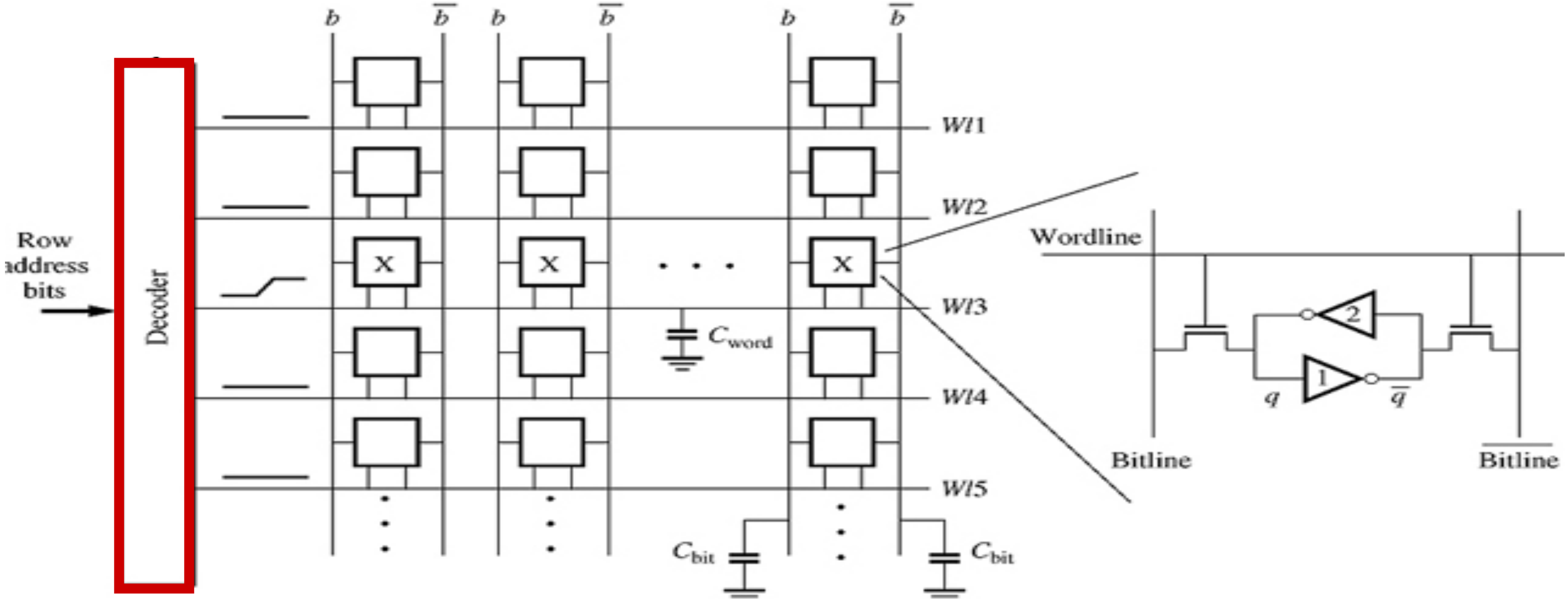
- Welcome back!
- Virtual Front Row for today 3/30:
  - *Bernard Chen*
  - *Matthew Tran*
  - *Jennifer Zhou*
  - *Suphakorn Lertruchtkul*
  - *Ruohan Yan*
- **Please ask question or make comments!**
- Homework assignment 7 (power & memory) posted.

(On-chip) SRAM Memory



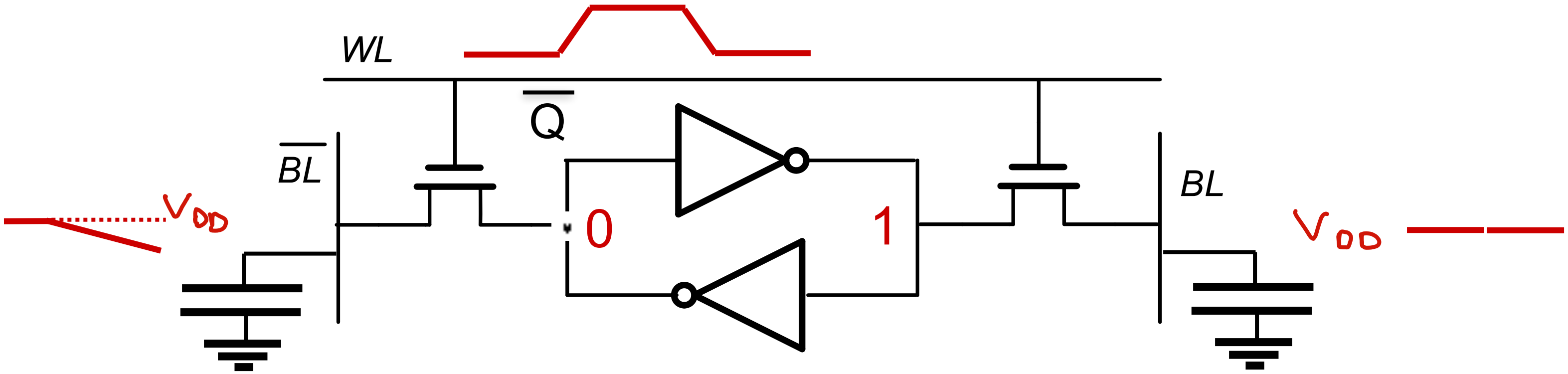
ARM A5 Photo

SRAM read/write operations



SRAM Operation - Read

1. Bit lines are "pre-charged" to VDD
2. Word line is driven high (pre-charger is turned off)
3. Cell pulls-down one bit line
4. Differential sensing circuit on periphery is activated to capture value on bit lines.

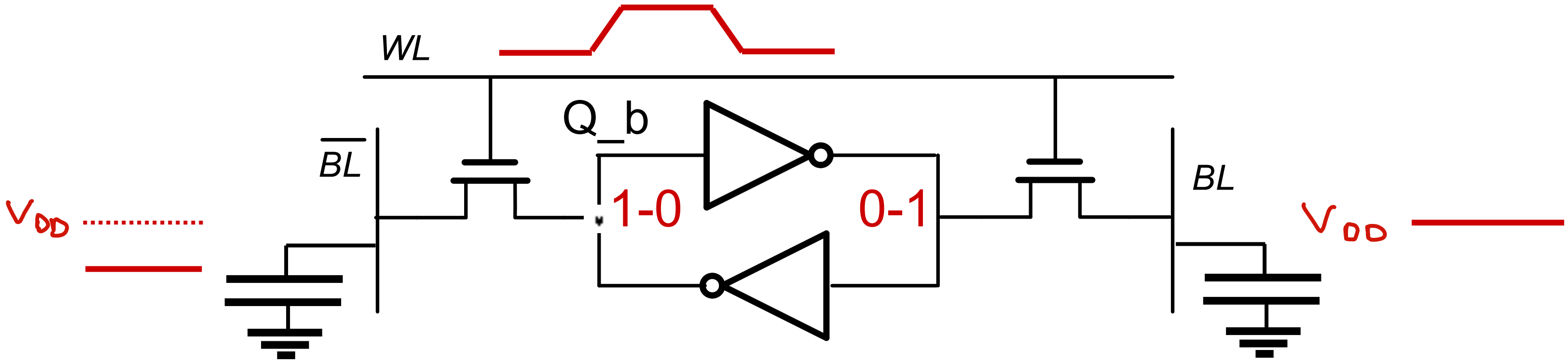


During read $\overline{Q}$ will get slightly pulled up when WL first goes high, but …

- *But by sizing the transistors correctly, reading the cell will not destroy the stored value*

SRAM Operation - Write

1. Column driver circuit on periphery differentially drives the bit lines
2. Word line is driven high (column driver stays on)
3. One side of cell is driven low, flips the other side

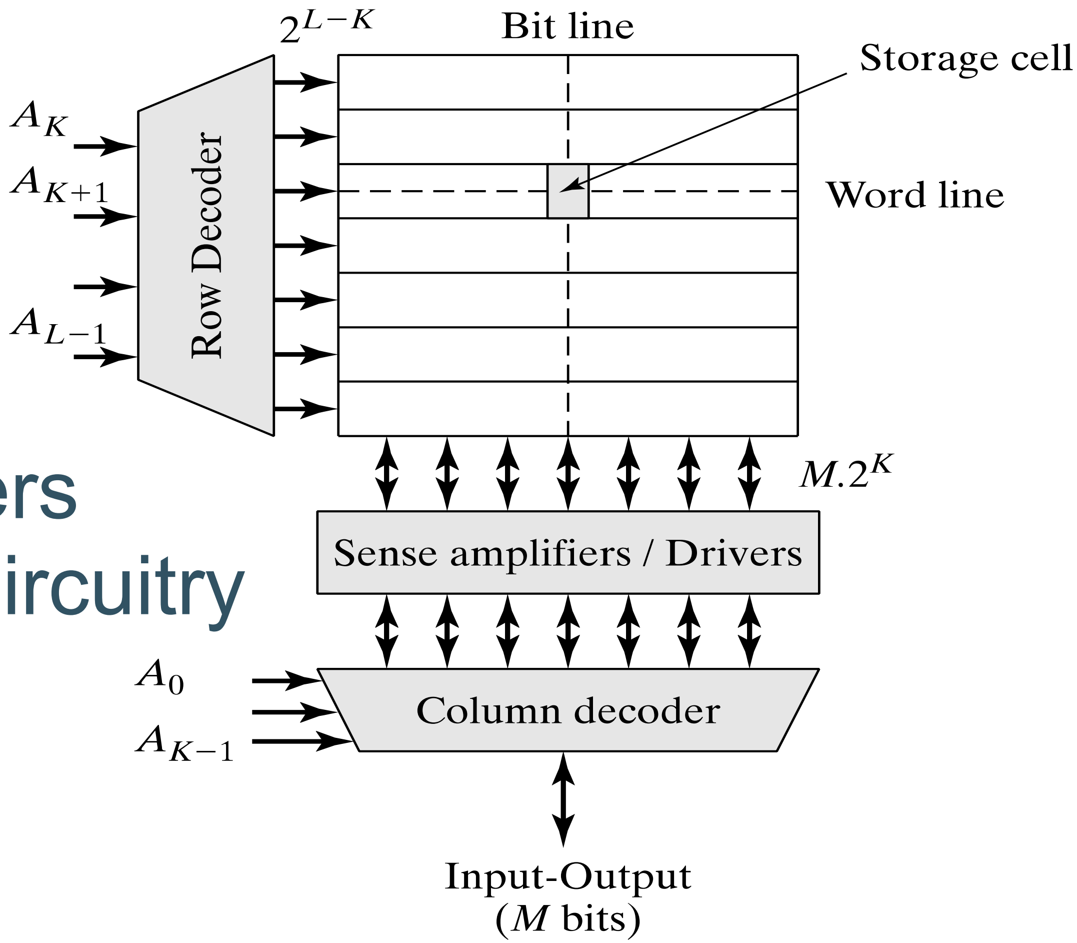


For successful write the access transistor needs to overpower the cell pullup. The transistors are sized to allow this to happen.

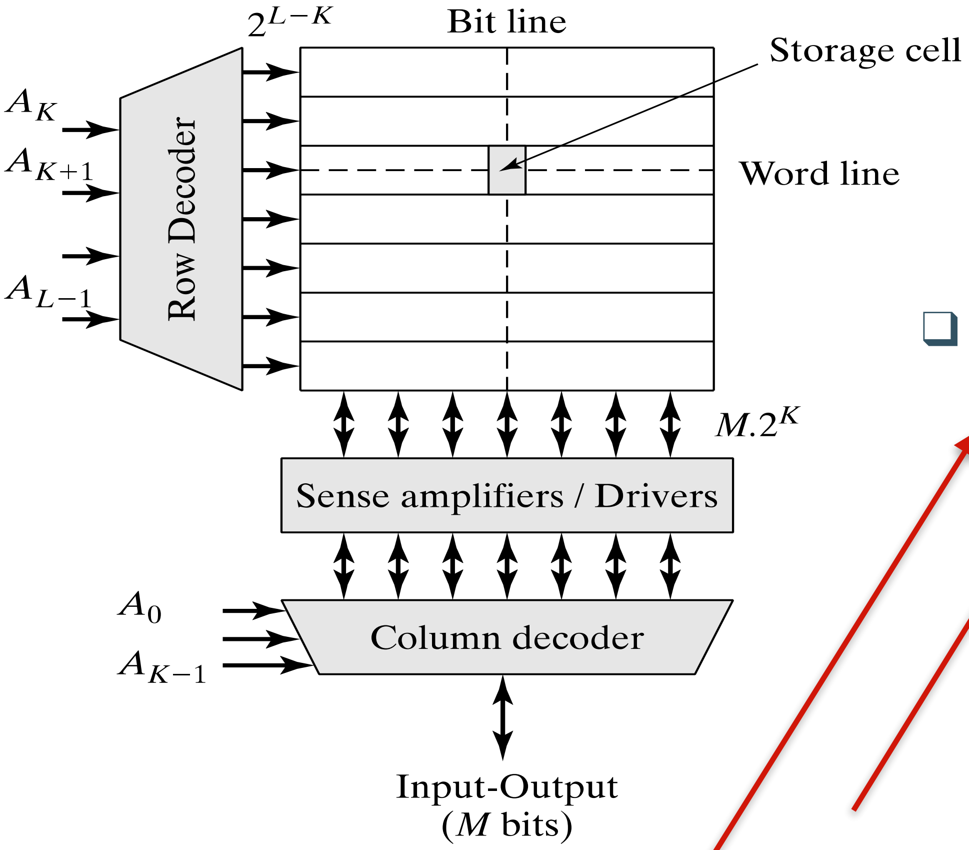
Memory Periphery

Periphery

- ❑ Decoders
- ❑ Sense Amplifiers
- ❑ Input/Output Buffers
- ❑ Control / Timing Circuitry



Row Decoder



- L total address bits
- K for column decoding
- L-K for row decoding
- Row decoder expands L-K address lines into 2^{L-K} word lines
- M bits per word

- ❑ Example: decoder for 8Kx8 memory block
  - ❑ core arranged as 256x256 cells
  - ❑ Need 256 AND gates, each driving one word line

each row has 32 8-bit words (8x32=256)

8K x 8 means 8K words of 8-bits each

In this case: L=13 total address bits (2^L=8K), K=5 (2^K=32), L-K=8 (2^{L-K}=256)

Row Decoders

(N)AND Decoder

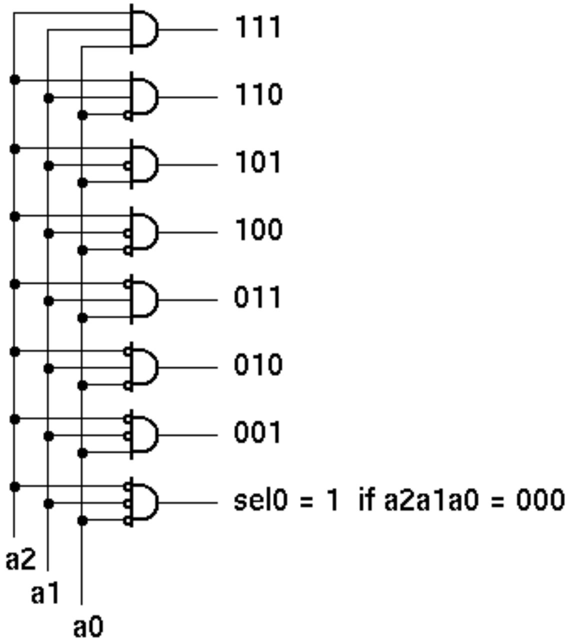
$$WL_0 = A_0A_1A_2A_3A_4A_5A_6A_7A_8A_9$$

$$WL_{511} = \bar{A}_0A_1A_2A_3A_4A_5A_6A_7A_8A_9$$

NOR Decoder

$$WL_0 = \overline{A_0 + A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8 + A_9}$$

$$WL_{511} = \overline{A_0 + \bar{A}_1 + \bar{A}_2 + \bar{A}_3 + \bar{A}_4 + \bar{A}_5 + \bar{A}_6 + \bar{A}_7 + \bar{A}_8 + \bar{A}_9}$$



Collection of 2^{L-K} logic gates, but need to be dense and fast.

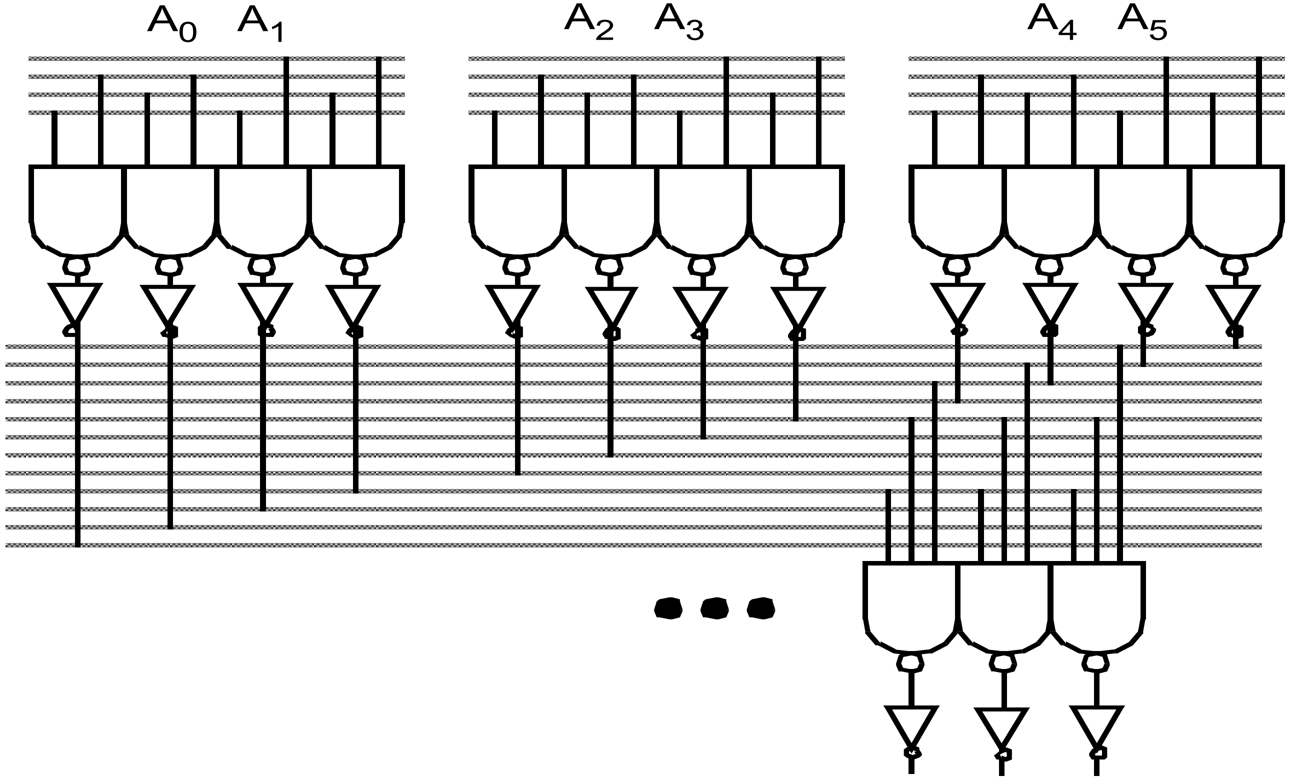
Naive solution would require L-K input gates: *Too big to pitch match to storage cells and too slow.*

Predecoders

$$\overline{a_5}\,\overline{a_4}\,\overline{a_3}\,\overline{a_2}\,\boxed{\overline{a_1}\,\overline{a_0}}$$
$$\overline{a_5}\,\overline{a_4}\,\overline{a_3}\,\overline{a_2}\,\overline{a_1}\,a_0$$
$$\overline{a_5}\,\overline{a_4}\,\overline{a_3}\,\overline{a_2}\,a_1\,\overline{a_0}$$
$$\overline{a_5}\,\overline{a_4}\,\overline{a_3}\,\overline{a_2}\,a_1\,a_0$$
$$\overline{a_5}\,\overline{a_4}\,\overline{a_3}\,a_2\,\boxed{\overline{a_1}\,\overline{a_0}}$$
$$\overline{a_5}\,\overline{a_4}\,\overline{a_3}\,a_2\,\overline{a_1}\,a_0$$
$$\overline{a_5}\,\overline{a_4}\,\overline{a_3}\,a_2\,a_1\,\overline{a_0}$$
$$\overline{a_5}\,\overline{a_4}\,\overline{a_3}\,a_2\,a_1\,a_0$$
$$\vdots$$
$$a_5\,a_4\,a_3\,a_2\,\boxed{\overline{a_1}\,\overline{a_0}}$$
$$a_5\,a_4\,a_3\,a_2\,\overline{a_1}\,a_0$$
$$a_5\,a_4\,a_3\,a_2\,a_1\,\overline{a_0}$$
$$a_5\,a_4\,a_3\,a_2\,a_1\,a_0$$

- Use a single gate for each of the shared terms
  - E.g., from $a_1, \overline{a_1}, a_0, \overline{a_0}$ generate four signals:
  - $\overline{a_1}\,\overline{a_0}\,, \overline{a_1}\,a_0\,, a_1\,\overline{a_0}\,, a_1\,a_0$
- In other words, we decode smaller groups of address bits first
  - And using the “predecoded” outputs to do the rest of the decoding

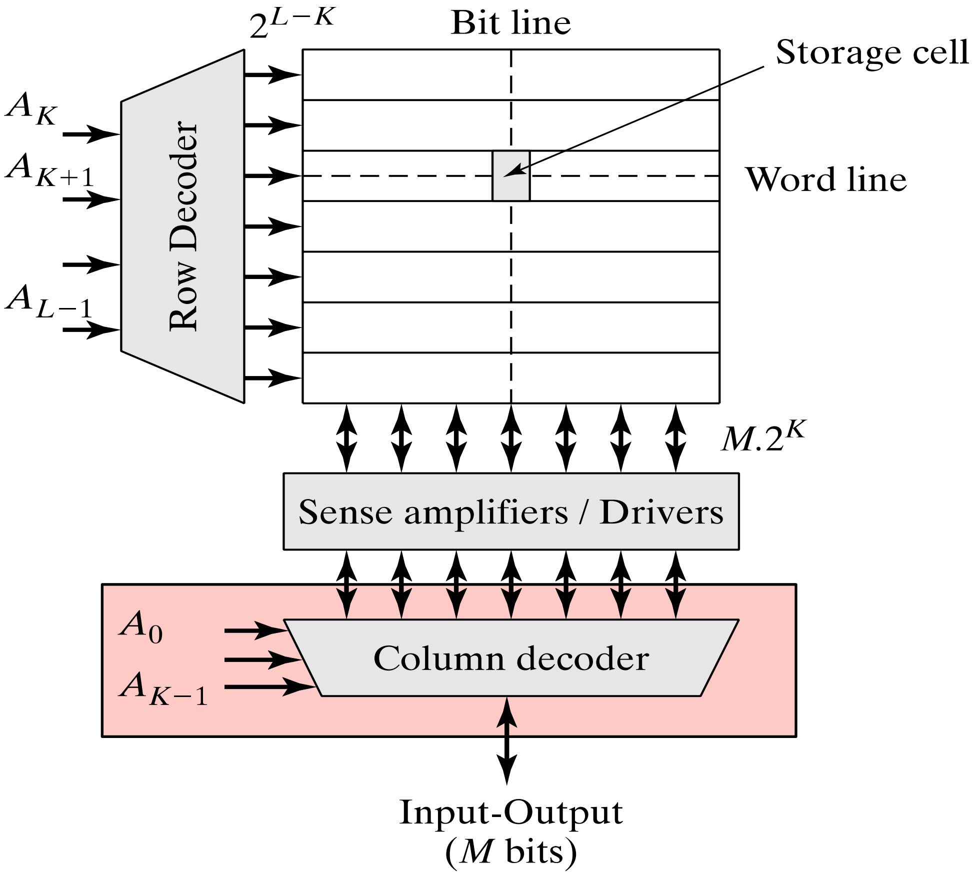
Predecoder and Decoder



Predecoders

Final Decoder

Column "Decoder"

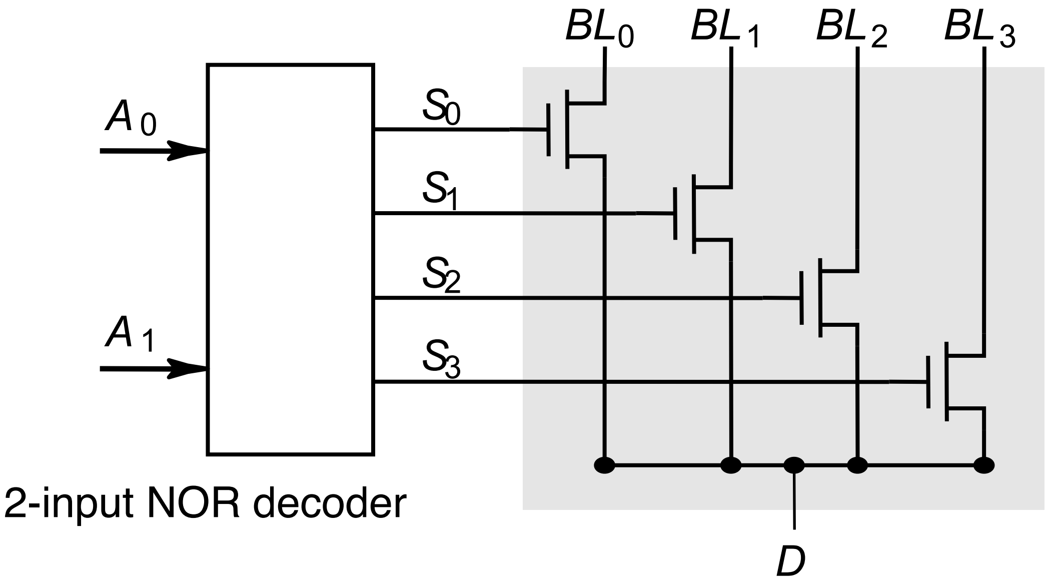


- Is basically a multiplexer
- Each row contains 2^K words each M bits wide.
- Bit of each of the 2^K are interleaved
  - ex: K=2, M=8

$$d_7c_7b_7a_7d_6c_6b_6a_6d_5c_5b_5a_5d_4c_4b_4a_4d_3c_3b_3a_3d_2c_2b_2a_2d_1c_1b_1a_1d_0c_0b_0a_0$$

4 interleaved words A, B, C, D

4-input pass-transistor based Column Decoder (for read)



(actual circuit would use a "differential signaling")

decoder shared across all 2^{K} x M row bits

Advantages: speed (Only one extra transistor in signal path, share sense amp)

Sense Amplifiers Speed Reading

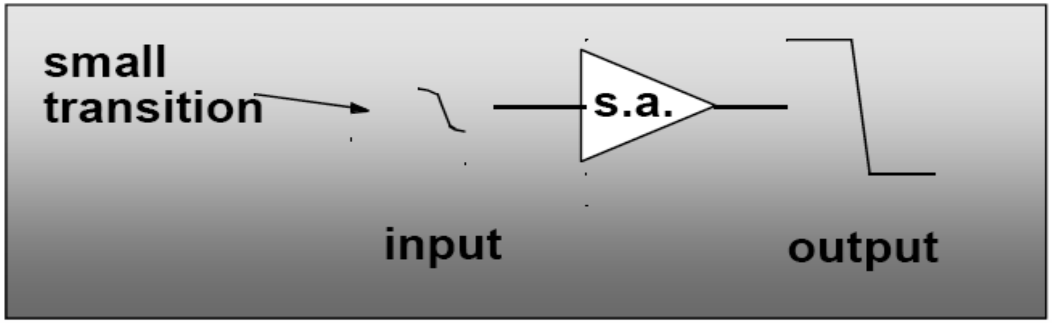
large Capacitance of bit lines

make as small as possible

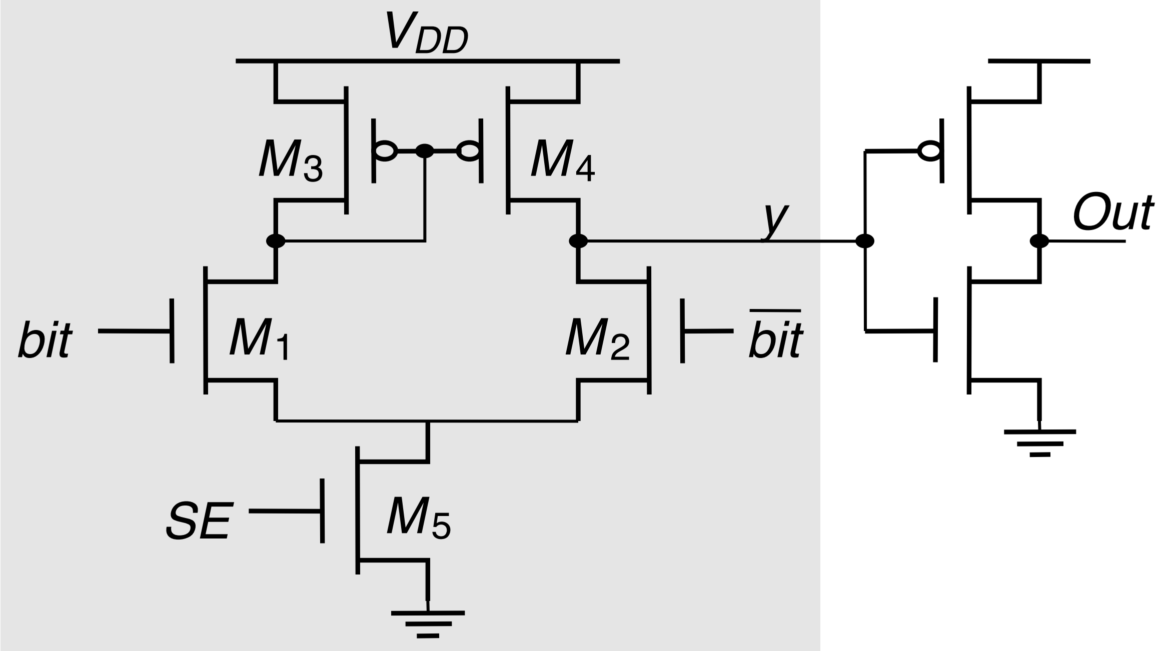
$$\tau_p \propto \frac{C \cdot \Delta V}{I_{av}}$$

small

Idea: Use “Sense Amplifier”

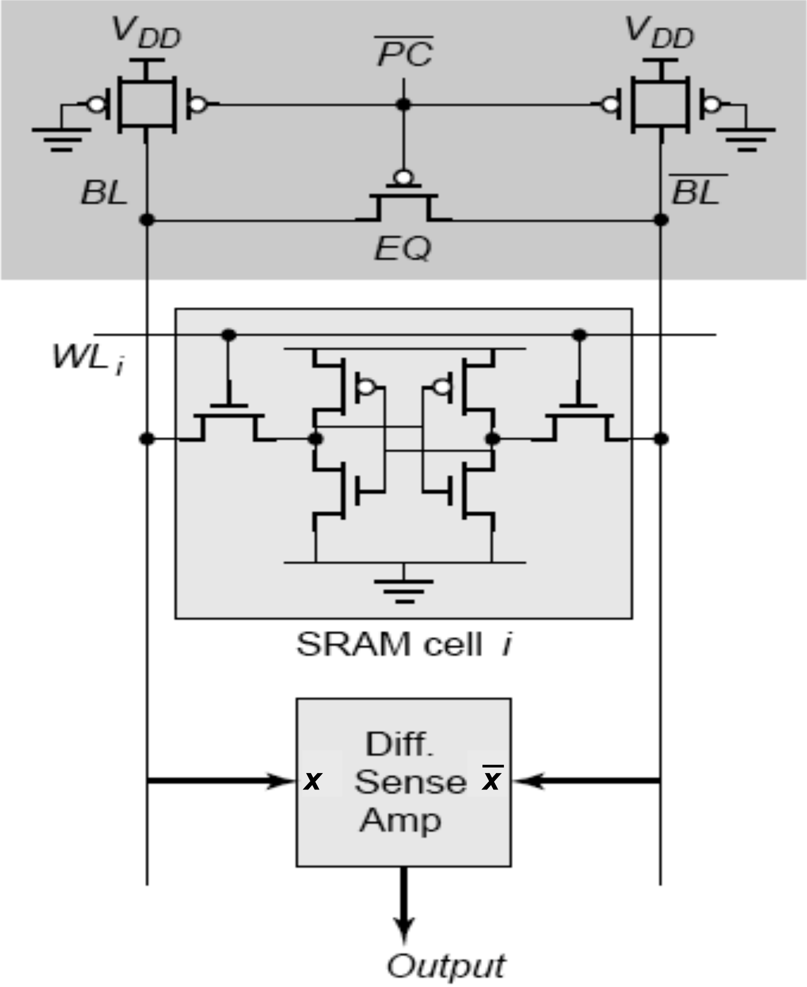


Differential Sense Amplifier



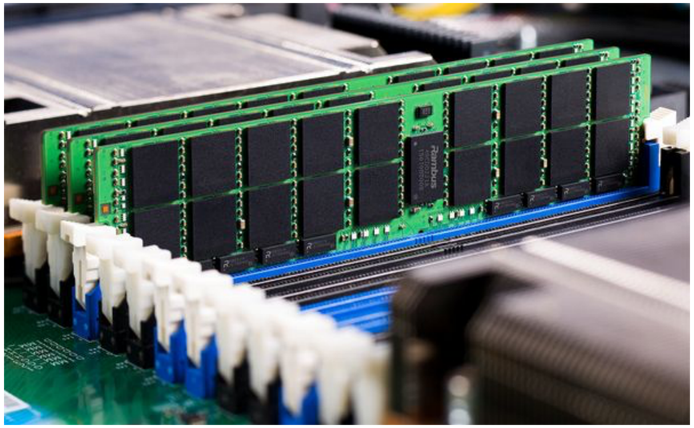
Classic Differential Amp structure - basis of opAmp

Differential Sensing — SRAM



(a) SRAM sensing scheme

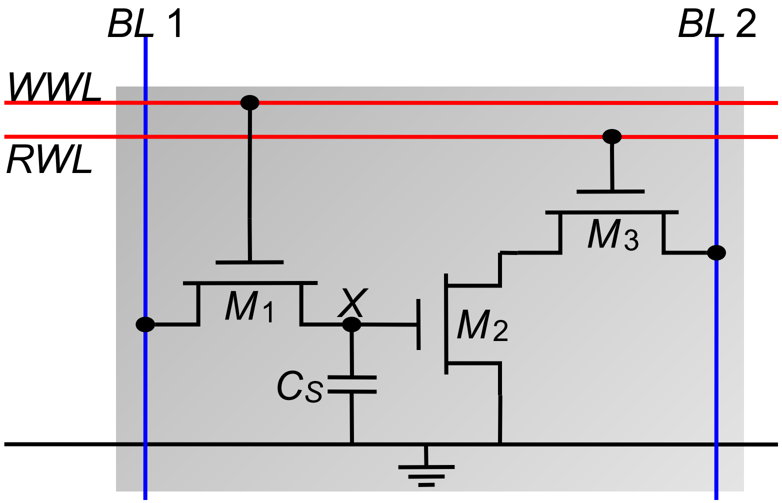
DRAM

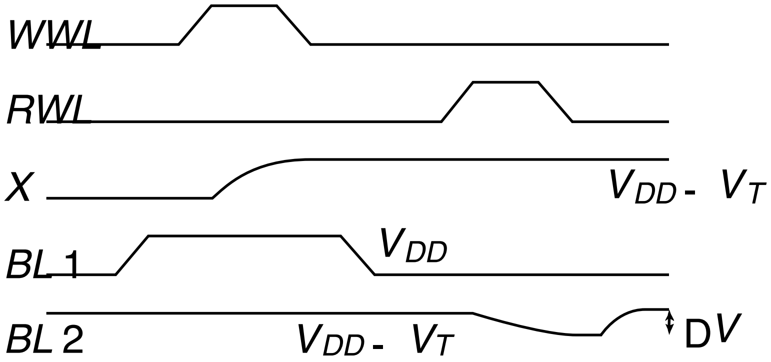


- Gets used for off-chip large inexpensive memories.
- Most commonly not compatible with logic processes. Requires special IC processing.

3-Transistor DRAM Cell

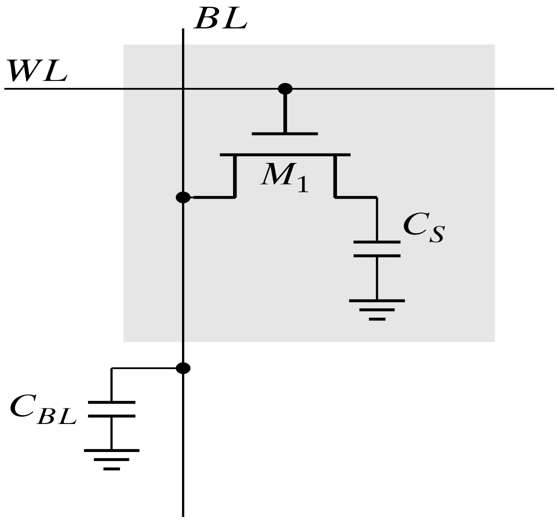
Can work with a normal logic IC process

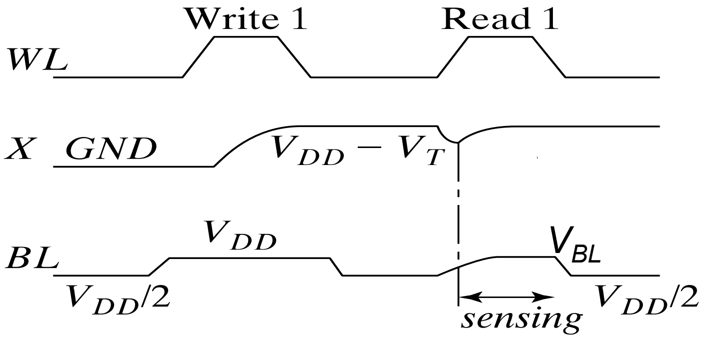




No constraints on device ratios
Reads are non-destructive
Value stored at node X when writing a "1" = $V_{WWL} - V_{Tn}$

1-Transistor DRAM Cell





V_{BIT}= 0 or $(V_{DD} – V_T)$

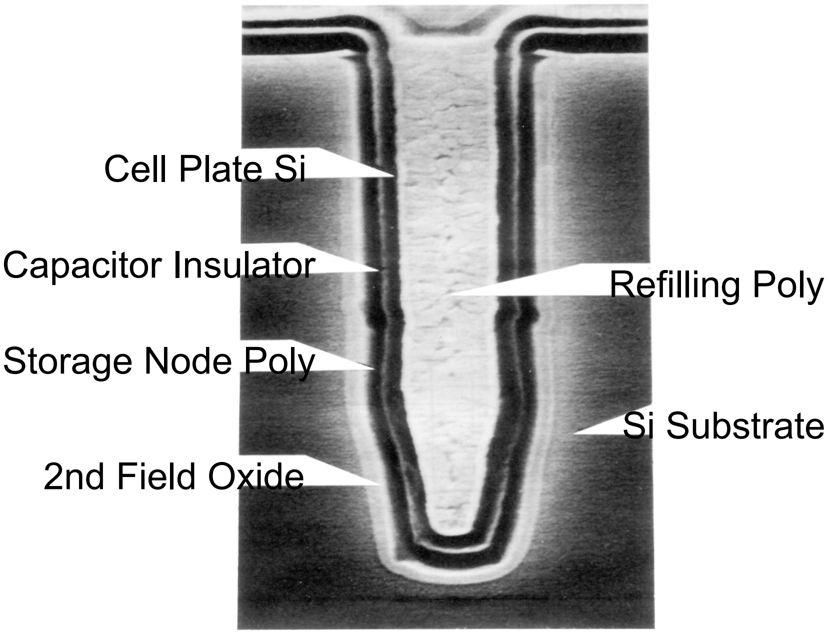
Write: C_s is charged or discharged by asserting WL and BL.

Read: Charge redistribution takes places between bit line and storage capacitance

$C_S << C_{BL}$ Voltage swing is small; typically around 250 mV or less.

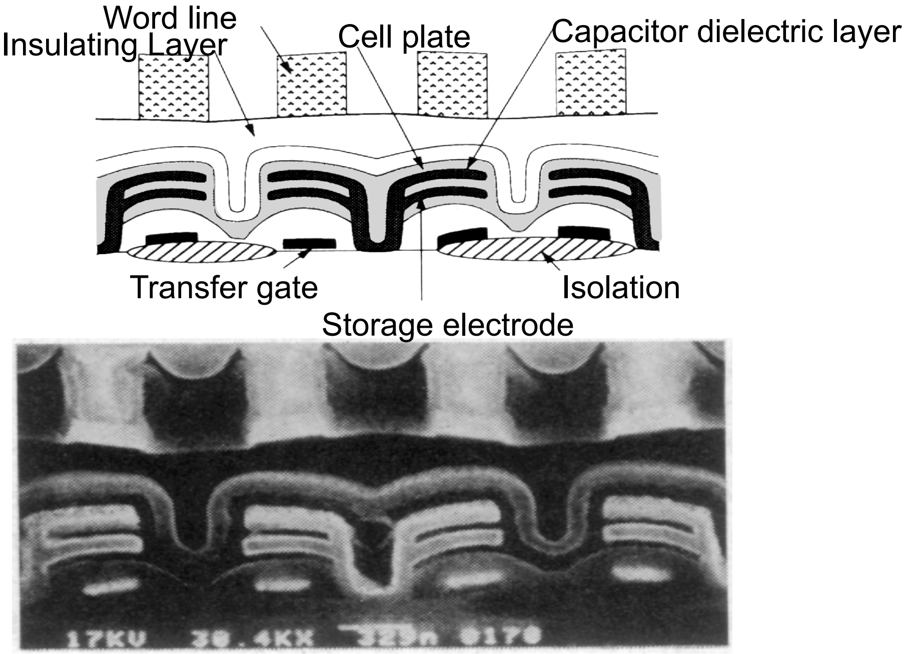
- To get sufficient C_s, **special IC process is used**
- Cell reading is destructive, therefore read operation always is followed by a write-back
- Cell looses charge (leaks away in ms - highly temperature dependent), therefore cells occasionally need to be “refreshed” using read/write cycle

Advanced 1T DRAM Cells



Trench Cell

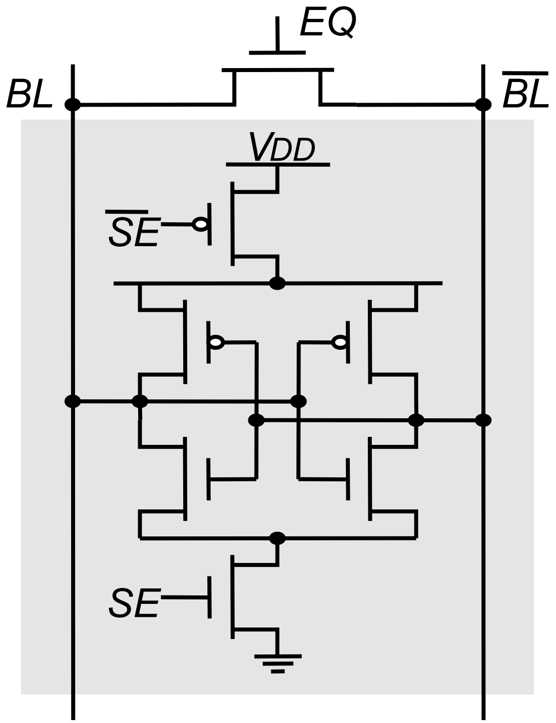
Not common



Stacked-capacitor Cell

Common

Latch-Based Sense Amplifier (DRAM)



- Bit lines equalized, with EQ, and precharged to Vdd/2
- Sense amp initialized to its meta-stable point with EQ
- Once adequate voltage gap created, sense amp enabled with SE
- Positive feedback quickly forces output to a stable operating point.
- Bit line from inactive array below/above used as reference for differential sensing.

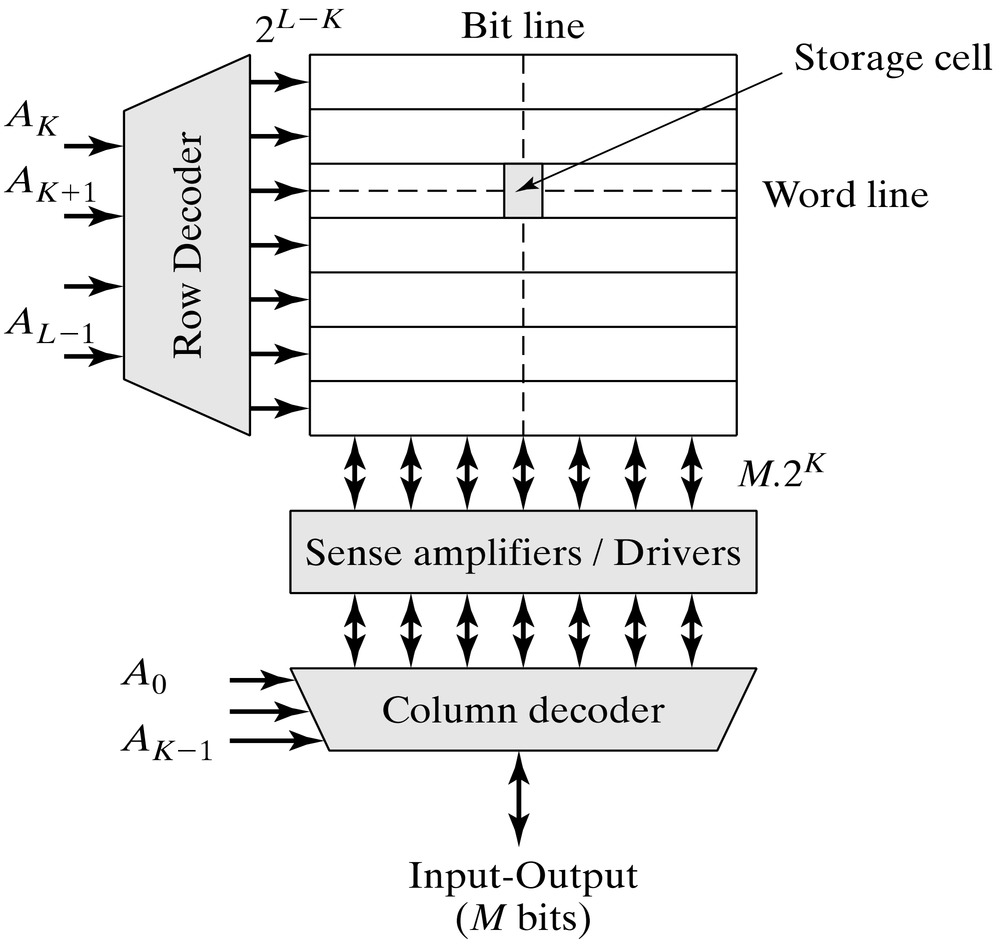
Memory Blocks

- ❑ Multi-ported RAM
- ❑ Combining Memory blocks
- ❑ FIFOs
- ❑ FPGA memory blocks
- ❑ Caches
- ❑ Memory Blocks in the Project

Multi-ported memory

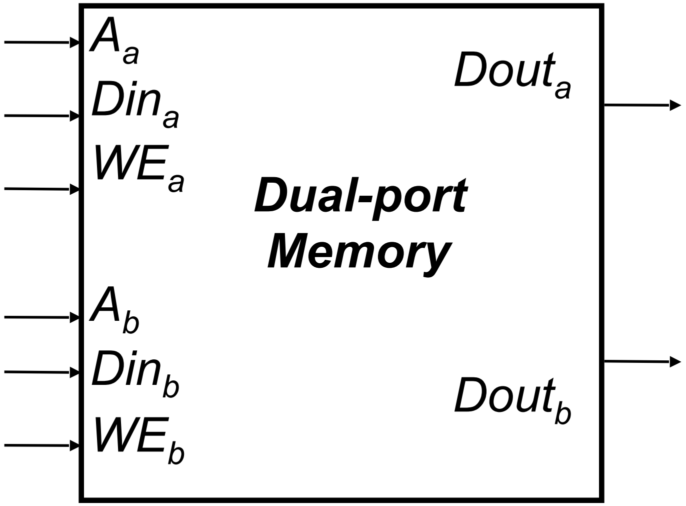
Memory Architecture Review

- **Word lines** used to select a row for reading or writing
- **Bit lines** carry data to/from periphery
- **Core** *aspect ratio* keep close to 1 to help balance delay on word line versus bit line
- **Address bits** are divided between the two decoders
- **Row decoder** used to select word line
- **Column decoder** used to select one or more columns for input/output of data

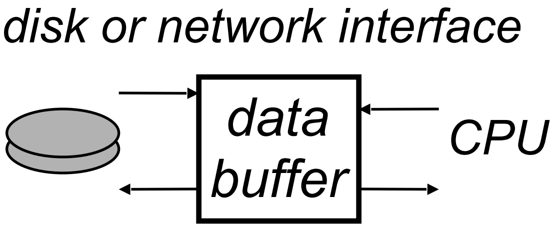


Multi-ported Memory

- Motivation:
  - Consider CPU core register file:
    - 1 read or write per cycle limits processor performance.
    - Complicates pipelining. Difficult for different instructions to simultaneously read or write regfile.
    - Common arrangement in pipelined CPUs is 2 read ports and 1 write port.



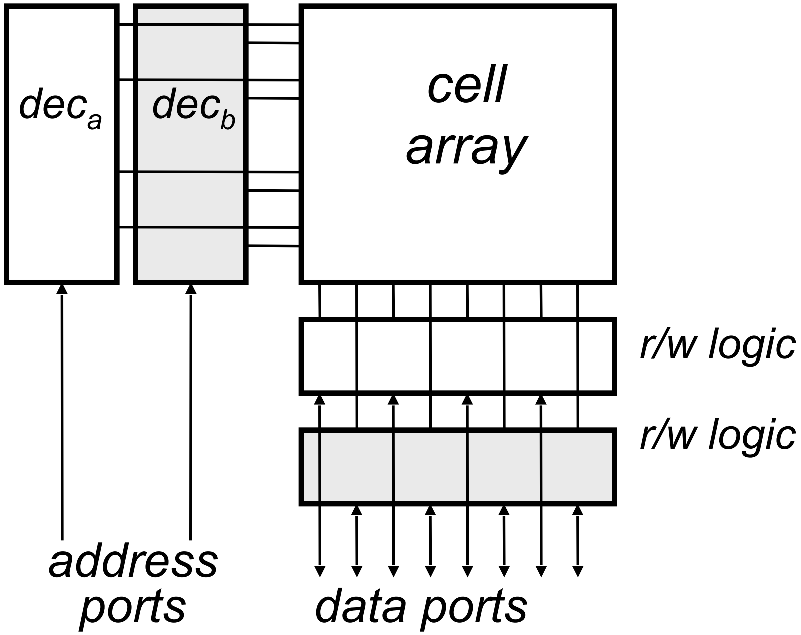
– I/O data buffering:



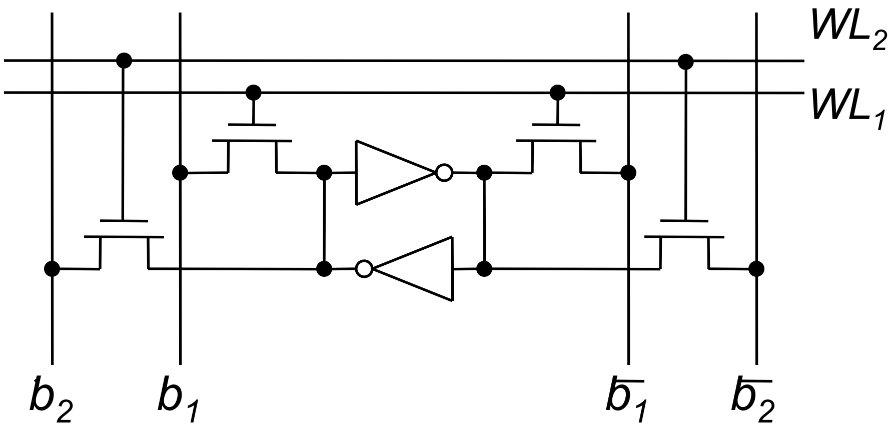
- dual-porting allows both sides to simultaneously access memory at full bandwidth.

Dual-ported Memory Internals

- Add decoder, another set of read/write logic, bits lines, word lines:



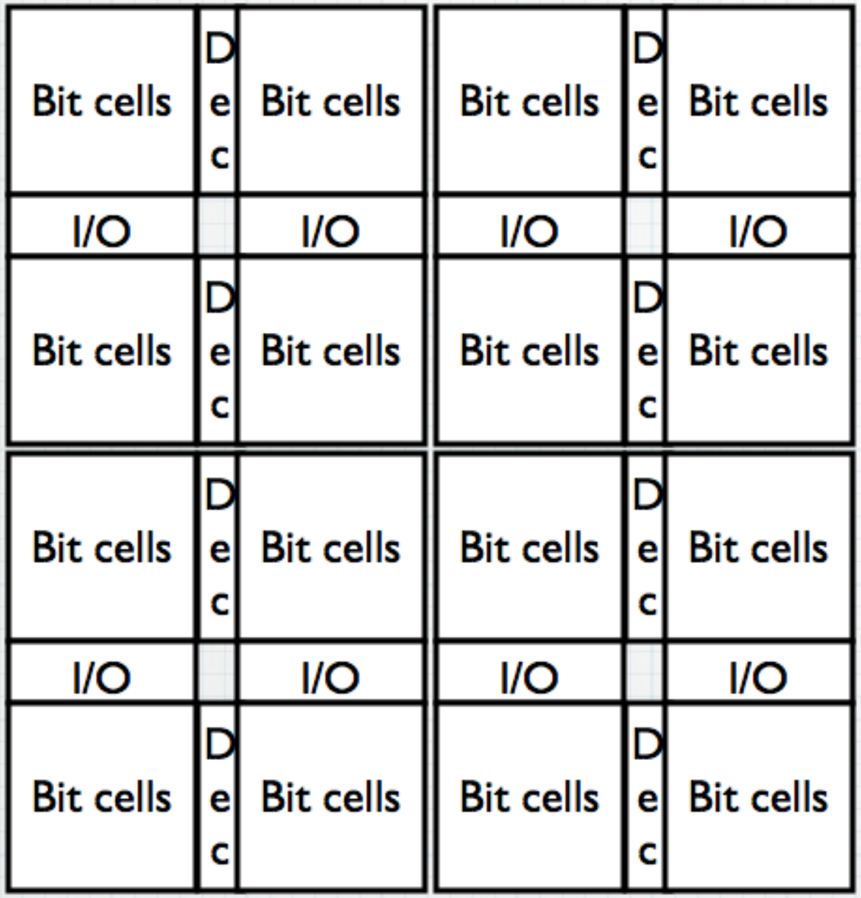
- *Example cell: SRAM*



- *Repeat everything but cross-coupled inverters.*
- *This scheme extends up to a couple more ports, then need to add additional transistors.*

Combining Memory Blocks

Building Larger Memories



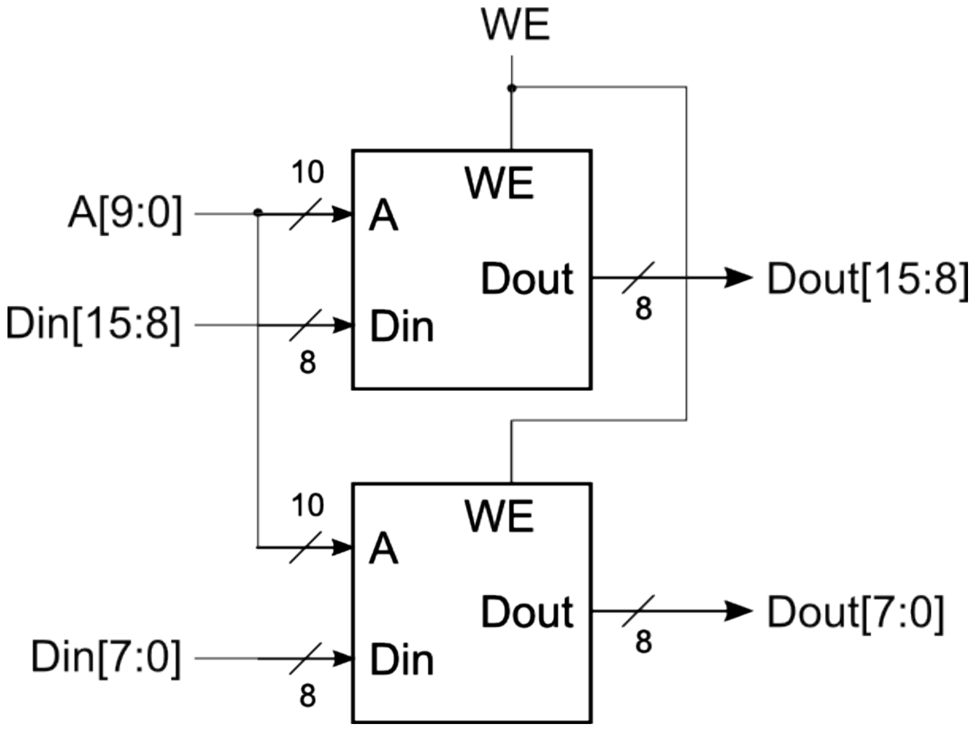
Bit cells	Dec	Bit cells	Bit cells	Dec	Bit cells
I/O		I/O	I/O		I/O
Bit cells	Dec	Bit cells	Bit cells	Dec	Bit cells
Bit cells	Dec	Bit cells	Bit cells	Dec	Bit cells
I/O		I/O	I/O		I/O
Bit cells	Dec	Bit cells	Bit cells	Dec	Bit cells

- 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

Cascading Memory-Blocks

How to make larger memory blocks out of smaller ones.

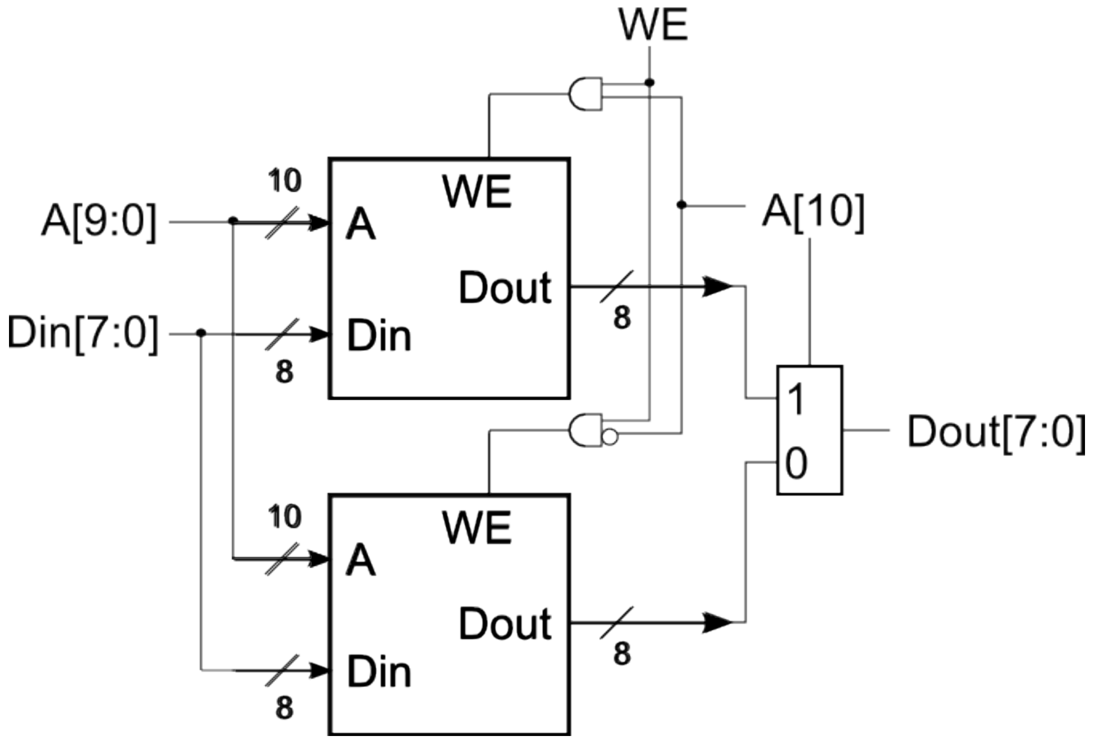
Increasing the width. Example: given 1Kx8, want 1Kx16



Cascading Memory-Blocks

How to make larger memory blocks out of smaller ones.

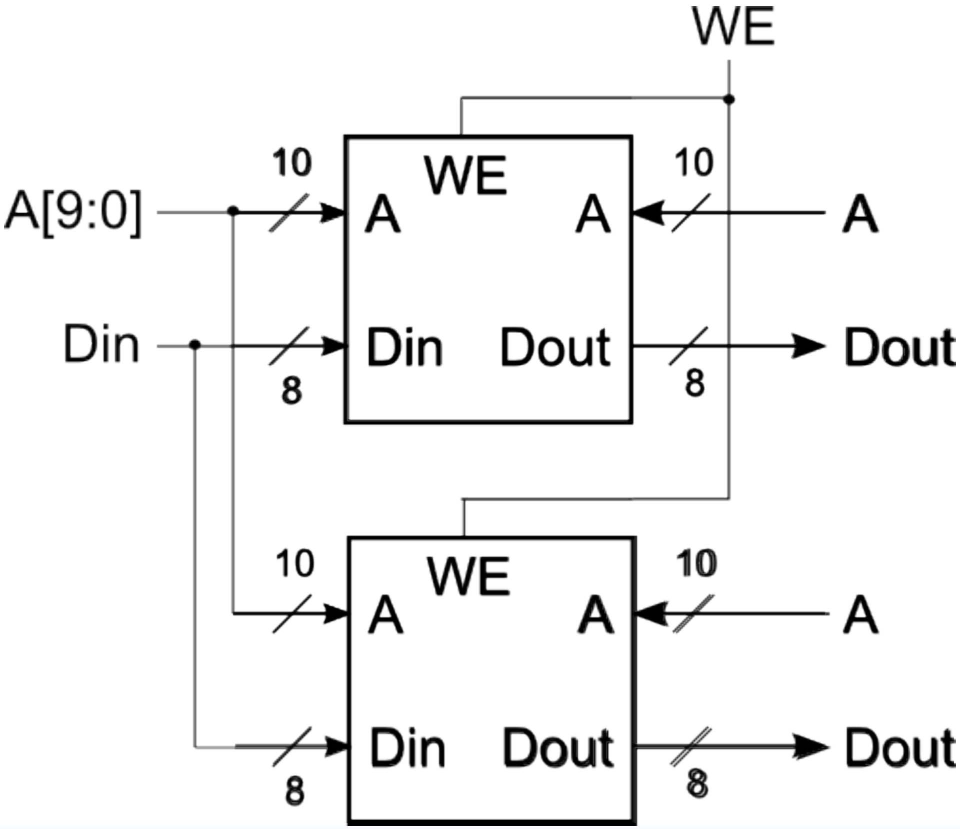
Increasing the depth. Example: given 1Kx8, want 2Kx8



Adding Ports to Primitive Memory Blocks

Adding a read port to a simple dual port (SDP) memory.

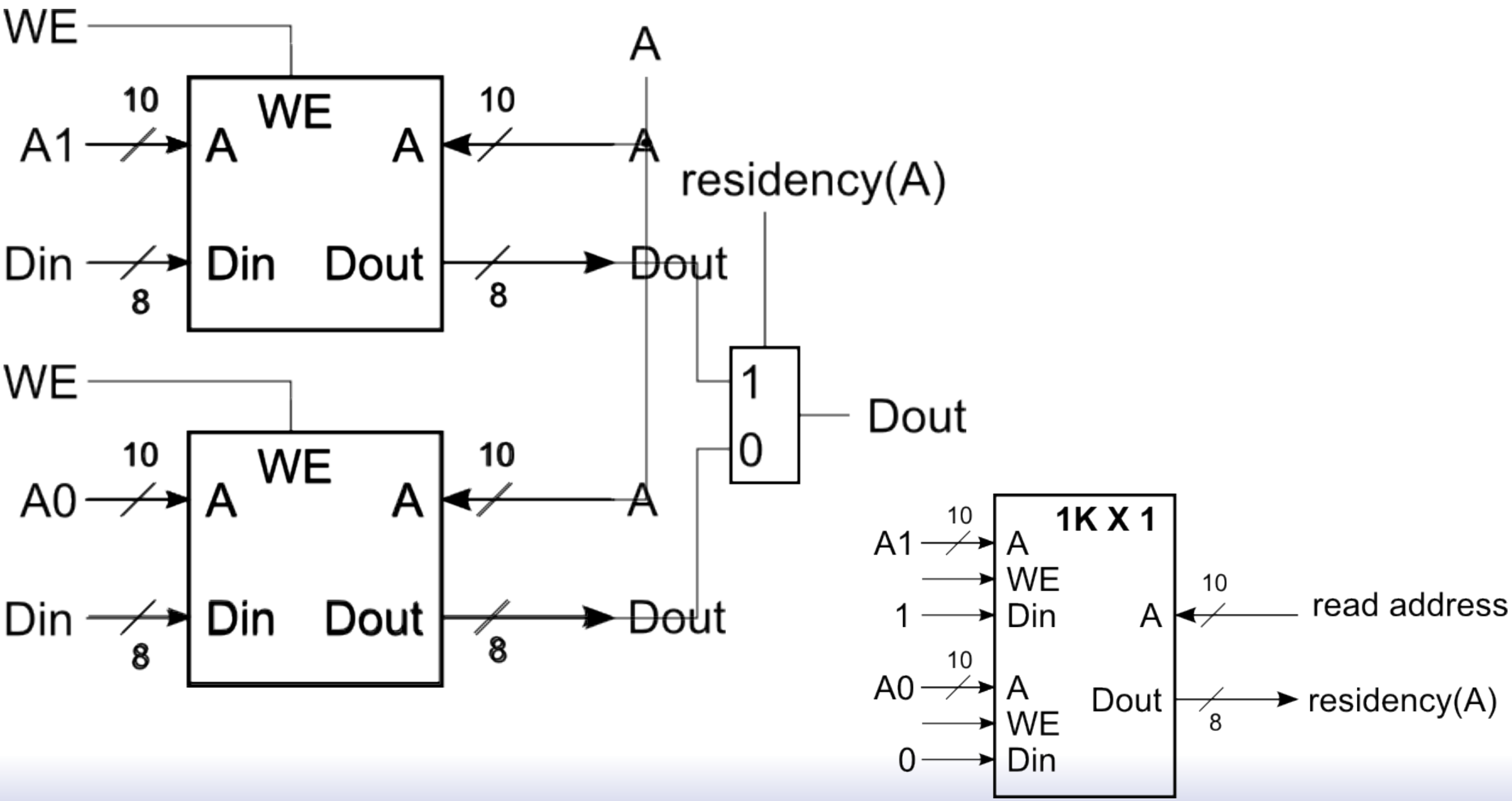
Example: given 1Kx8 SDP, want 1 write & 2 read ports.



Adding Ports to Primitive Memory Blocks

How to add a write port to a simple dual port memory.

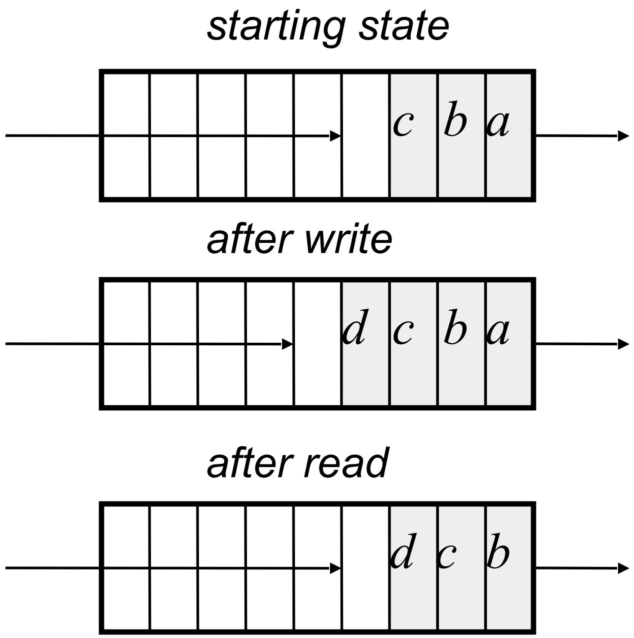
Example: given 1Kx8 SDP, want 1 read & 2 write ports.



FIFOs

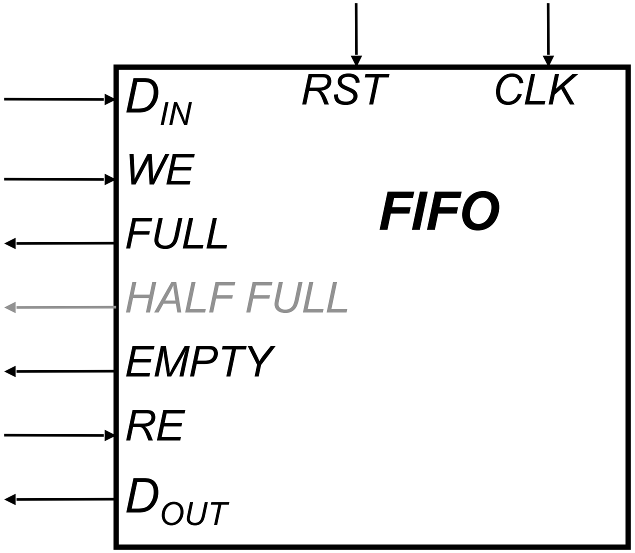
First-in-first-out (FIFO) Memory

- Used to implement *queues*.
- These find common use in computers and communication circuits.
- Generally, used to “decouple” actions of producer and consumer:



- *Producer can perform many writes without consumer performing any reads (or vis versa). However, because of finite buffer size, on average, need equal number of reads and writes.*
- *Typical uses:*
  - *interfacing I/O devices. Example network interface. Data bursts from network, then processor bursts to memory buffer (or reads one word at a time from interface). Operations not synchronized.*
  - *Example: Audio output. Processor produces output samples in bursts (during process swap-in time). Audio DAC clocks it out at constant sample rate.*

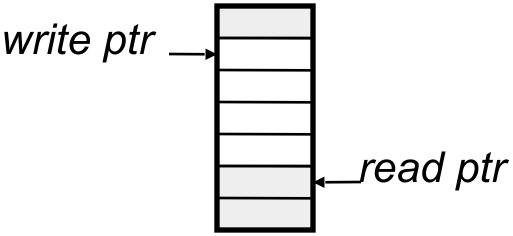
FIFO Interfaces



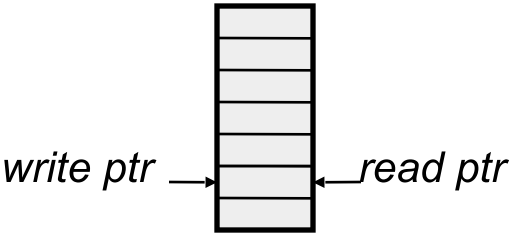
- ❑ After write or read operation, FULL and EMPTY indicate status of buffer.
- ❑ Used by external logic to control it's own reading from or writing to the buffer.
- ❑ FIFO resets to EMPTY state.
- ❑ HALF FULL (or other indicator of partial fullness) is optional.

"Circular buffer" implementation:

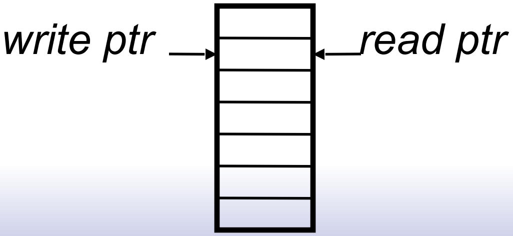
- *Address pointers are used internally to keep next write position and next read position into a dual-port memory.*



- *If pointers equal after write ⇒ FULL:*



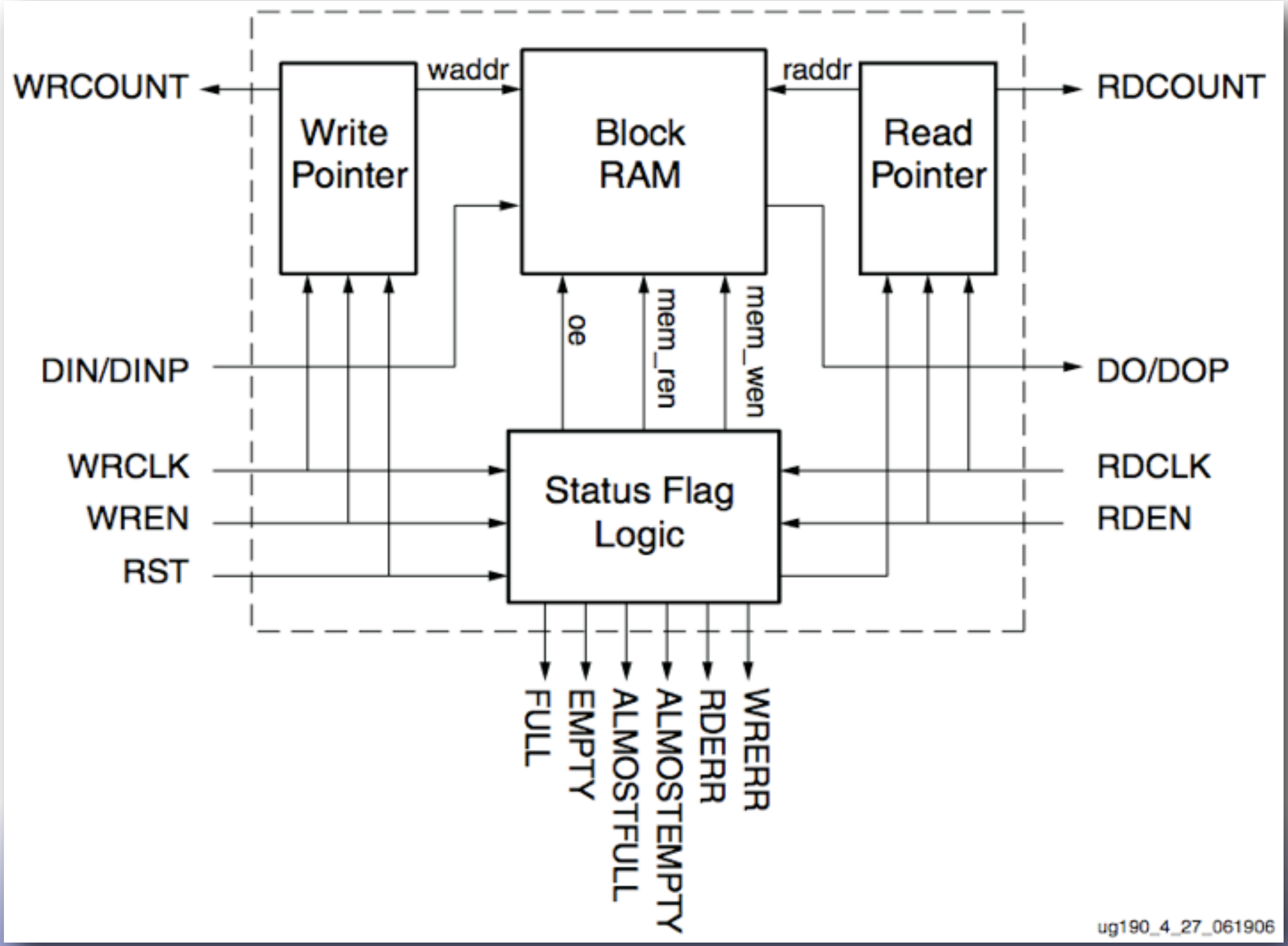
- *If pointers equal after read ⇒ EMPTY:*



Note: pointer incrementing is done "mod size-of-buffer"

Xilinx Virtex5 FIFOs

- Virtex5 BlockRAMS include dedicated circuits for FIFOs.
- Details in User Guide (ug190).
- Takes advantage of separate dual ports and independent ports clocks.

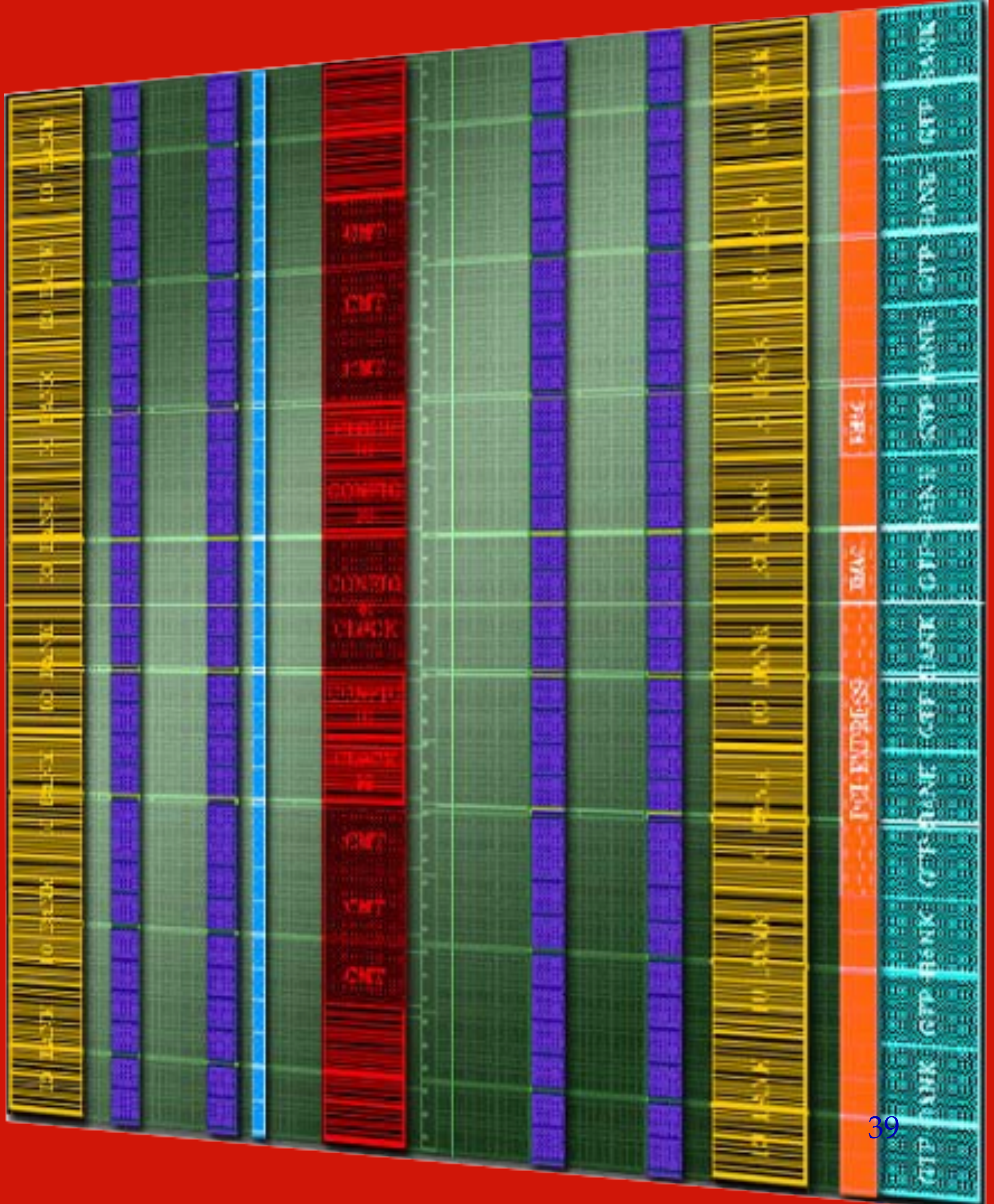


Memory on FPGAs

Virtex-5 LX110T memory blocks.

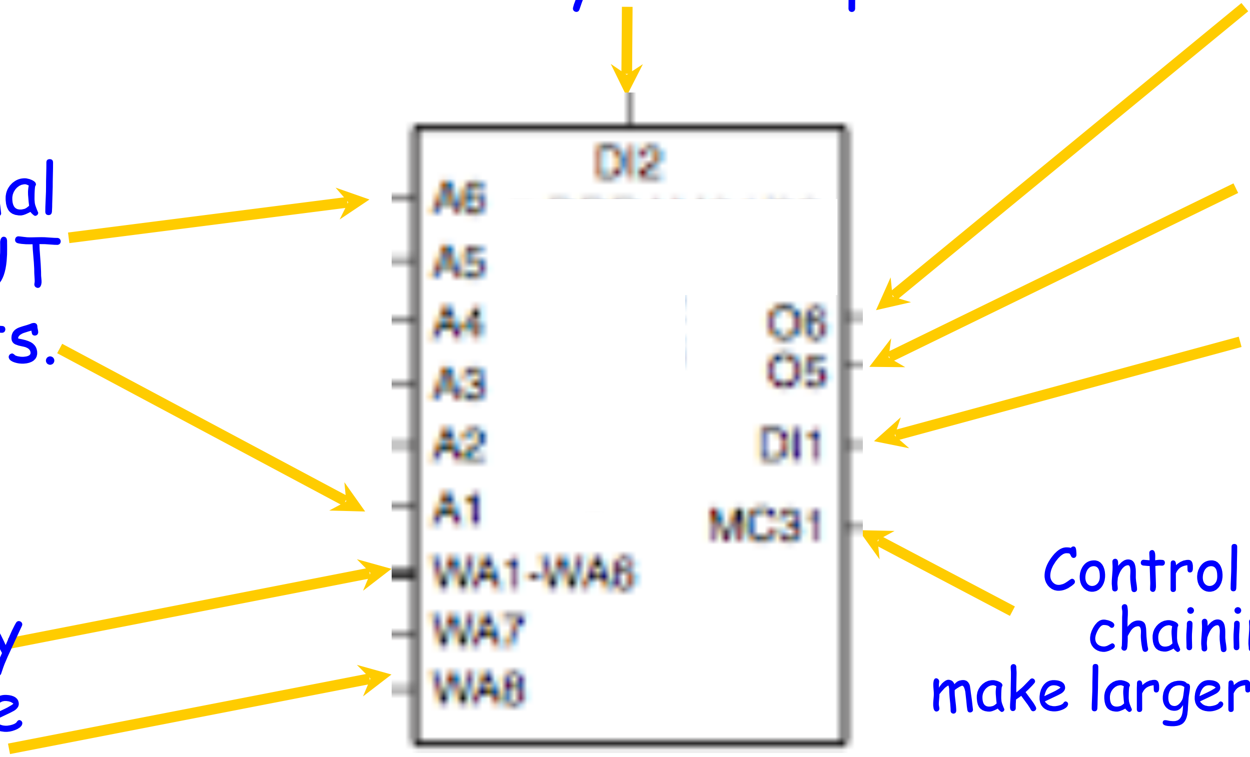
Distributed RAM using LUTs among the CLBs.

Block RAMs in four columns.



A SLICEM 6-LUT … (‘distributed RAM”)

Memory data input



Normal 6-LUT inputs.

Memory write address

Normal 5/6-LUT outputs.

Memory data input.

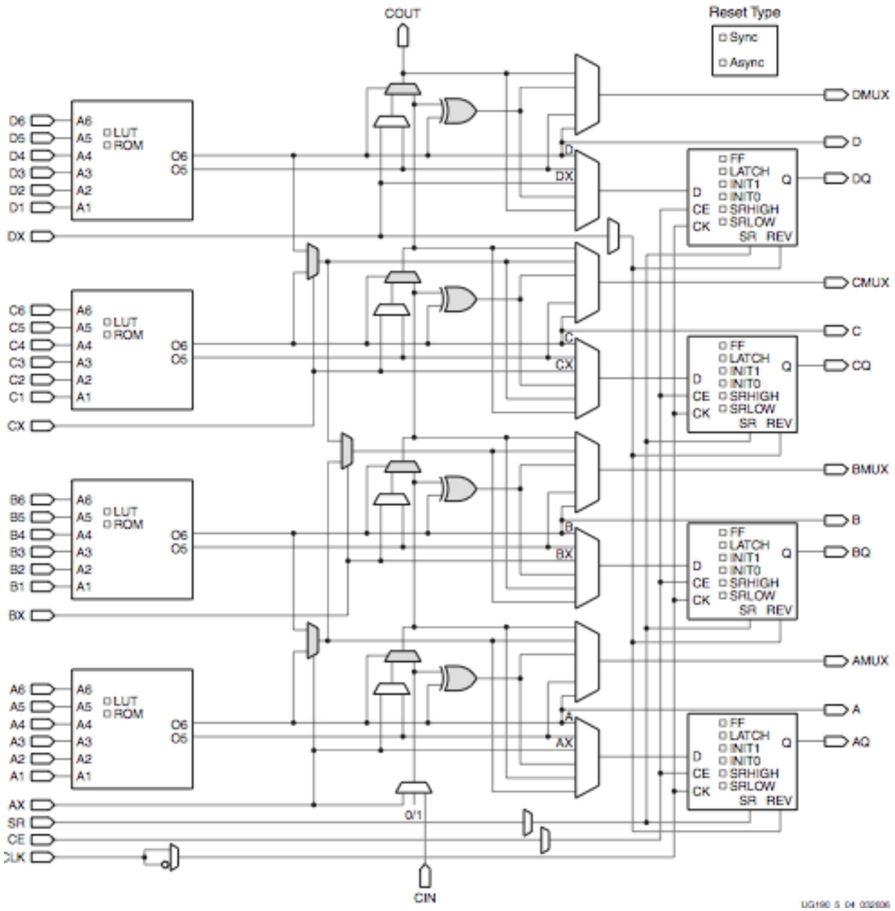
Control output for chaining LUTs to make larger memories.

Synchronous write / asychronous read

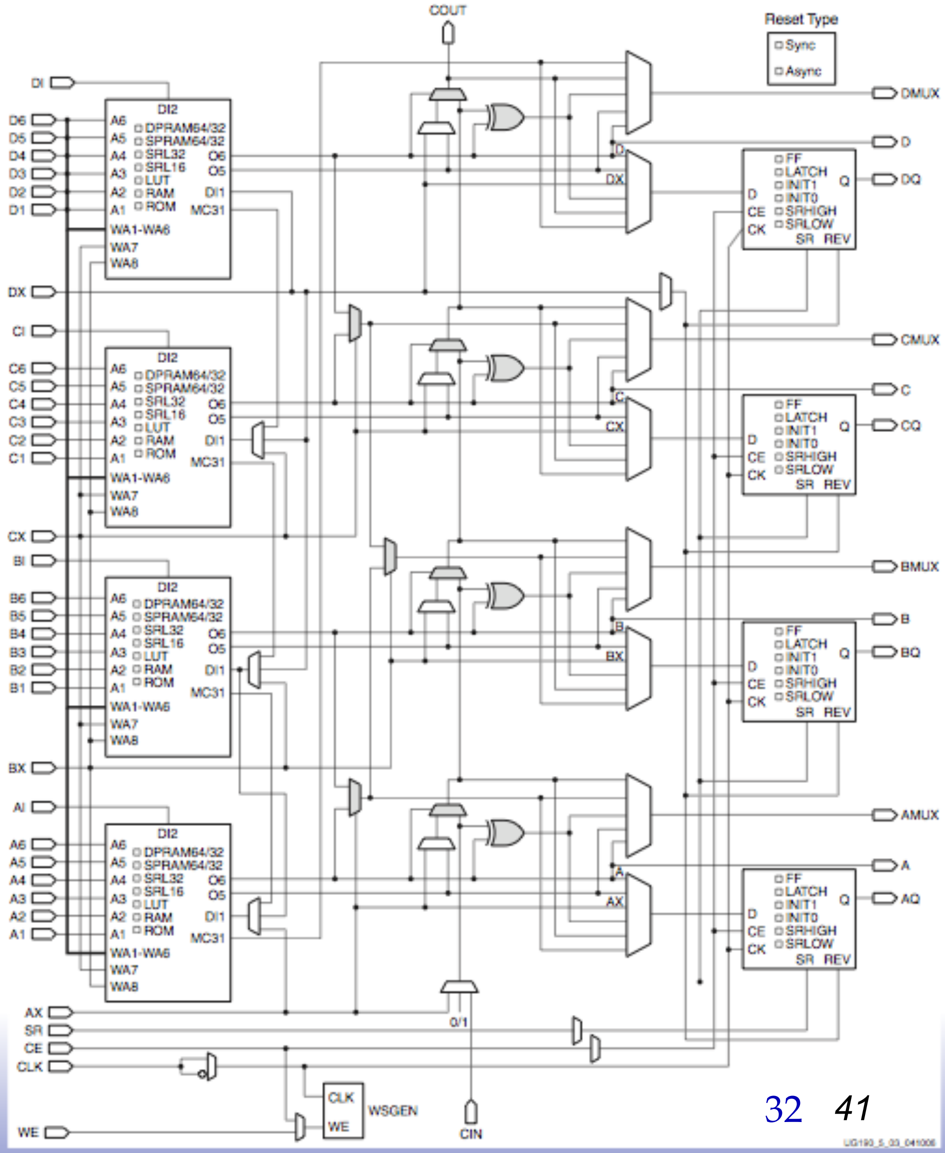
A 1.1 Mb distributed RAM can be made if all SLICEMs of an LX110T are used as RAM.

SLICEL vs SLICEM ...

SLICEL



SLICEM



SLICEM adds memory features to LUTs, + muxes.

Example Distributed RAM (LUT RAM)

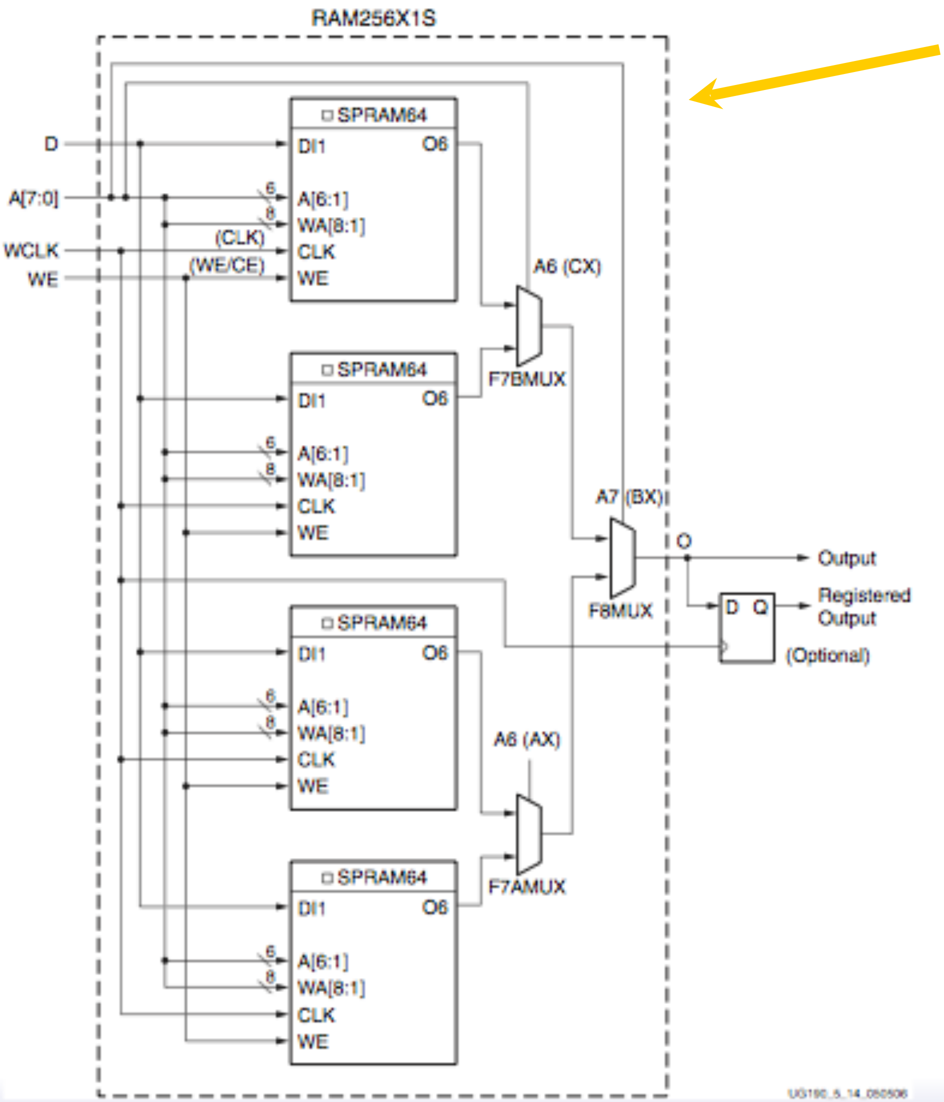
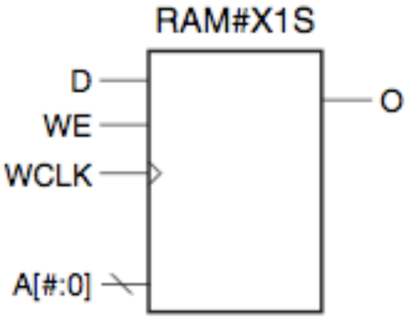
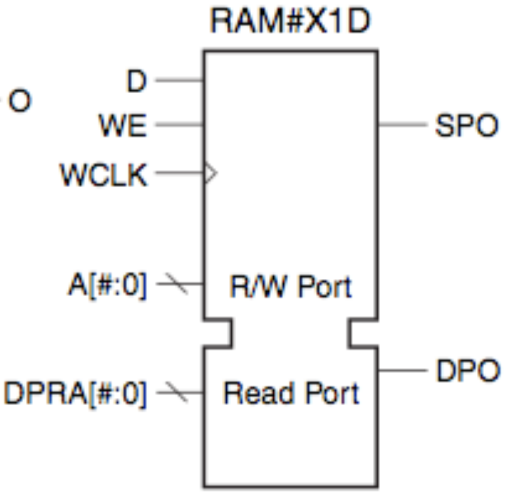


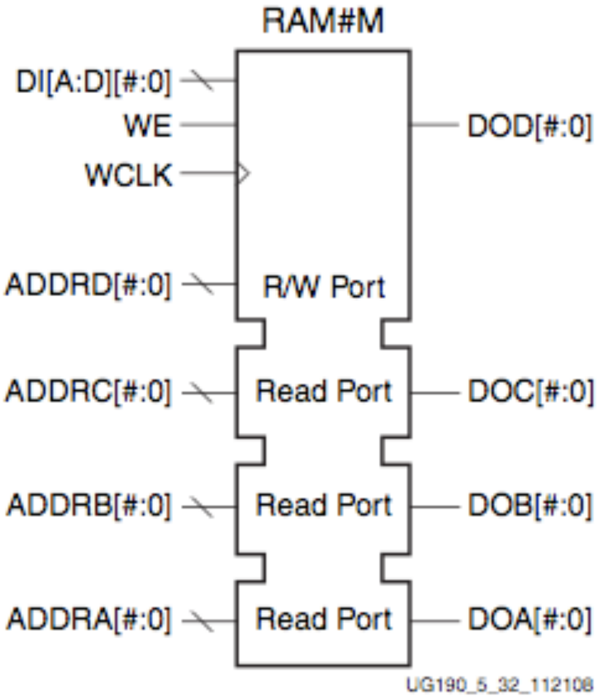
Figure 5-14: **Distributed RAM (RAM256X1S)**

Example configuration:
Single-port 256b x 1,
registered output.

Distributed RAM Primitives







- Single-Port 32 x 1-bit RAM
- Dual-Port 32 x 1-bit RAM
- Quad-Port 32 x 2-bit RAM
- Simple Dual-Port 32 x 6-bit RAM
- Single-Port 64 x 1-bit RAM
- Dual-Port 64 x 1-bit RAM
- Quad-Port 64 x 1-bit RAM
- Simple Dual-Port 64 x 3-bit RAM
- Single-Port 128 x 1-bit RAM
- Dual-Port 128 x 1-bit RAM
- Single-Port 256 x 1-bit RAM

All are built from a single slice or less.

Remember, though, that the SLICEM LUT is naturally only 1 read and 1 write port.

Distributed RAM Timing

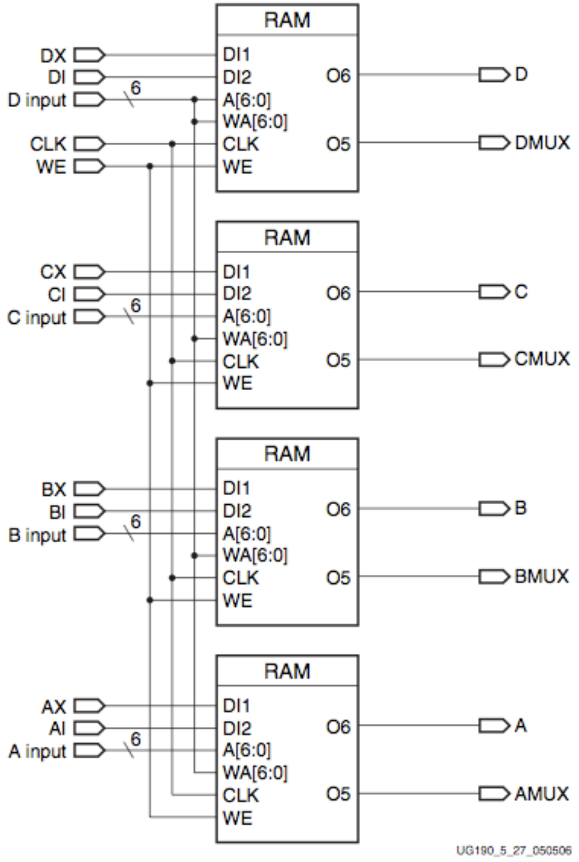
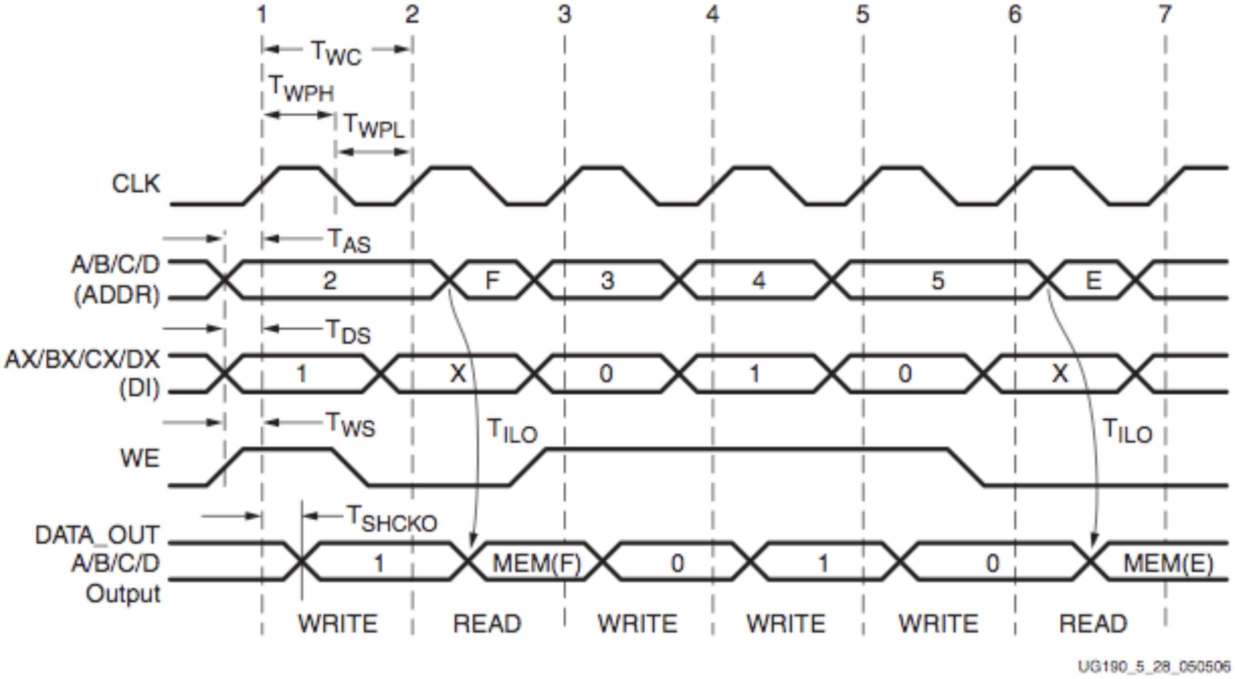
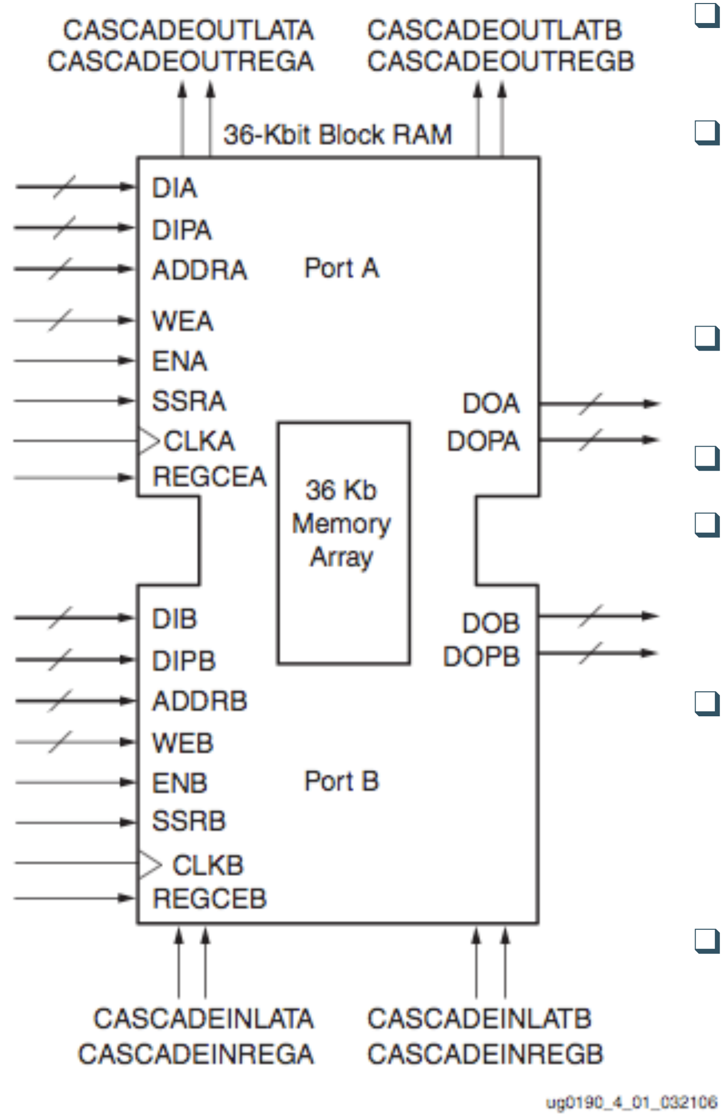


Figure 5-27: **Simplified Virtex-5 FPGA SLICEM Distributed RAM**

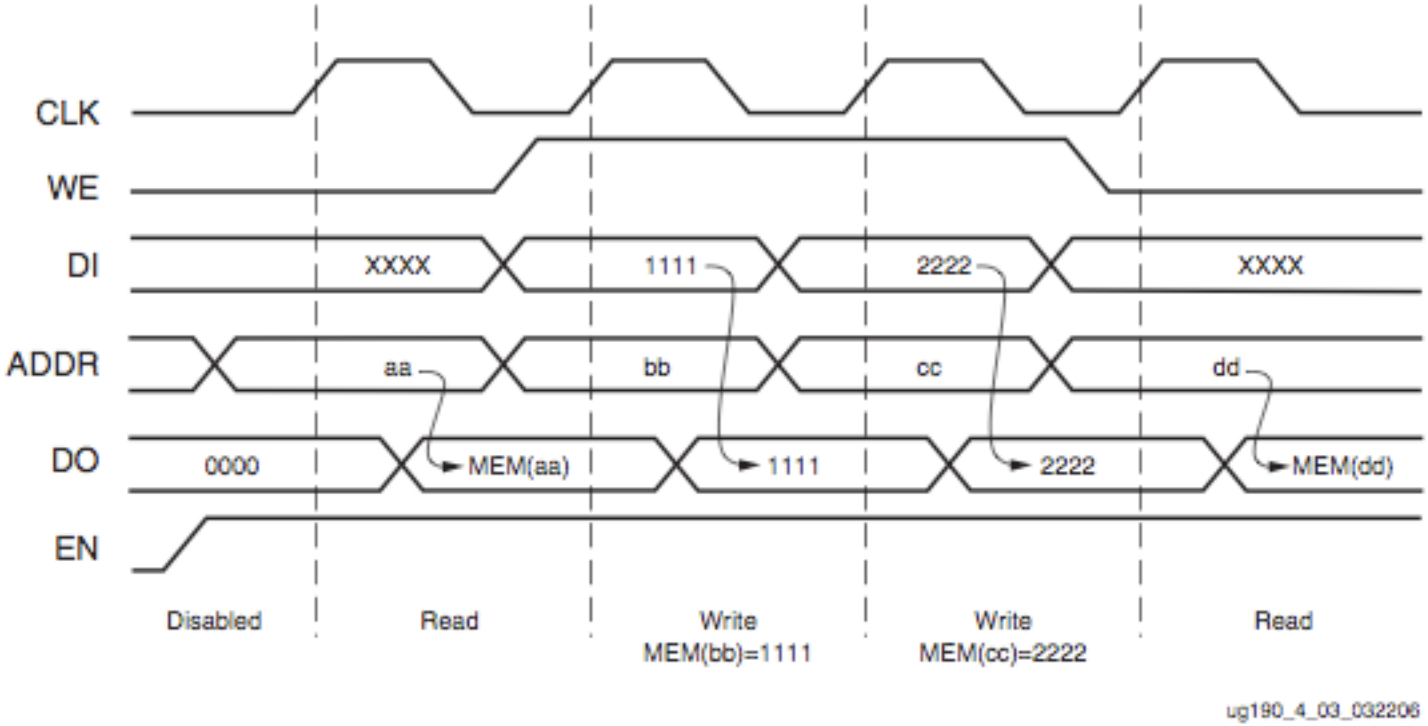


Block RAM Overview



- 36K bits of data total, can be configured as:
  - 2 independent 18Kb RAMs, or one 36Kb RAM.
- Each 36Kb block RAM can be configured as:
  - 64Kx1 (when cascaded with an adjacent 36Kb block RAM), 32Kx1, 16Kx2, 8Kx4, 4Kx9, 2Kx18, or 1Kx36 memory.
- Each 18Kb block RAM can be configured as:
  - 16Kx1, 8Kx2, 4Kx4, 2Kx9, or 1Kx18 memory.
- Write and Read are synchronous operations.
- The two ports are symmetrical and totally independent (can have different clocks), sharing only the stored data.
- Each port can be configured in one of the available widths, independent of the other port. The read port width can be different from the write port width for each port.
- The memory content can be initialized or cleared by the configuration bitstream.

Block RAM Timing



- Optional output register, would delay appearance of output data by one cycle.
- Maximum clock rate, roughly 400MHz.

Ultra-RAM Blocks

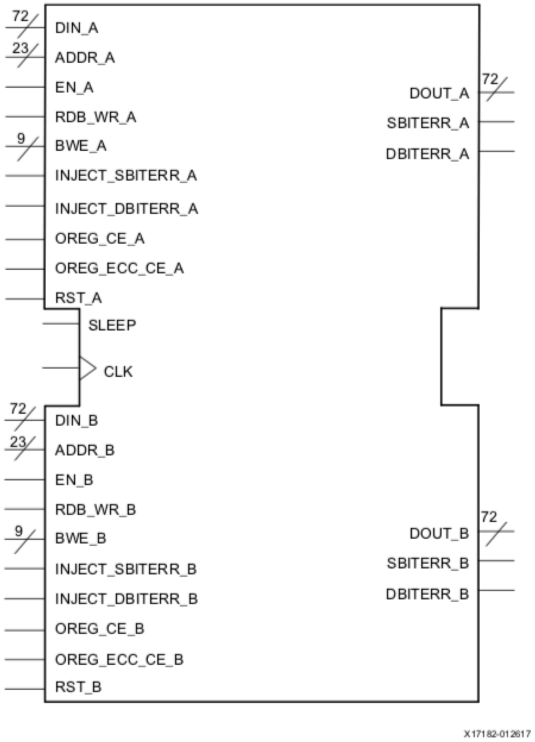
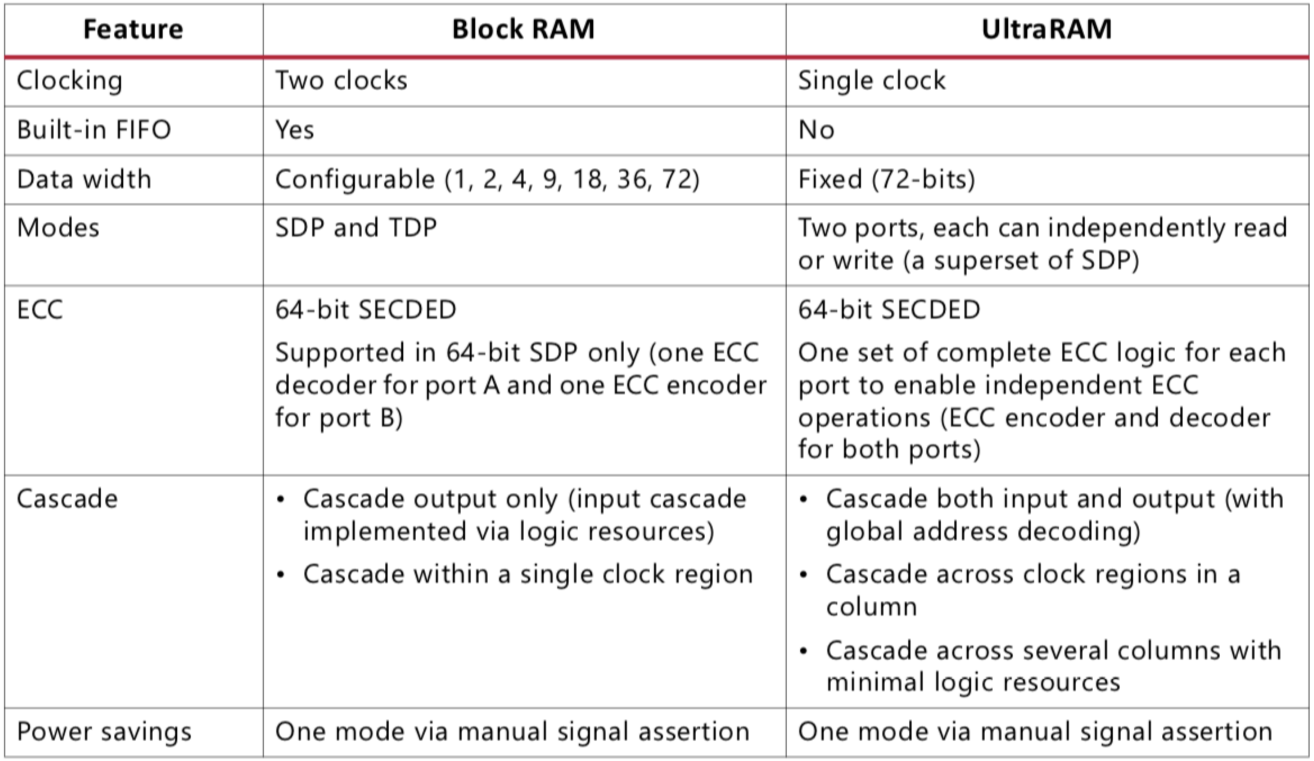


Figure 2-1: **UltraRAM URAM288_BASE Primitive**

Table 2-1: **Block RAM and UltraRAM Comparison**

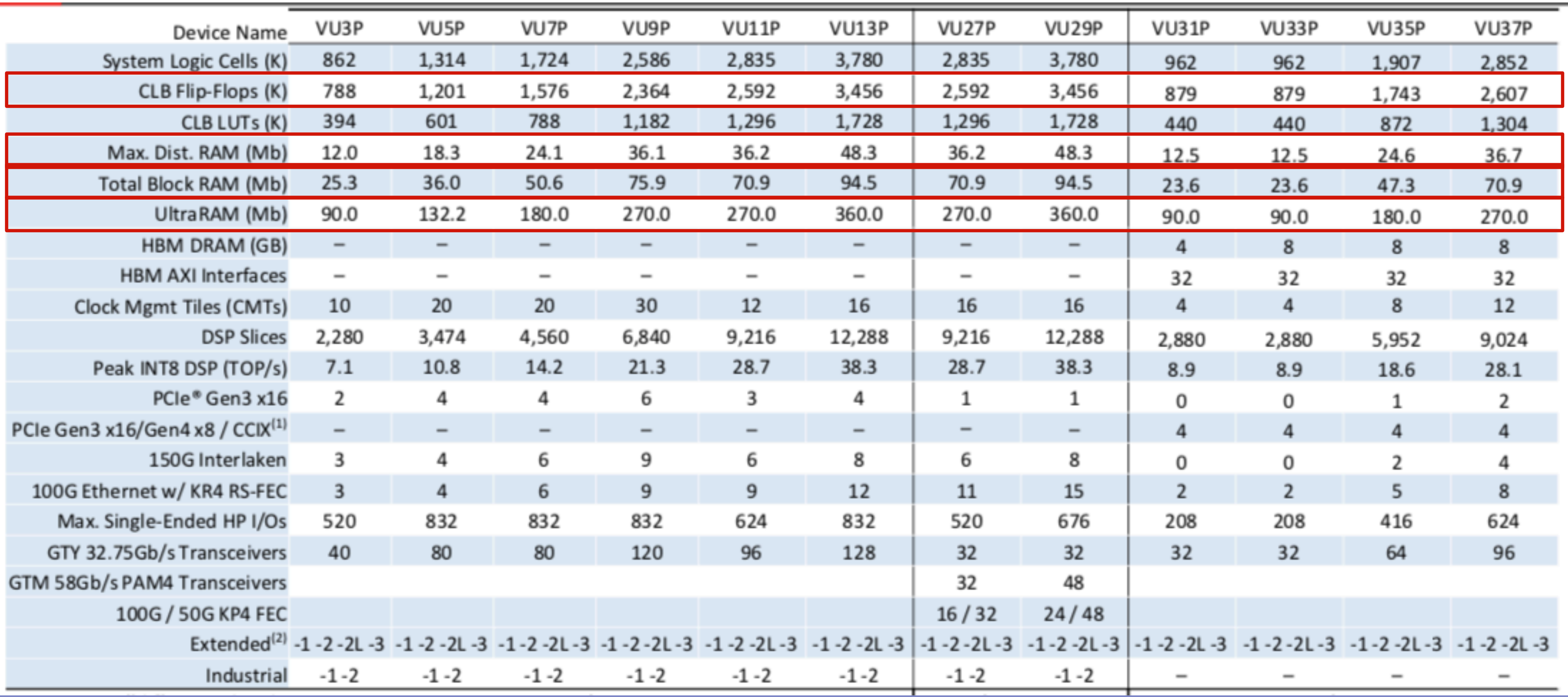


Feature	Block RAM	UltraRAM
Clocking	Two clocks	Single clock
Built-in FIFO	Yes	No
Data width	Configurable (1, 2, 4, 9, 18, 36, 72)	Fixed (72-bits)
Modes	SDP and TDP	Two ports, each can independently read or write (a superset of SDP)
ECC	64-bit SECDED Supported in 64-bit SDP only (one ECC decoder for port A and one ECC encoder for port B)	64-bit SECDED One set of complete ECC logic for each port to enable independent ECC operations (ECC encoder and decoder for both ports)
Cascade	• Cascade output only (input cascade implemented via logic resources) • Cascade within a single clock region	• Cascade both input and output (with global address decoding) • Cascade across clock regions in a column • Cascade across several columns with minimal logic resources
Power savings	One mode via manual signal assertion	One mode via manual signal assertion

State-of-the-Art - Xilinx FPGAs

45nm	28nm	20nm	16nm
SPARTAN 6	VIRTEX 7	VIRTEX UltraSCALE	VIRTEX UltraSCALE+
	KINTEX 7	KINTEX UltraSCALE	KINTEX UltraSCALE+
	ARTIX 7		
	SPARTAN 7		

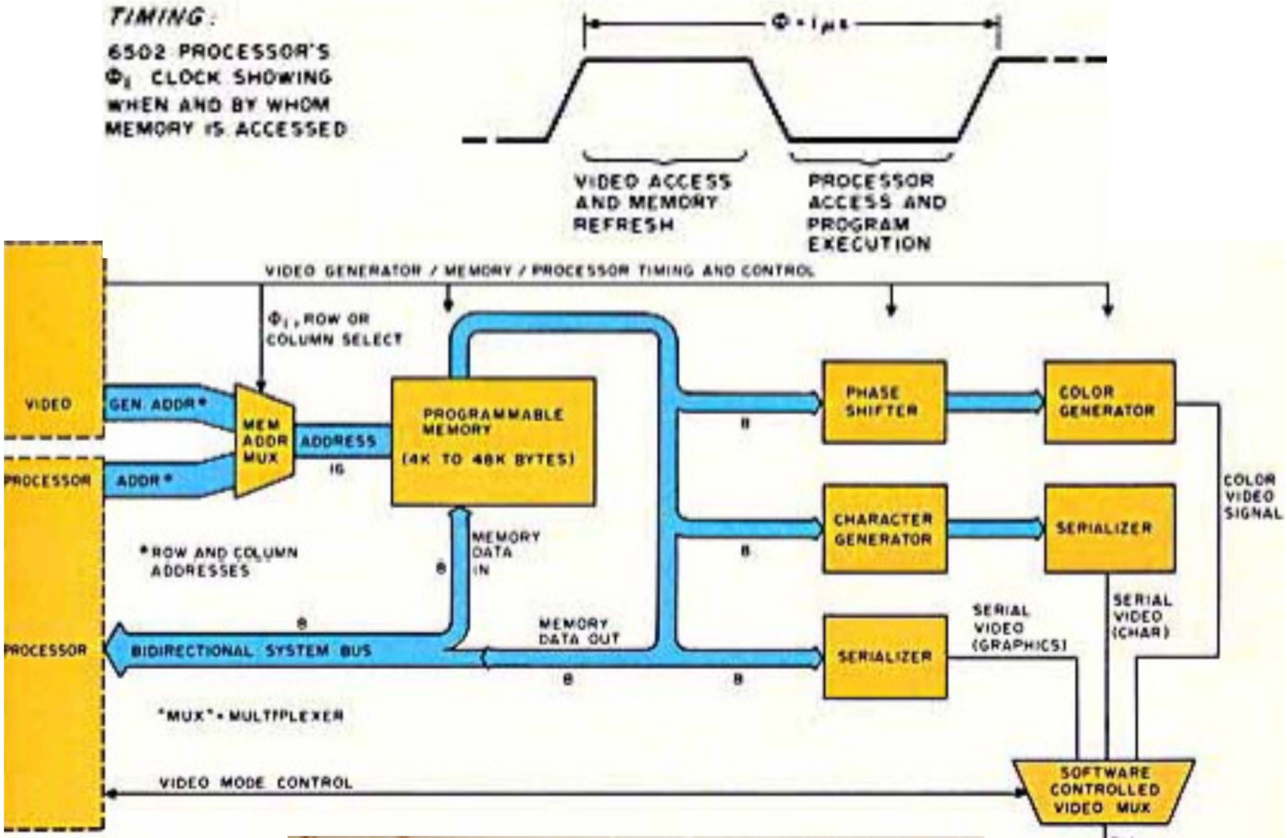
Virtex Ultra-scale



Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU27P	VU29P	VU31P	VU33P	VU35P	VU37P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	2,835	3,780	962	962	1,907	2,852
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	2,592	3,456	879	879	1,743	2,607
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	1,296	1,728	440	440	872	1,304
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	36.2	48.3	12.5	12.5	24.6	36.7
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	70.9	94.5	23.6	23.6	47.3	70.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	270.0	360.0	90.0	90.0	180.0	270.0
HBM DRAM (GB)	–	–	–	–	–	–	–	–	4	8	8	8
HBM AXI Interfaces	–	–	–	–	–	–	–	–	32	32	32	32
Clock Mgmt Tiles (CMTs)	10	20	20	30	12	16	16	16	4	4	8	12
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	9,216	12,288	2,880	2,880	5,952	9,024
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	28.7	38.3	8.9	8.9	18.6	28.1
PCIe® Gen3 x16	2	4	4	6	3	4	1	1	0	0	1	2
PCIe Gen3 x16/Gen4 x8 / CCIX[1]	–	–	–	–	–	–	–	–	4	4	4	4
150G Interlaken	3	4	6	9	6	8	6	8	0	0	2	4
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	11	15	2	2	5	8
Max. Single-Ended HP I/Os	520	832	832	832	624	832	520	676	208	208	416	624
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	32	32	32	32	64	96
GTM 58Gb/s PAM4 Transceivers							32	48				
100G / 50G KP4 FEC							16 / 32	24 / 48				
Extended[2]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–	–	–	–

Caches

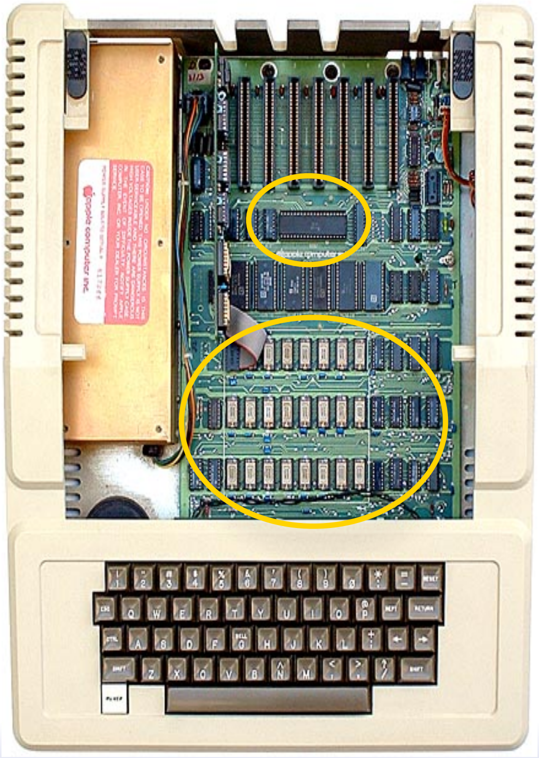
1977: DRAM faster than microprocessors

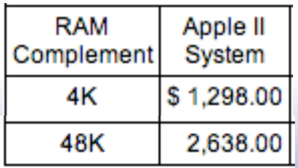


Apple II (1977)

CPU: 1000 ns

DRAM: 400 ns





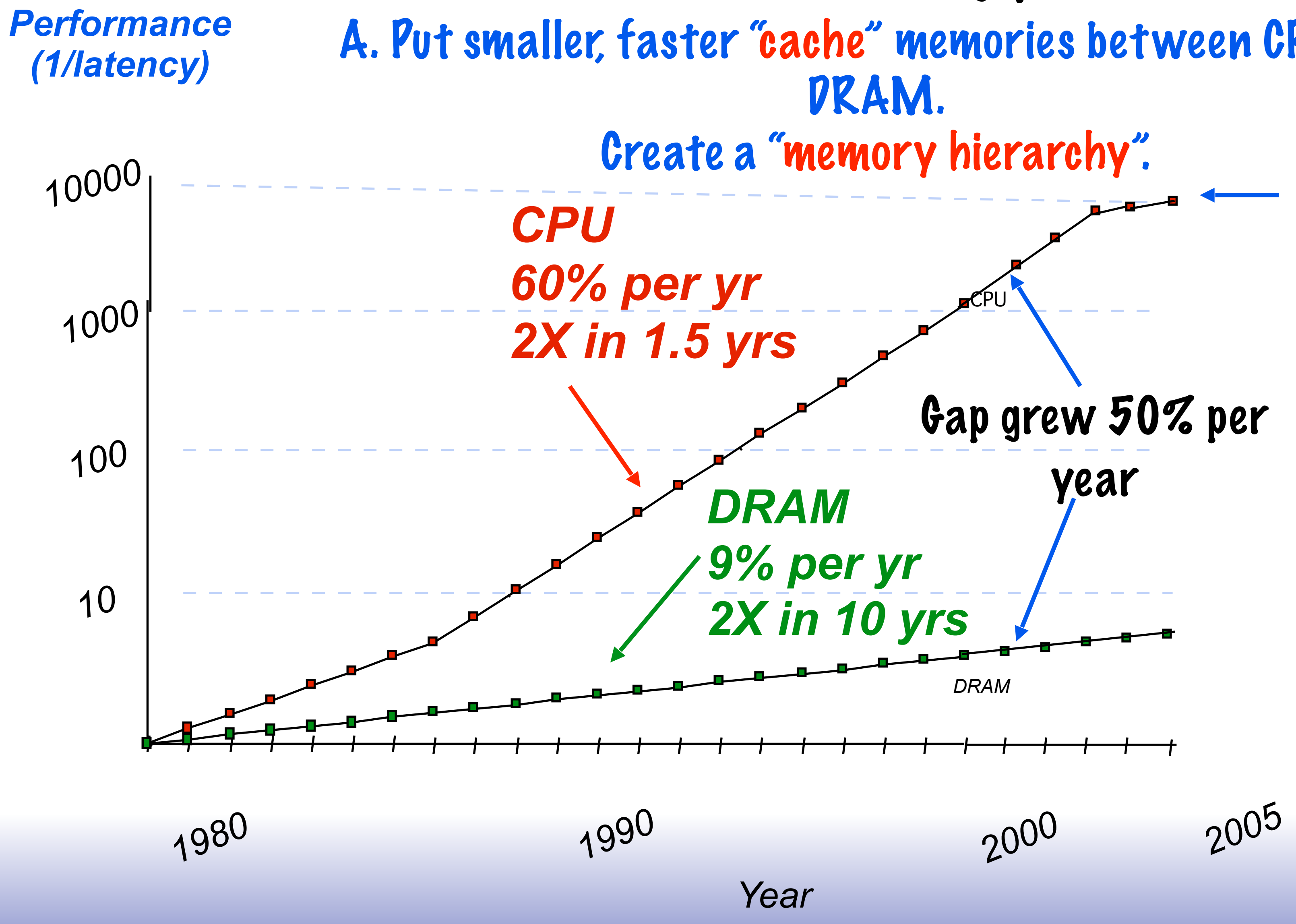
RAM Complement	Apple II System
4K	$ 1,298.00
48K	2,638.00

1980-2003, CPU speed outpaced DRAM ...

Q. How did architects address this gap?

A. Put smaller, faster "cache" memories between CPU and DRAM.

Create a "memory hierarchy".



CPU
60% per yr
2X in 1.5 yrs

DRAM
9% per yr
2X in 10 yrs

Gap grew 50% per year

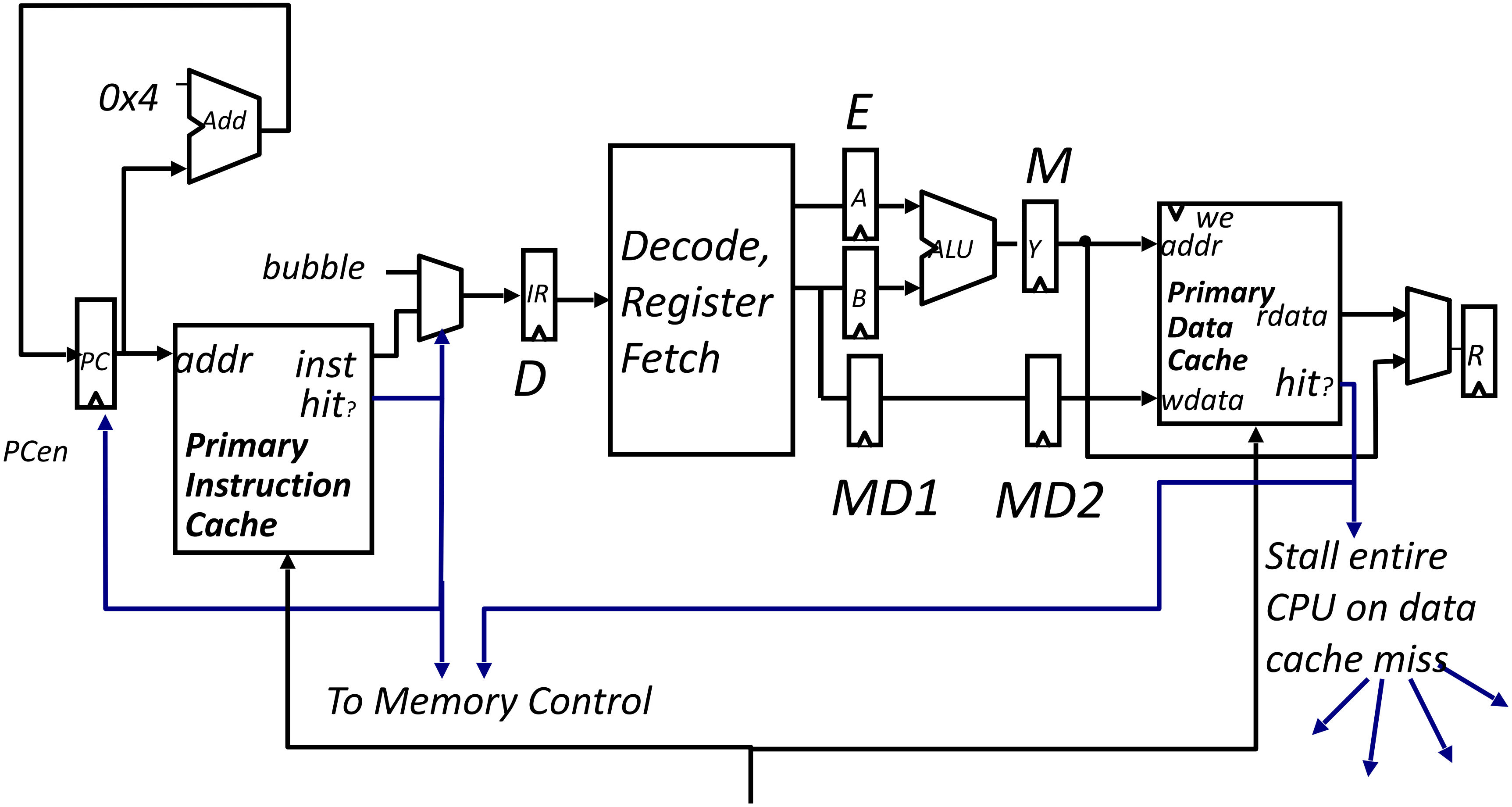
The power wall

Review from 61C

- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon.
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon.
- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.
- DRAM is slow but cheap and dense:
  - Good choice for presenting the user with a BIG memory system
- SRAM is fast but expensive and not very dense:
  - Good choice for providing the user FAST access time.

CPU-Cache Interaction

(5-stage pipeline)

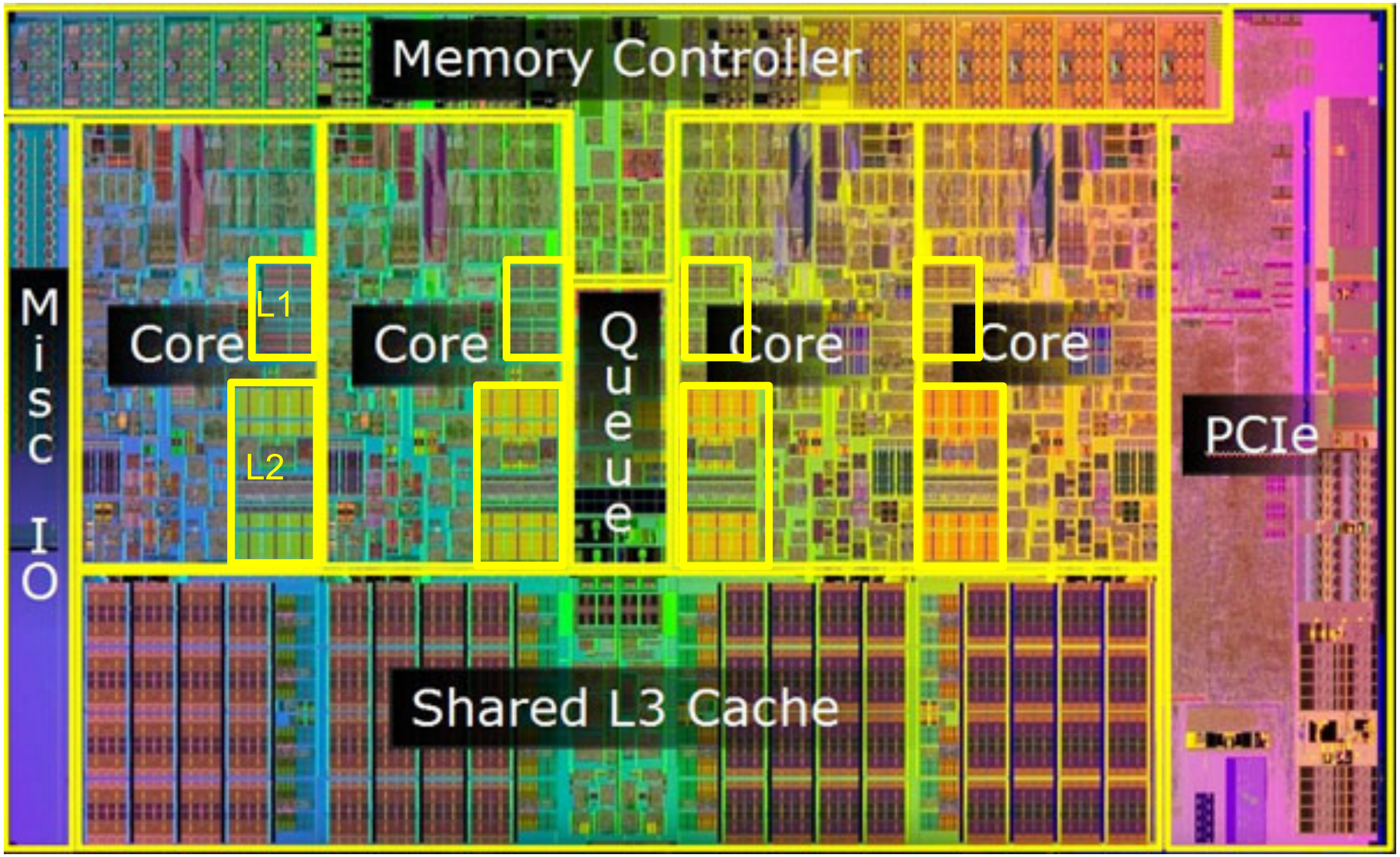


Stall entire CPU on data cache miss

To Memory Control

Cache Refill Data from Lower Levels of Memory Hierarchy

Nahalem Die Photo (i7, i5)

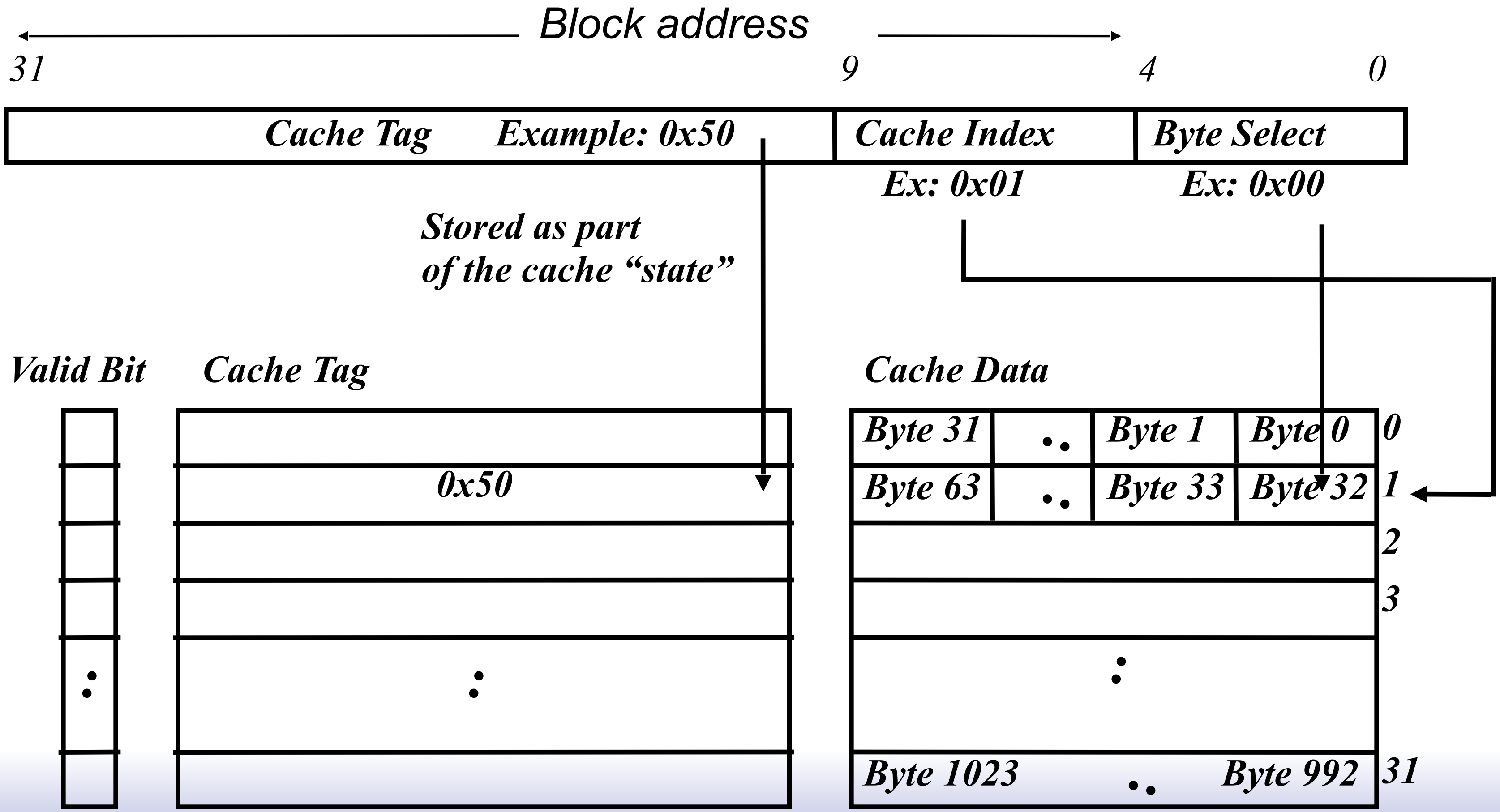


- Per core:
  - 32KB L1 I-Cache (4-way set associative (SA))
  - 32KB L1 D-Cache (8-way SA)
  - 256KB unified L2 (8-way SA, 64B blocks)
  - Common L3 8MB cache
- Common L3 8MB cache

Example: 1 KB Direct Mapped Cache with 32 B Blocks

For a 2^N byte cache:

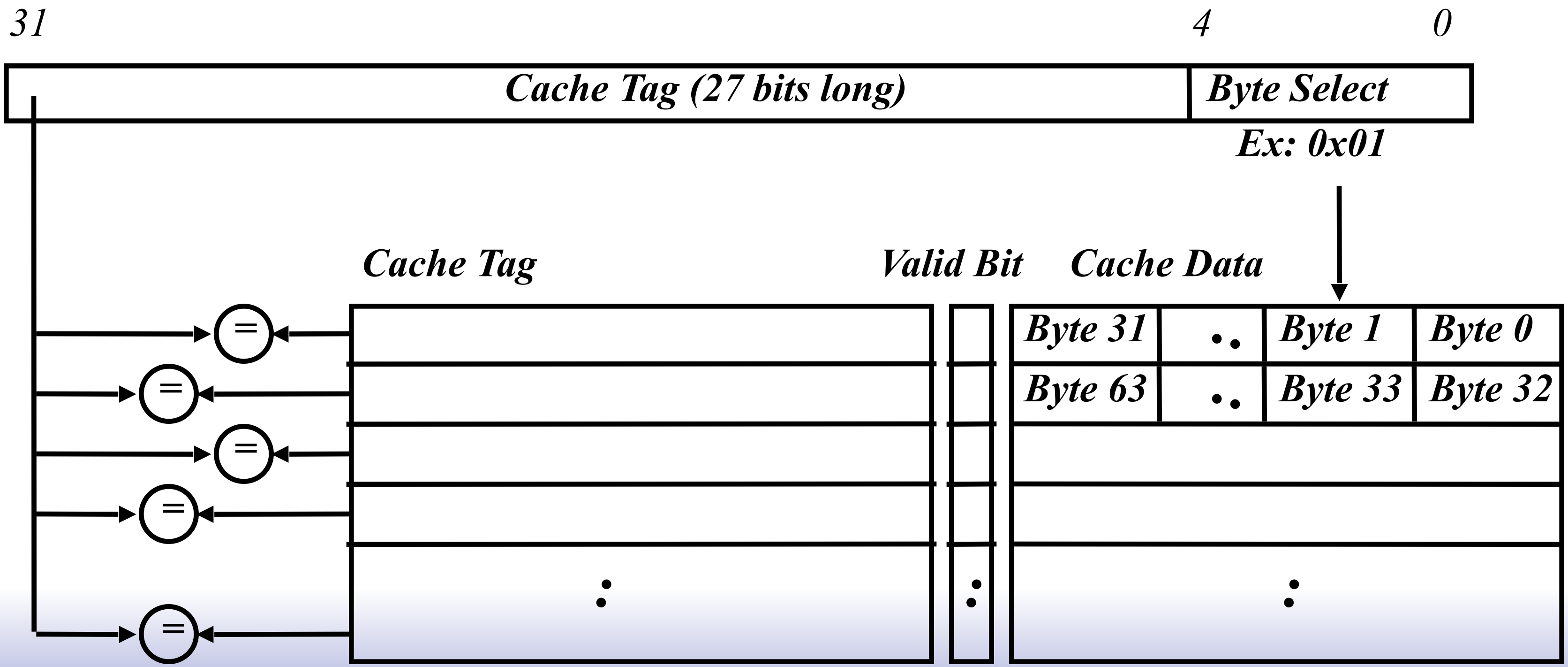
- The uppermost (32 - N) bits are always the Cache Tag
- The lowest M bits are the Byte Select (Block Size = 2^M)



Fully Associative

Fully Associative Cache

- No Cache Index
- For read, compare the Cache Tags of all cache entries in parallel
- Example: Block Size = 32 B blocks, we need N 27-bit comparators



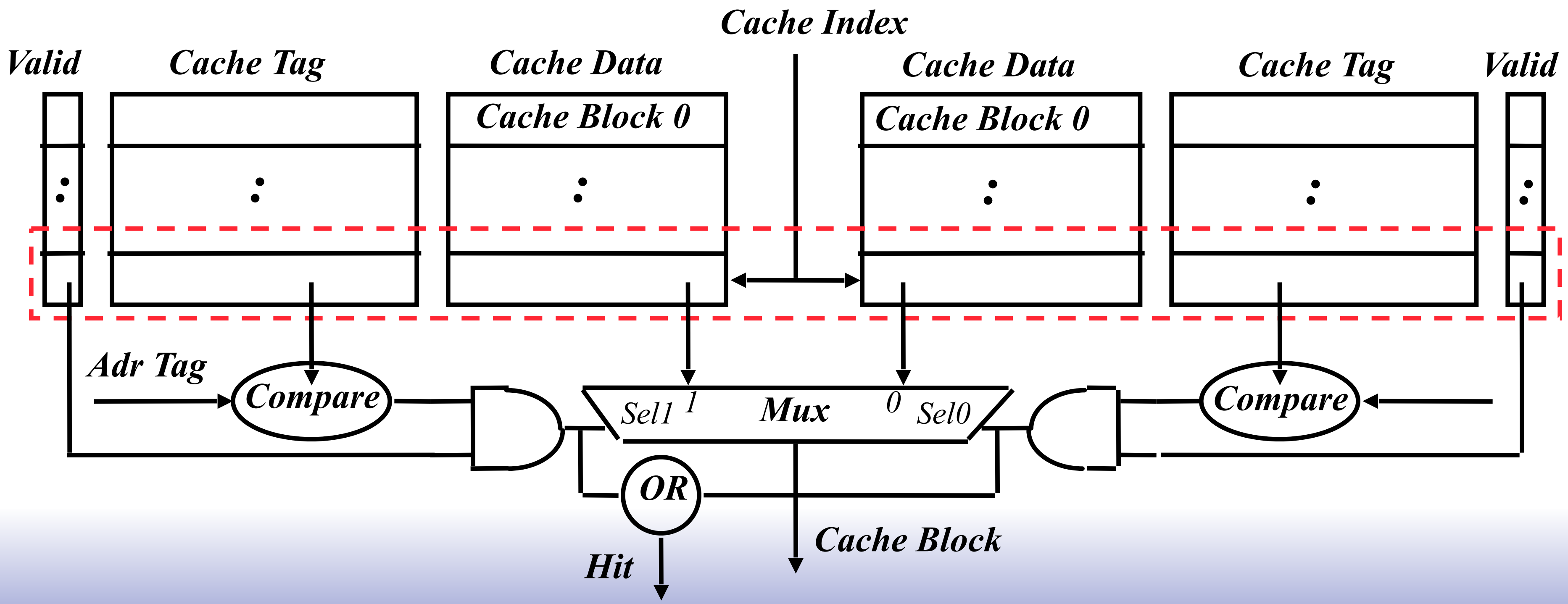
Set Associative Cache

N-way set associative: N entries for each Cache Index

- (N direct mapped caches operates in parallel)

Example: Two-way set associative cache

- Cache Index selects a "set" from the cache
- The two tags in the set are compared to the input in parallel
- Data is selected based on the tag result



RAM Blocks and the Project

Processor Design Considerations (FPGA Version)

- **Register File: Consider distributed RAM (LUT RAM)**
  - Size is close to what is needed: distributed RAM primitive configurations are 32 or 64 bits deep. Extra width is easily achieved by parallel arrangements.
  - LUT-RAM configurations offer multi-porting options - useful for register files.
  - Asynchronous read, might be useful by providing flexibility on where to put register read in the pipeline.
- **Instruction / Data Memories : Consider Block RAM**
  - Higher density, lower cost for large number of bits
  - A single 36kbit Block RAM implements 1K 32-bit words.
  - Configuration stream based initialization, permits a simple “boot strap” procedure.

Processor Design Considerations (ASIC Version)

- **Register File: use synthesized RAM**
  - At this size (1k bits) synthesized is competitive with dense RAM block
  - Latch-based instead of flip-flop-based would save on area.
  - Asynchronous read, might be useful by providing flexibility on where to put register read in the pipeline.
- **Instruction / Data Caches : Use generated dense Block RAM**
  - Higher density, lower cost for large number of bits
  - We will provide for you