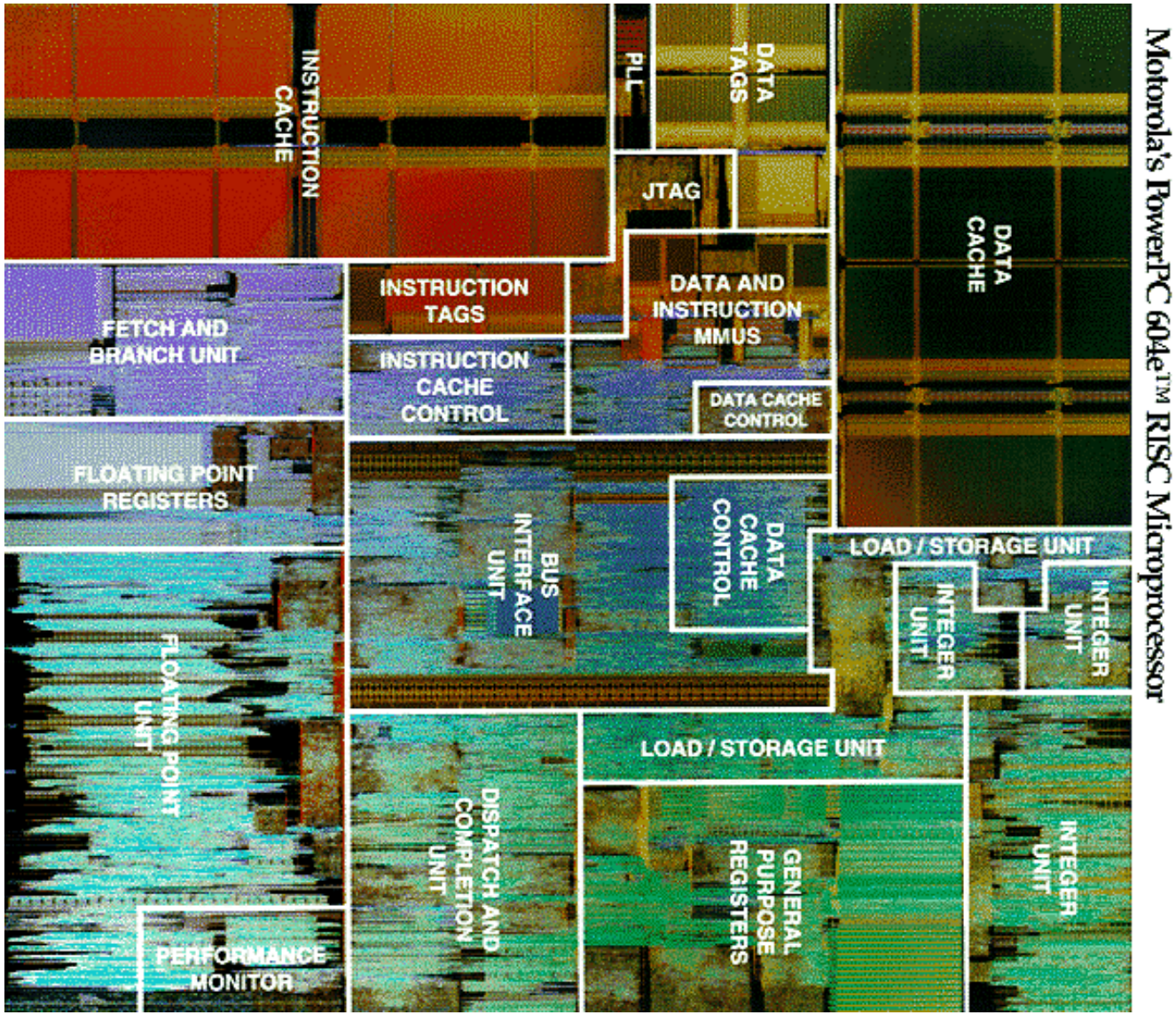
Memory

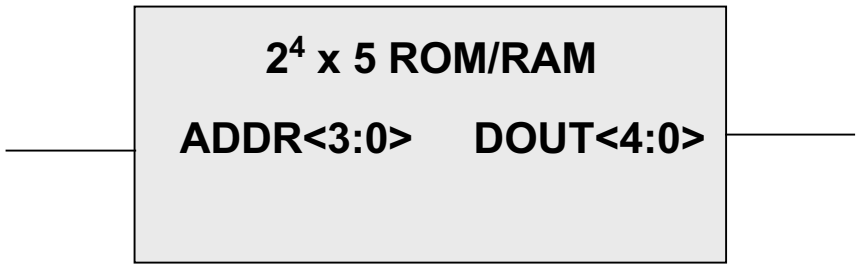
Memory

- Memory structures are critical to any large, complex digital design.



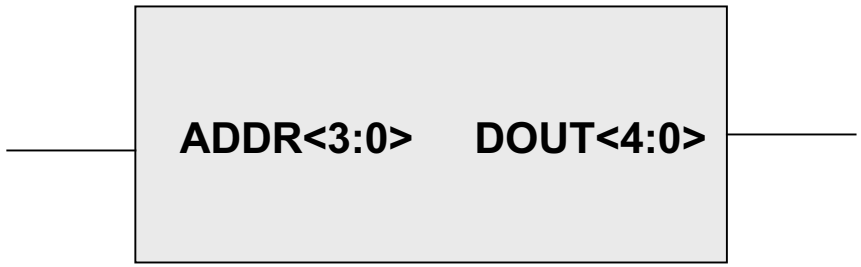
Memory

- Memory structures are crucial in digital design.
  - ROM, PROM, EPROM, RAM, SRAM, (S)DRAM, RDRAM,..
- All memory structures have an address bus and a data bus
  - Possibly other control signals to control output etc.
- E.g. 4 Bit Address bus with 5 Bit Data Bus



Memory

- Internal organisation
  - 'Lookup Table of values'
  - For each address there is a corresponding data output



ADDR	DOUT
0000	10101
0001	11111
1111	11001

16 possible address values with 5 bit output (16 x 5 ROM)

Memory

- Usually consider a repository for data or program code.
- Indexing of the data and ability to both read and write suggests a mailslot analogy.
  - Byte = 8 bits
  - Word = whatever data width you're using
- But,
- especially when considering 'read only',
- 'Lookup Table' ≡ 'Truth Table"

ADDR	DOUT
0000	10101
0001	11111
1111	11001

ROM

- k-input, n-output Read Only Memory (ROM) could be
  - 2^k x n element lookup table
  - k-input, n-output truth table
  - k-input, n-output combinatorial logic block
  - **DOUT(0) = $\overline{\text{ADDR(3)}}.\overline{\text{ADDR(2)}}.\overline{\text{ADDR(1)}}.\overline{\text{ADDR(0)}}$ + $\overline{\text{ADDR(3)}}$.ADDR(2)..**

- *Any* k-input, n-output combinatorial block is available
- Benefits
  - Configure once PCB complete
  - Flatten complex logic hierarchy to faster design
  - Simple to create using high level language - no minimisation.

ROM - VHDL

- VHDL Code
- Various forms of coding possible
  - E.g. `if addr = "0000" then ...`
  - Or `case addr is, when "0000" =>` (better)
- A neater approach is to use a table of constants
  - See next slide !
- Could also easily add in further control signals etc.

ROM - VHDL

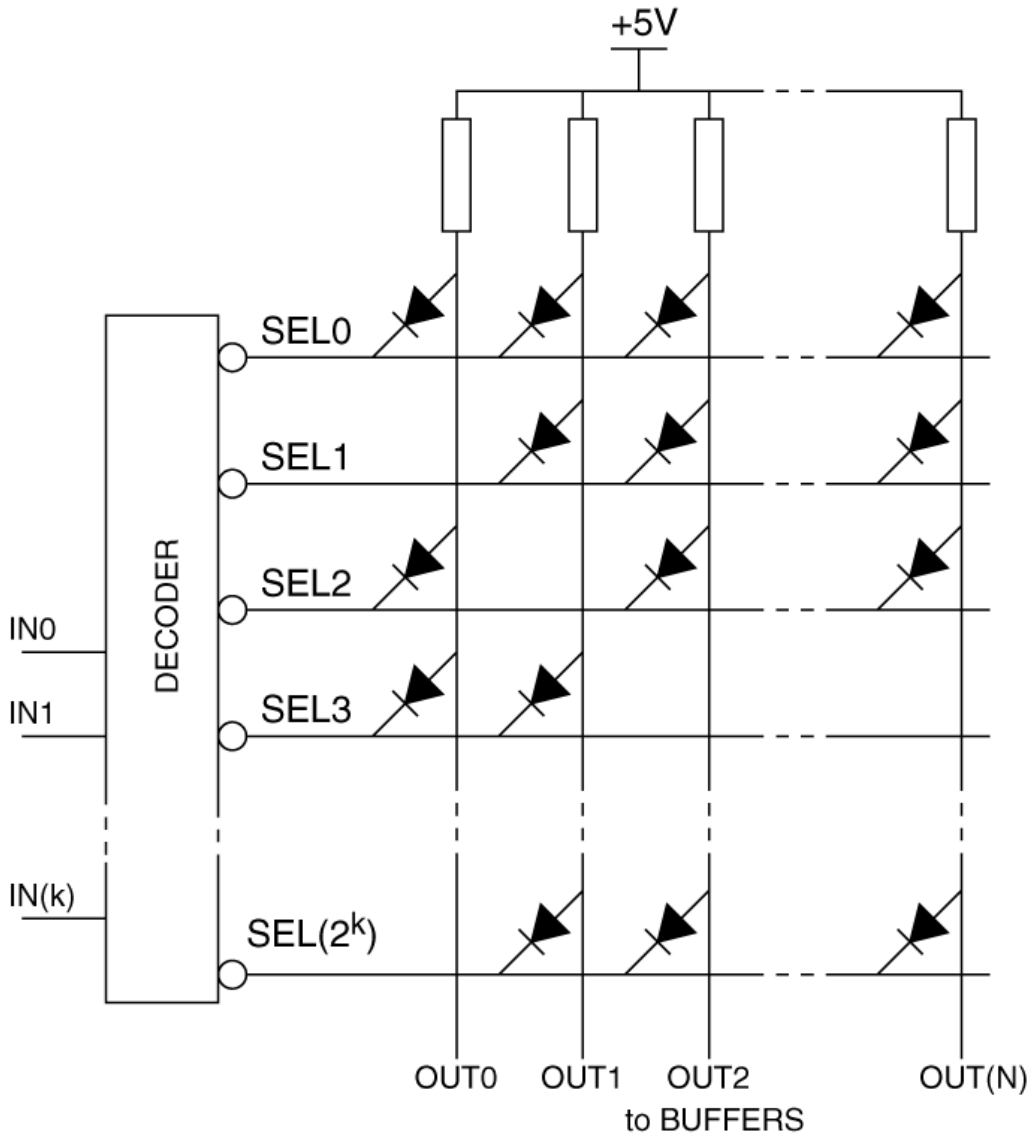
```
use ieee.numeric_std.all;
architecture BEHAVIOR of ROM is
  type romtable is array (0 to 15) of std_logic_vector(4 downto 0);
  constant romdata : romtable :=
    (“10101”,
     “11111”,   etc.)
    );
begin
  process (ADDR)
  begin
    DOUT <= ROMDATA (to_integer(unsigned(ADDR)));
  end process;
end BEHAVIOR;
```

ROM - VHDL

- Result after synthesis is simply a combinatorial logic implementation of the ROM
  - i.e. DOUT(0) = ADDR(0) + ADDR(1) …
- In practical terms, memory structures can be implemented on Silicon much more efficiently by use of technology specific implementation
  - E.g. I need a 16 x 4 ROM with the values ….
  - As noted above - use high level language to calculate the values.
  - For simulation/ small structures the above approach is feasible !

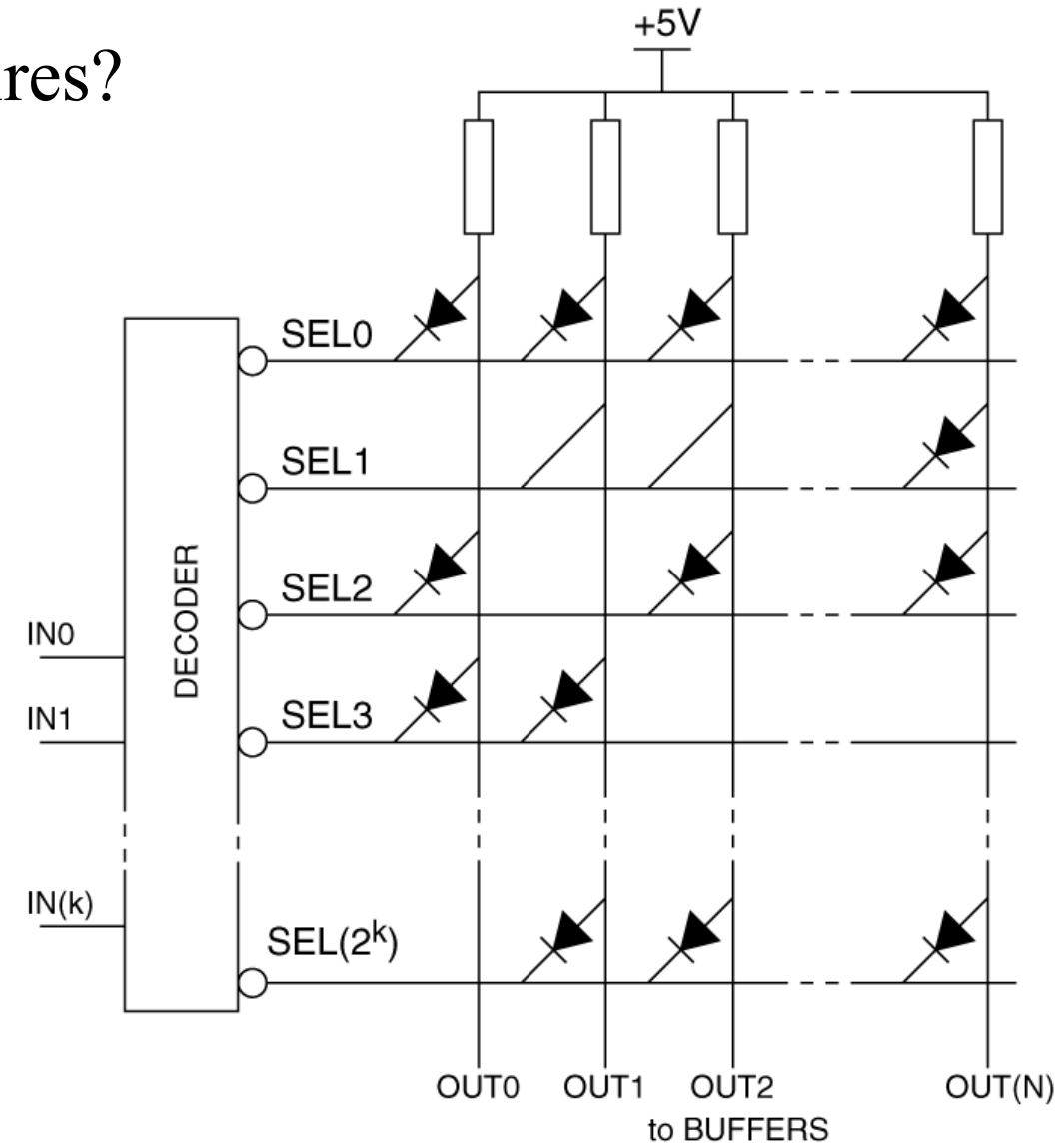
ROM - Internal Structure

- Historically



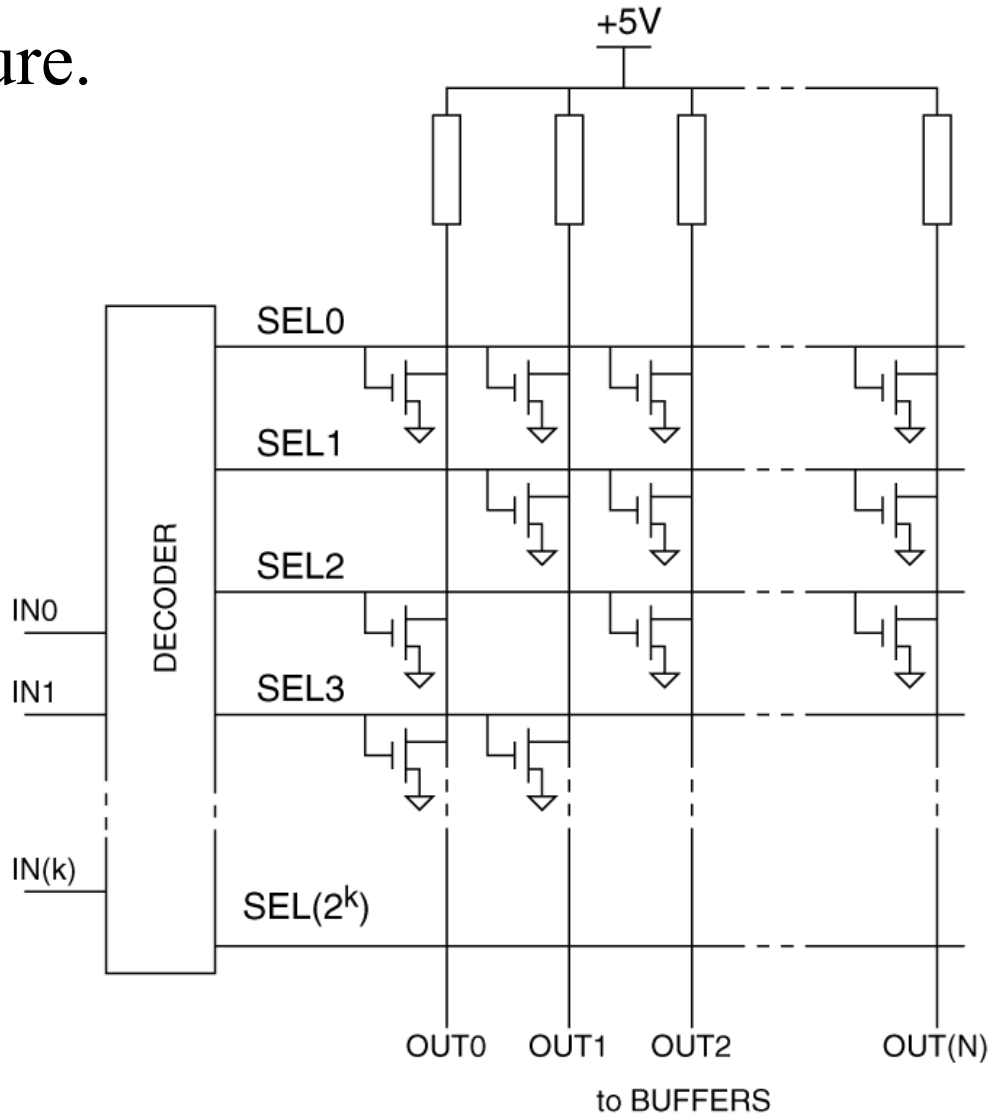
ROM - Internal Structure

- Why not just wires?



ROM - Internal Structure

- Modern structure.



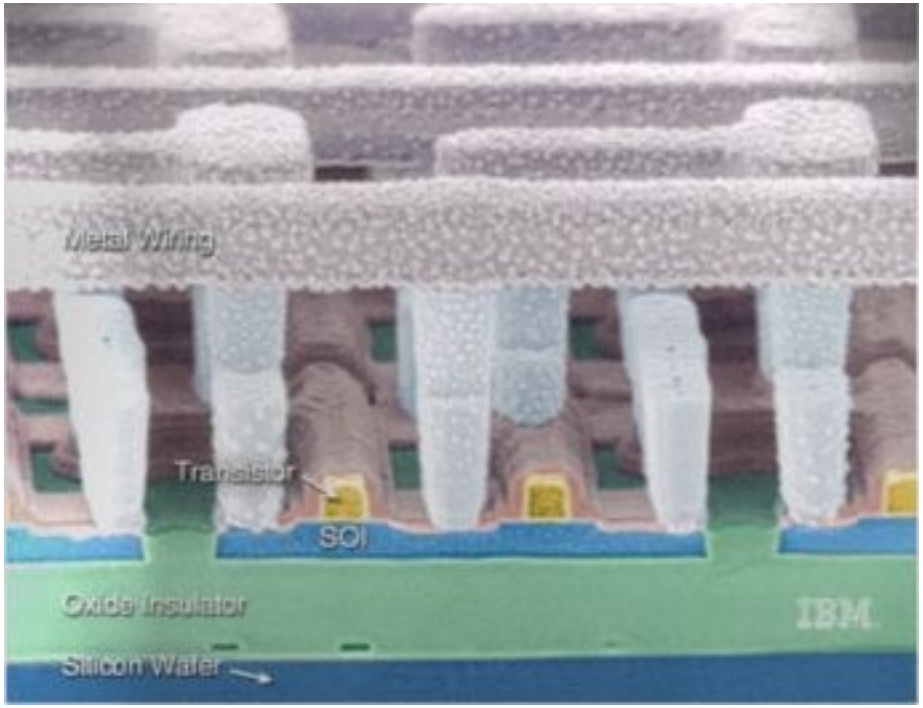
ROM - Internal Structure

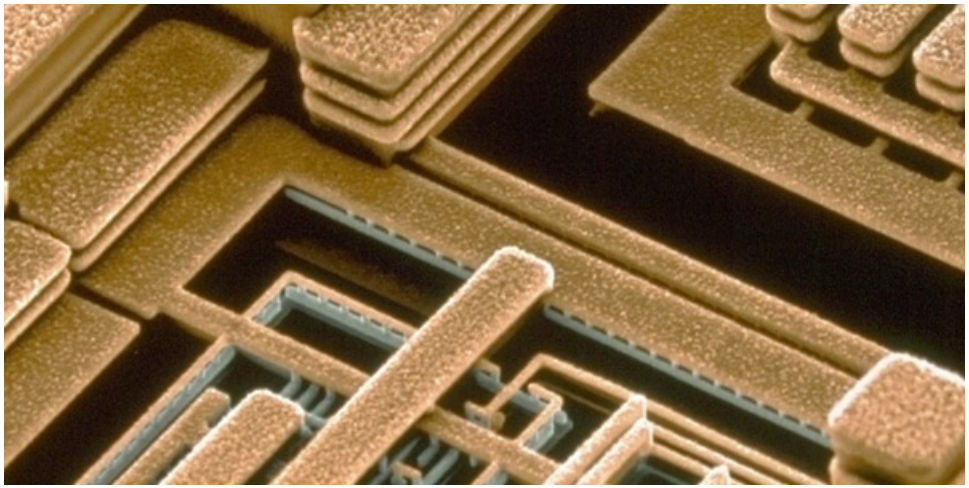
- The problem of decoding
  - A k-input ROM requires a k-to-2^k decoder
  - Such a decoder requires 2^k, k-input NAND gates, k buffers and k inverters, each with fanout of 2^{k-1}.
  - 1Mbyte memory would obviously require over 1 million 20 input NAND gates, and 40 buffers/inverters with fanout of half a million, or a long (delay ridden) buffer chain.
  - Ugh.
- And just how does such a beast fit into the system timing.

- Answer these questions when dealing with RAM.

Types of ROM - mask ROM

- ‘Connections’ between word and bit lines are made by altering one or more interconnect masks in the fabrication process.





Types of ROM - mask ROM -2

- Ideal for high volume, low cost production.
- Turn around on mask production and NRE (non recurring engineering) costs make them pointless for prototyping.

Types of ROM - PROM

- ‘Connections’ between word and bit lines destroyed by vaporising fusible links.
- Bipolar transistors typically used in these devices, so relatively high power, but fast
  - nothing much faster than a wire in a microelectronic circuit!
- OTP (one time programmable) - obviously.
- Cheaper than EPROM or EEPROM and so often used in short production runs, or where the contents of the ROM may be altered right up to product launch but then set in stone.

Types of ROM - PROM - 2

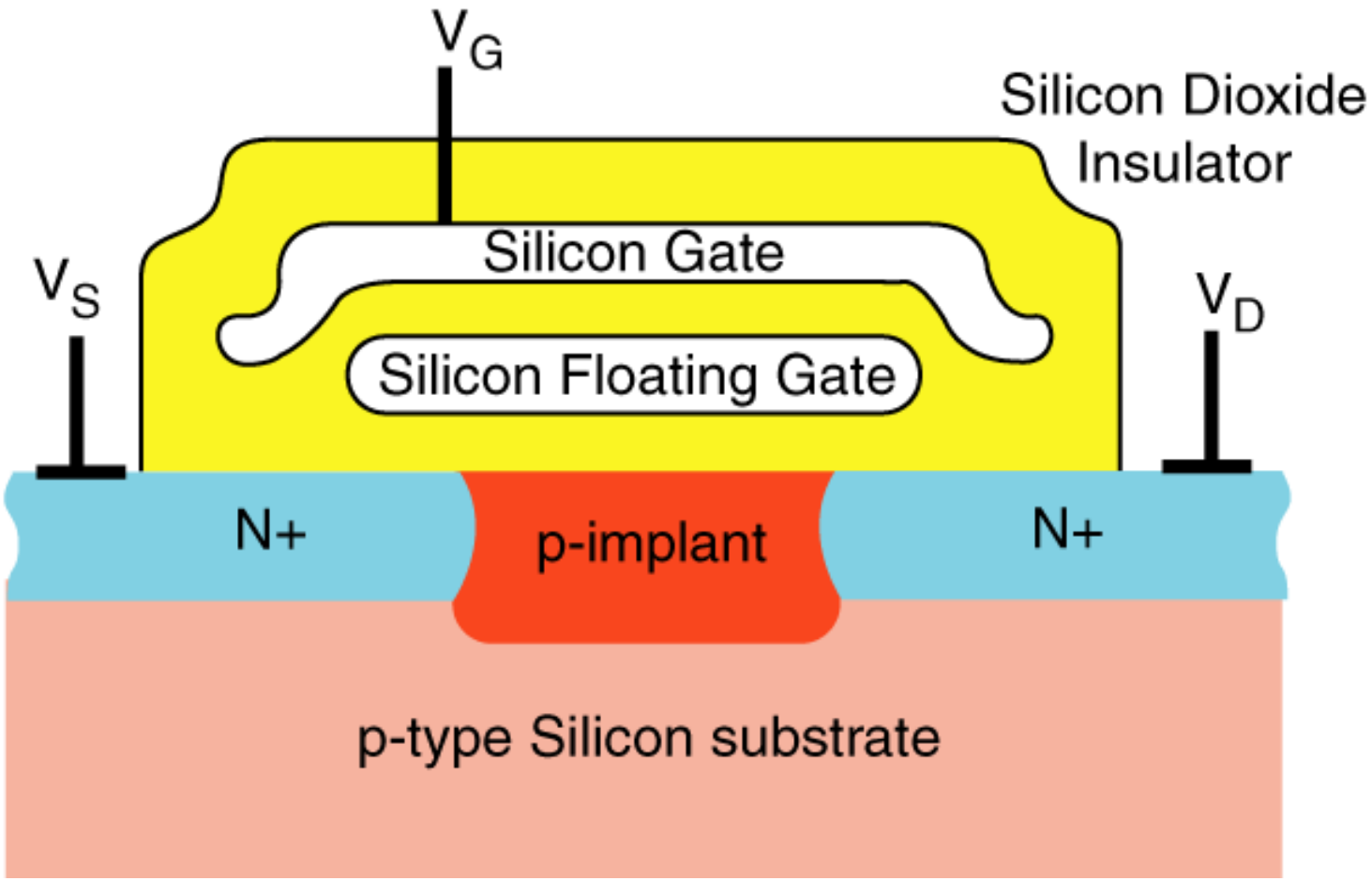
- Technology can be employed in the look up tables / fuse maps of OTP PLDs or, more rarely FPGAs.

Types of ROM - EPROM

- EPROM: Erasable, Programmable, Read Only Memory
- EEPROM: Electrically Erasable, Programmable, Read Only Memory
- flash EEPROM: a hybrid of the two.

- typically today ‘EEPROM’ and ‘flash EEPROM’ are both applied to flash EEPROM technology.

Types of ROM - EPROM - 2



- Non volatile - 70% of charge remains after 10 years

Types of ROM - EPROM - 3

Device	EPROM	EEPROM	flash EEPROM
Channel-Floating Gate	100 nm	10 nm	10 nm
Programme	Avalanche Breakdown	Fowler-Nordheim tunnelling	Avalanche Breakdown
Programming Time	micro s	milli s	micro s
Erasure	UV light	Fowler-Nordheim tunnelling	Fowler-Nordheim tunnelling
Erasure Time	kilo s	milli s / word	milli s / word
Programme/Erasure Cycles	100	10 000	10 000

E/EPROM Applications

- Firmware for embedded systems
  - Microcontrollers - staggeringly huge market…
  - modems, graphics cards, etc.
- Embedded OS software for palmtops, set-top boxes,…
- Bootstrap loaders
- Relatively cheap to produce.
- As markets expand, economies of scale make cheaper.
- OTP devices are now usually EPROMs without quartz lids.

E/EPROM Applications - 2

- Technology is often employed in look up tables / fuse maps of PLDs or FPGAs
- Both PROMs and EEPROMs are readily available from 16 kbyte to 8 Mbyte configurations.
- Multiple flash EEPROMs can be packaged into flash card solid state memories.
  - Can lever 40 mp3 encoded CD tracks into the largest sold today.

Random Access Memory

Static, Dynamic,
Synchronous
and Asynchronous

RAM - Overview

- RAM (random access memory)
  - read and write to any location given a valid address
  - Historically term had more meaning when tape drives and punched cards were used for mass storage.
  - Today hard disks are ‘almost’ random, and modern RAM is ‘not completely’ random.

RAM - Overview

- SRAM (static)
  - Formed from internal latches - 6 transistors per bit.
  - Latch will store information as long as power supplied.
  - Inherently synchronous.
  - Integrable.
  - Fast , 12 ns access time direct from Farnell
  - Fast, 4 ns access time in ECL from Cypress
  - On chip caches, off chip caches, HD caches,…

RAM - Overview

- DRAM (dynamic)
  - Storage of charge on a capacitor gated by a transistor
  - 1 ish transistor per bit.
  - High packing density, large cheap memory,
  - Cheap + economies of scale = very cheap. Commodity item.
  - Less integrable
  - Seriously faster than hard disk, 70 ns direct from Farnell
  - Main memory
  - These days most DRAM is also synchronous SDRAM

RAM - Overview

- SDRAM today
- PC100, k=64, 64 bit datapath at 100MHz
  - How many bytes per second throughput / bandwidth?
  - How many ns per clock cycle at 100MHz ?
  - Didn't we just say it took 70 ns to supply a byte from Farnell ?
- PC133, k=64, 64 bit datapath at 133MHz
  - How many bytes per second throughput / bandwidth ?

RAM - Trends

- RDRAM (Rambus DRAM)
- PC600, k=16, at 266MHz, clocked on each edge!
- PC800, k=16, at 400MHz, clocked on each edge!
  - Bandwidths ?
- History
  - Nintendo 64, Intel i820 chipset (bleugh), Playstation 2, *not* the Nintendo Dolphin, and practically no-one else.

RAM - Trends

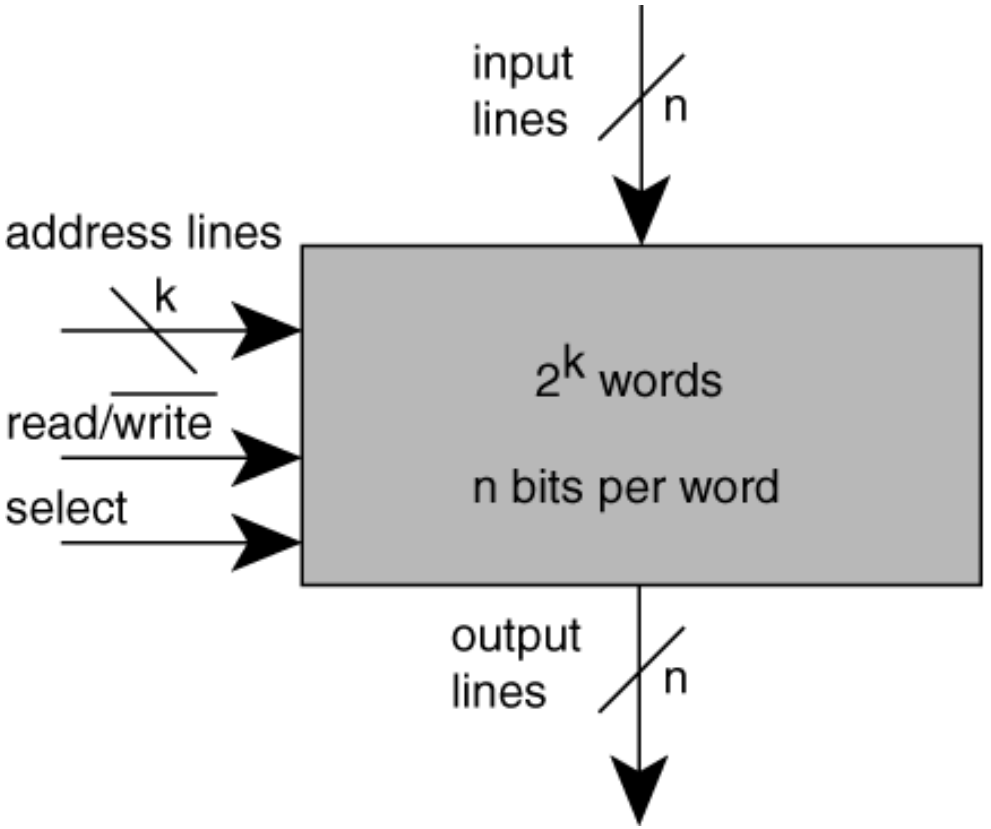
- RDRAM Benefits
  - Fewer pins, so smaller PCB
  - Internal guts does good look-ahead for streamed memory access.
- But
  - Proprietary ($$)
  - However more difficult initial design due to high speed
  - Streamed memory access ≠ random access memory
  - Big heatsinks (not really built for your pretty Mac Cube)

RAM - Trends

- DDRDRAM (Double data rate DRAM)
- PC1600, k=64, at 100MHz, clocked on each edge!
- PC2100, k=64, at 133MHz, clocked on each edge!
- Benefits
  - Cheaper
  - Lower power consumption
  - Better for truly random access (but not streamed)
  - Already used in graphics cards
  - AMD and almost everyone else apart from Intel .

RAM - interface

- Unless specifically designed (dual port), memory cannot be read from and written to at the same time.
- Instead of read and write, use r/^w (or write enable) and chip select signals.

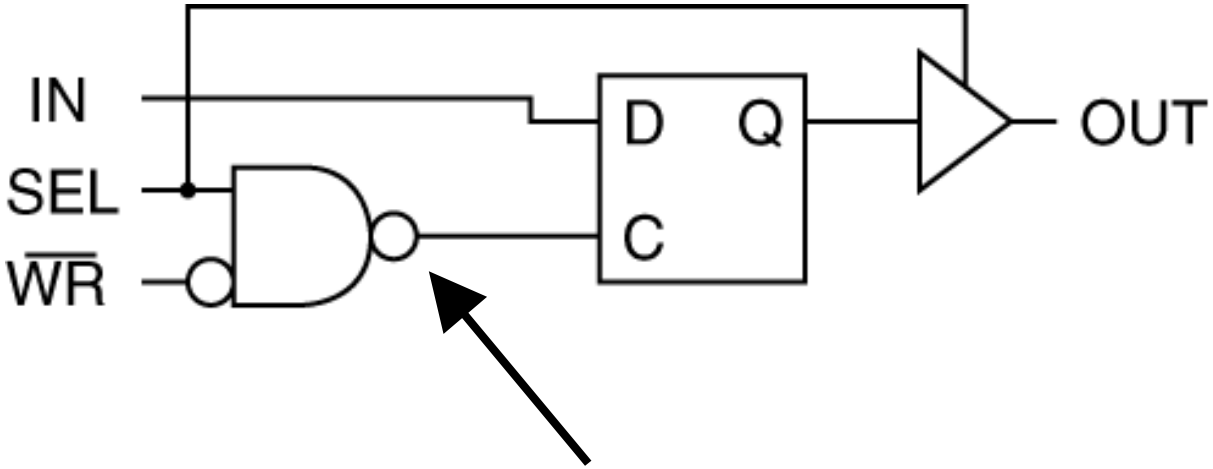


Where are we ?

- VHDL
- VHDL
  - Combinatorial logic
  - Counters, registers, state machines
- Memory
  - ROM, SRAM, DRAM
- Arithmetic Logic Unit (ALU)
- Datapaths
- Sequencing and Control

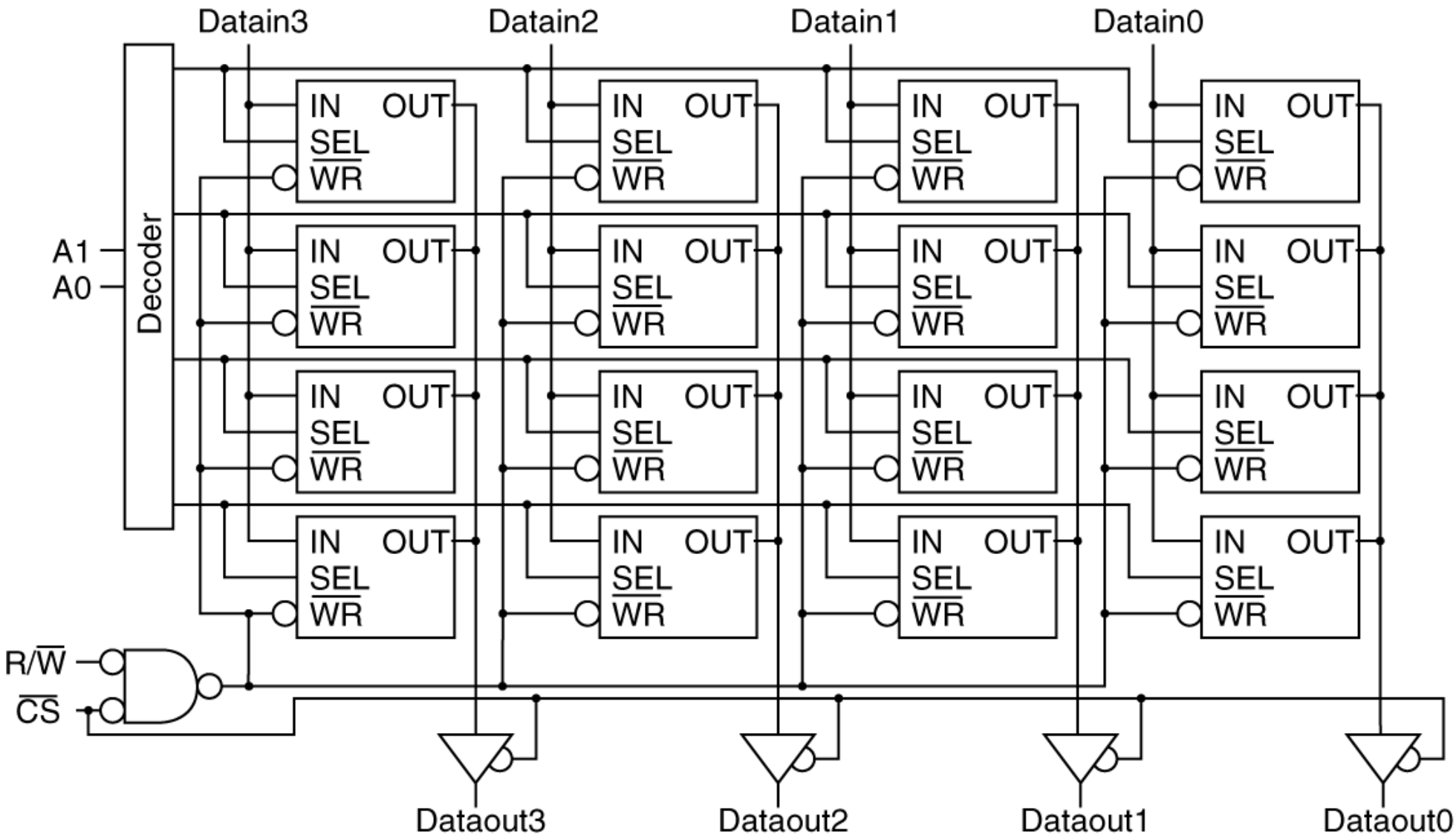
async SRAM - internal structure

- Single SRAM cell



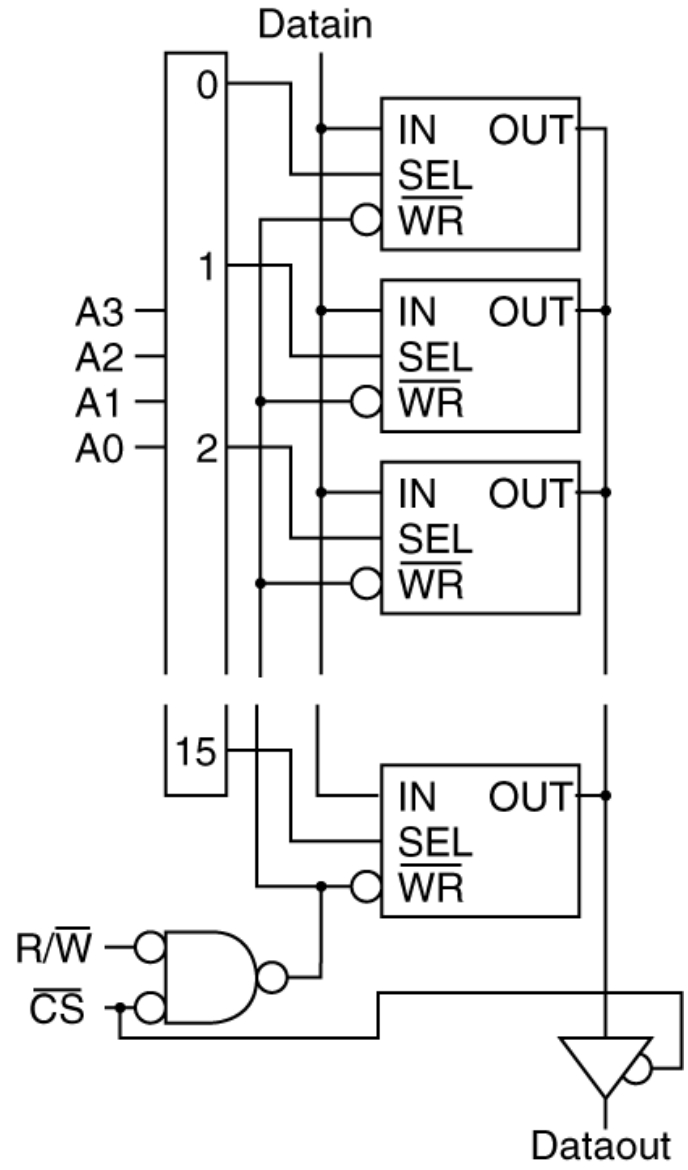
BUG!!

async SRAM - internal structure

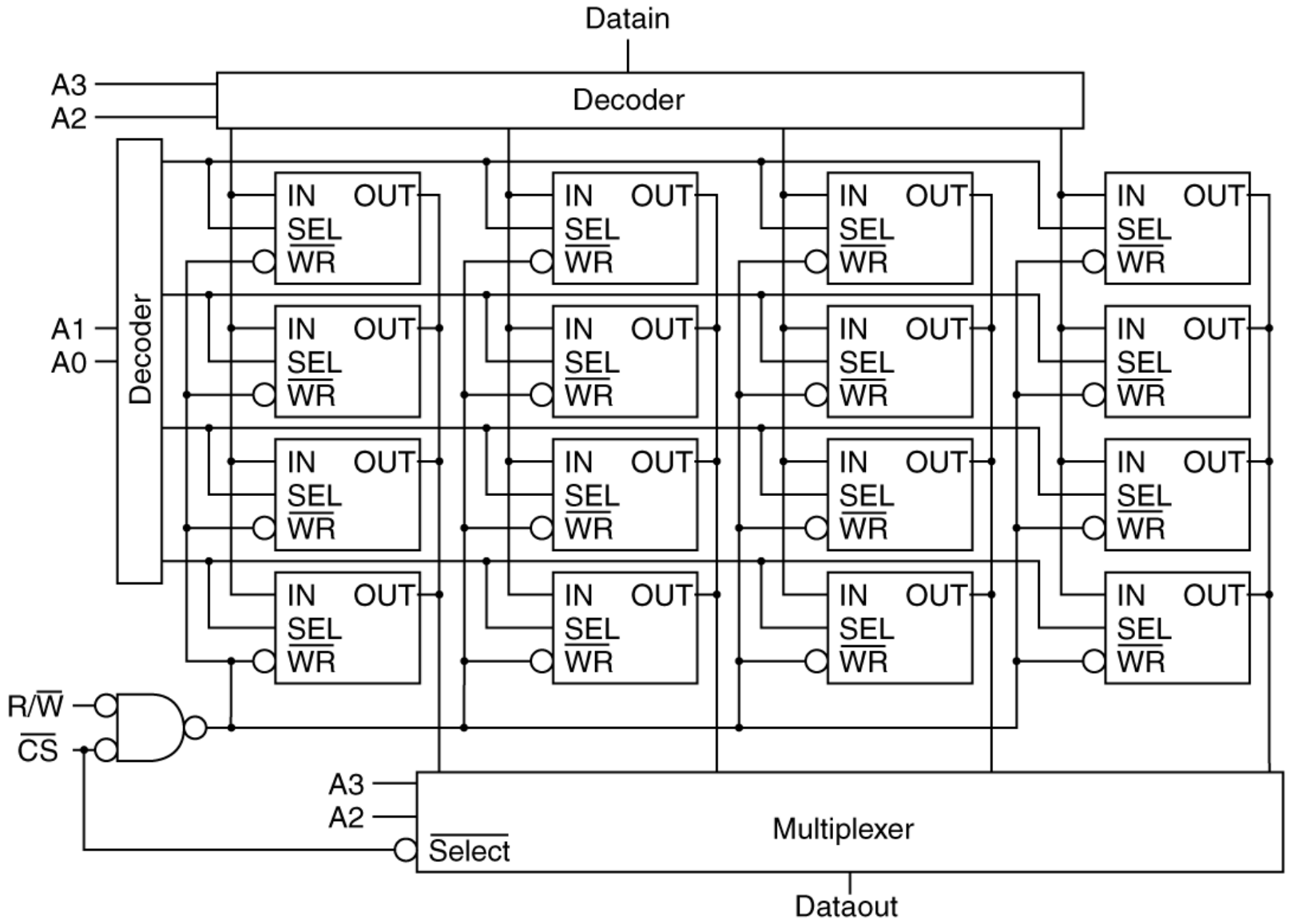


async SRAM - 16 ×1

- The wrong way



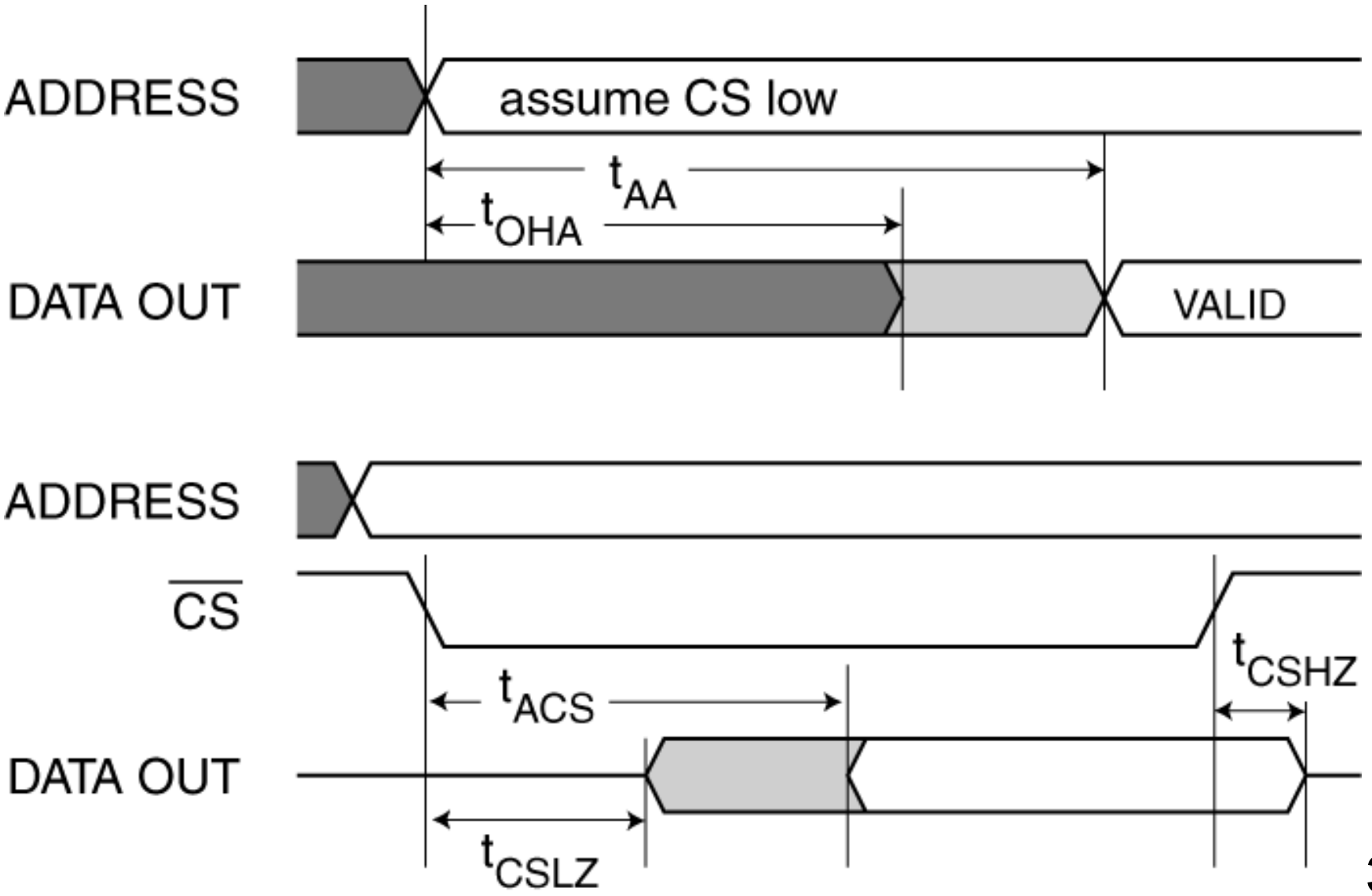
async SRAM - 16 ×1



RAM - timing

- Read process
  - Choose desired word by applying address
  - Ensure r/^w high
  - Push chip select high and wait for a clock edge
- Read timing (CY7C102A, 256K ×4 Static RAM)
  - t_{AA}, Address to valid data (12ns max)
  - t_{OHA}, Data hold from address change (3 ns min)
  - t_{ACS}, chip select to data valid (12 ns max)
  - t_{CSLZ}, chip select to low impedance (3 ns min)
  - t_{CSHZ}, chip select to high impedance (6 ns max)

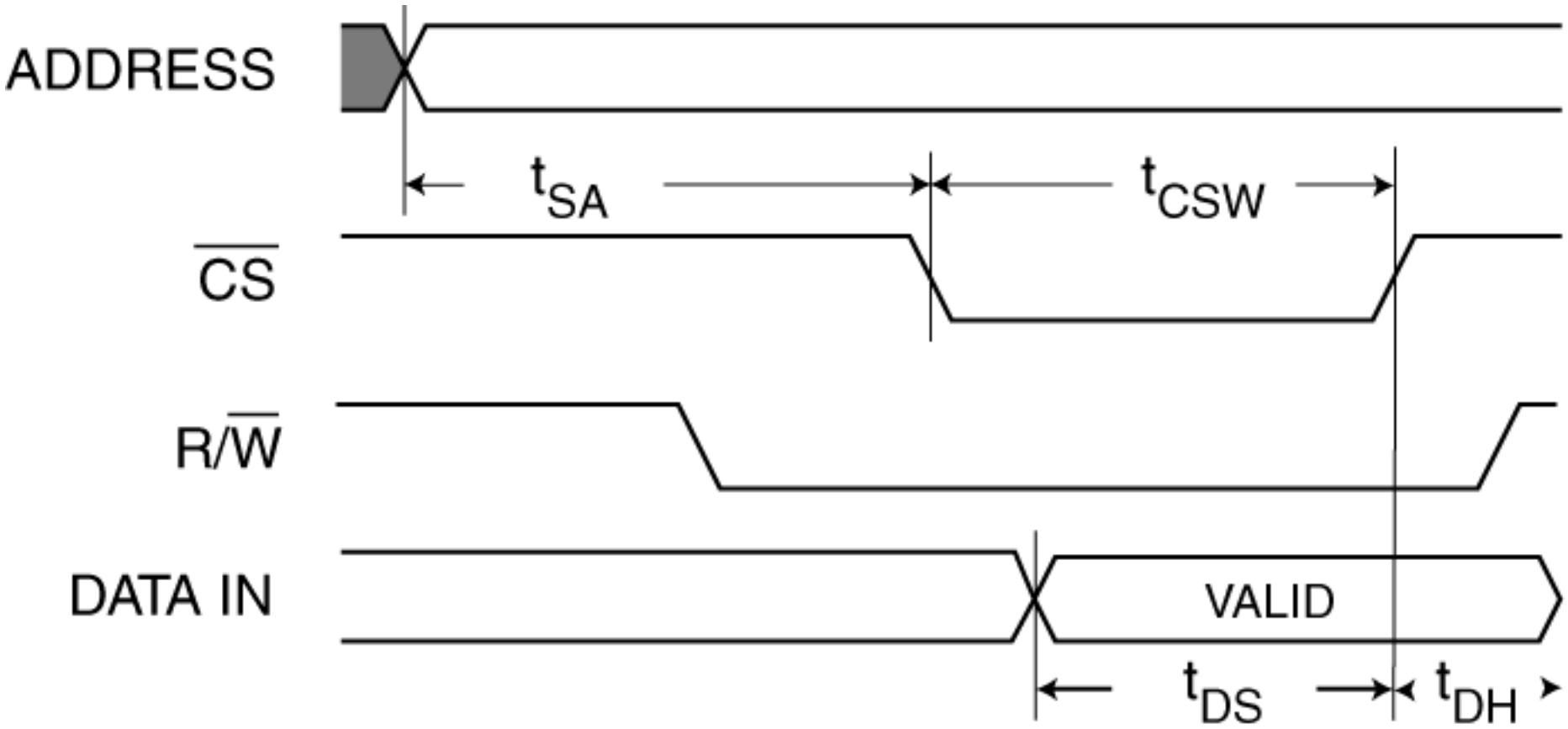
RAM - timing


ADDRESS
assume CS low
t_{AA}
t_{OHA}
DATA OUT
VALID
ADDRESS
$\overline{CS}$
t_{CSHZ}
t_{ACS}
DATA OUT
t_{CSLZ}

RAM - timing

- Write process
  - Choose desired word slot by applying address to address lines
  - Push r/^w low
  - Push chip select low and apply data before it's raised high again.
- Write timing (CY7C102A)
  - t_{AS}, address set to write start (0 ns min)
  - t_{CSW}, chip enable pulse length (10 ns min)
  - t_{DS}, data setup to wite end (7 ns min)
  - t_{DH}, data hold to write end (0 ns min)
- all this assuming CS controlled write

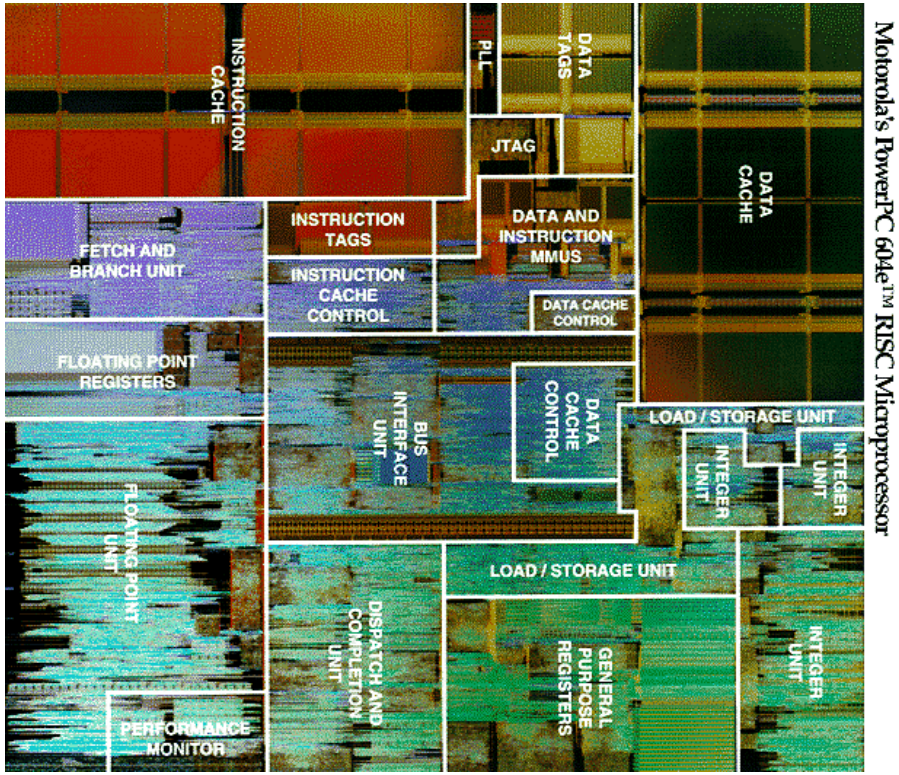
RAM - timing


ADDRESS
t_{SA}
t_{CSW}
$\overline{CS}$
R/$\overline{W}$
DATA IN
VALID
t_{DS}
t_{DH}

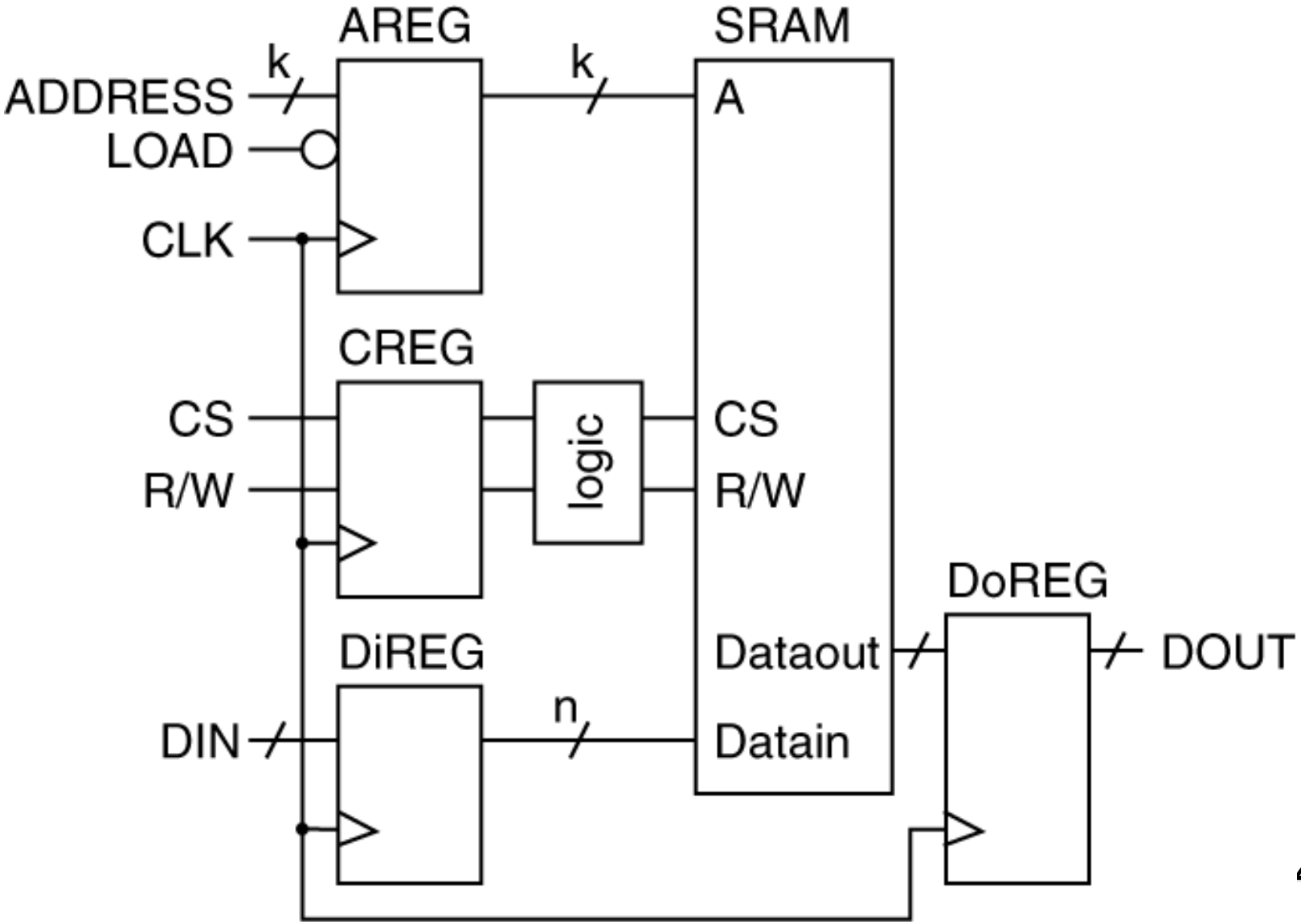
RAM - timing

- Read process similar to ROMs
- Latches - from schematic obviously no clock
- But aren't all our devices supposed to be synchronous ?

  – Edge triggered flip-flops would double silicon area
  – Designers just accept the hassle in real systems.



synchronous SRAM - collaring/glue

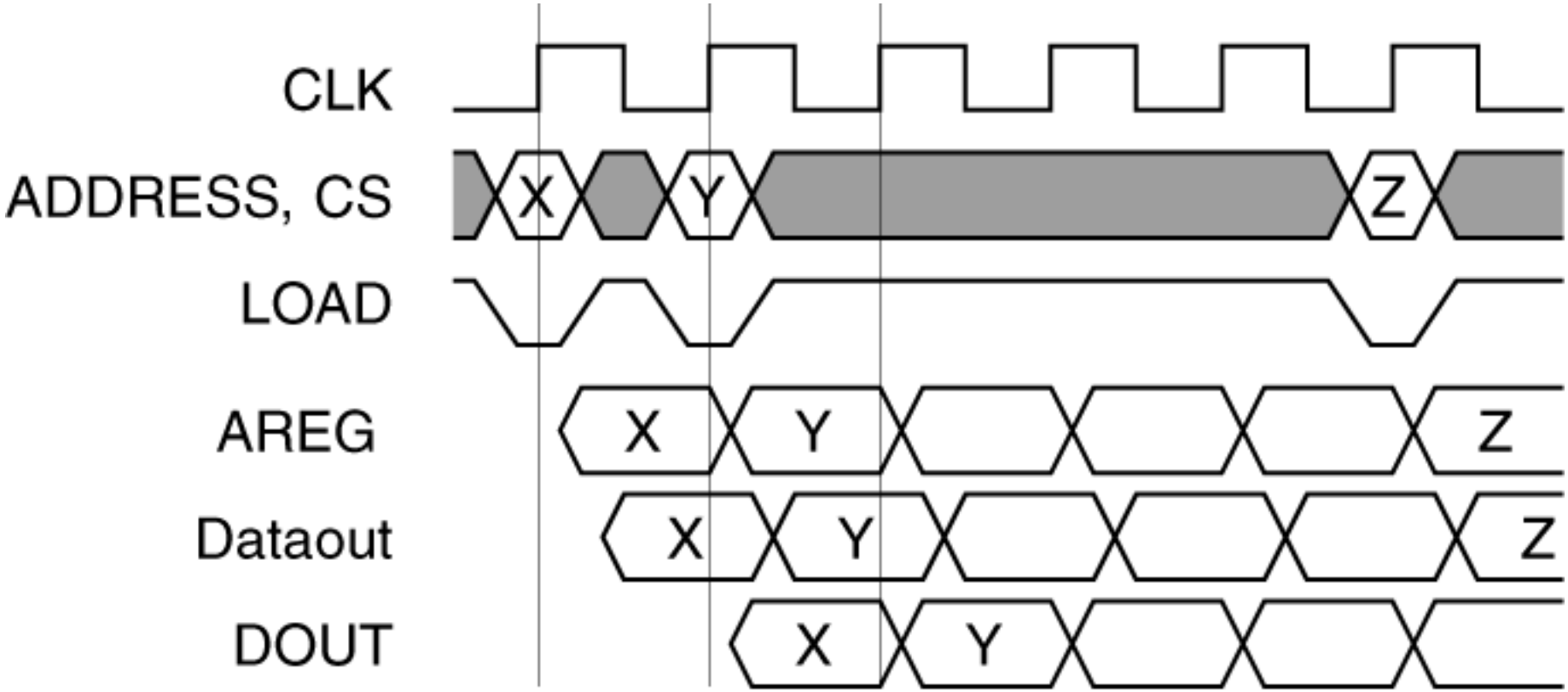


sync SRAM

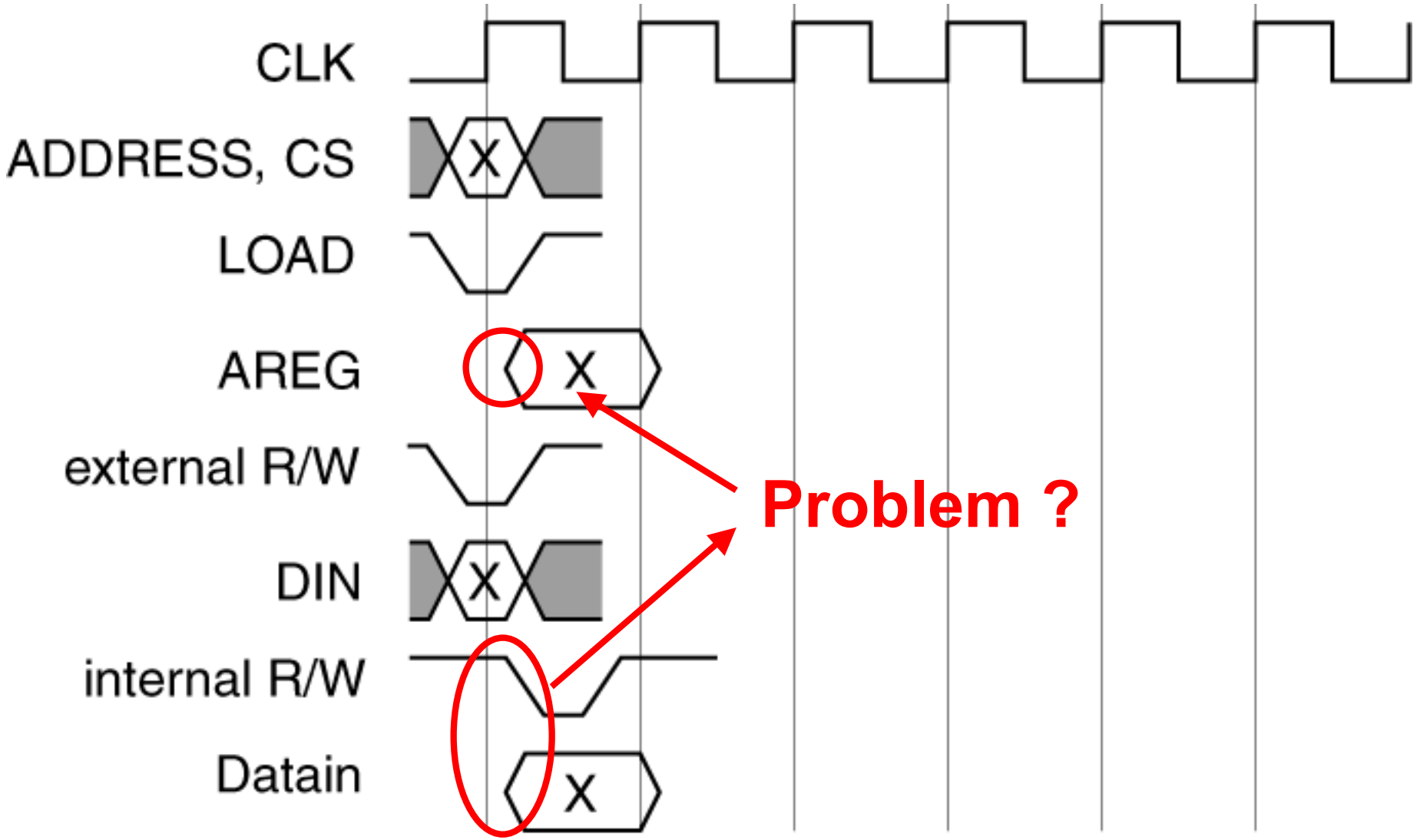
- Follow the same line as Wakerly ‘Digital Design Principles and Practices’ - revise from §10.3

- Chapter 6 of Mano & Kine ‘Logic and Computer Design Fundamentals’ is fine as well.

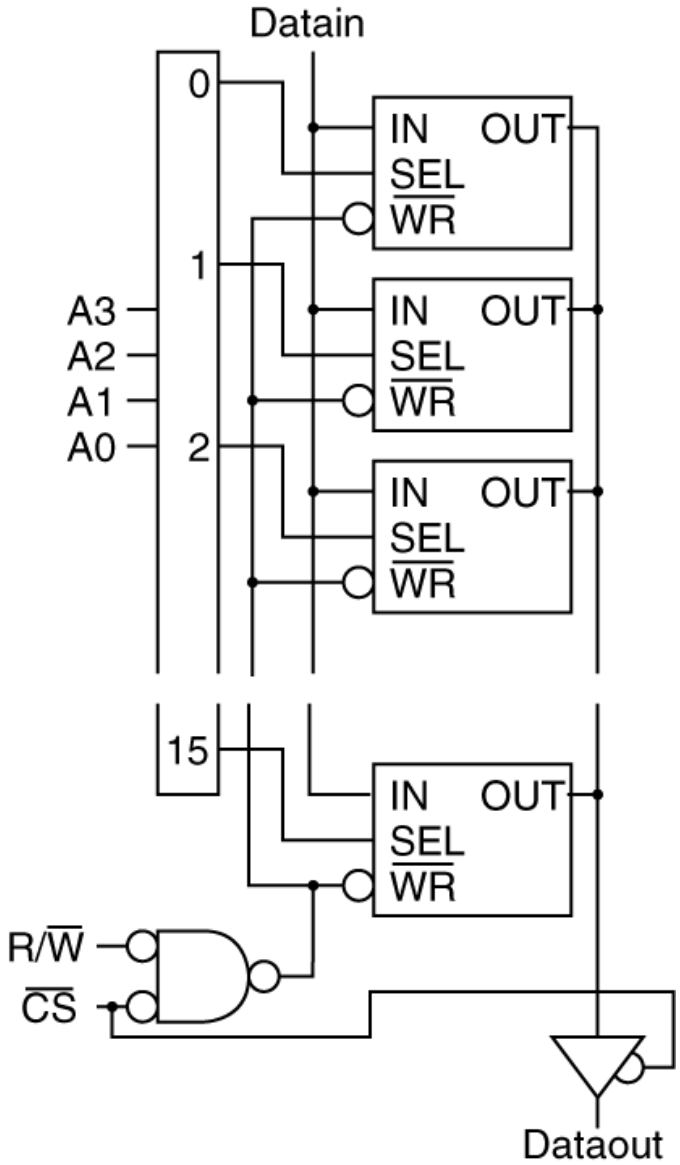
sync SRAM - read operations

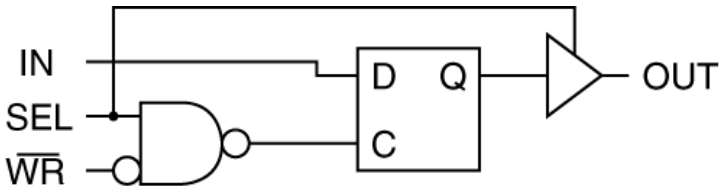


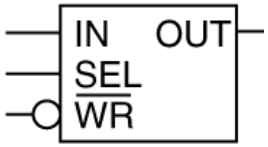
sync SRAM - write operations



sync SRAM - write operations

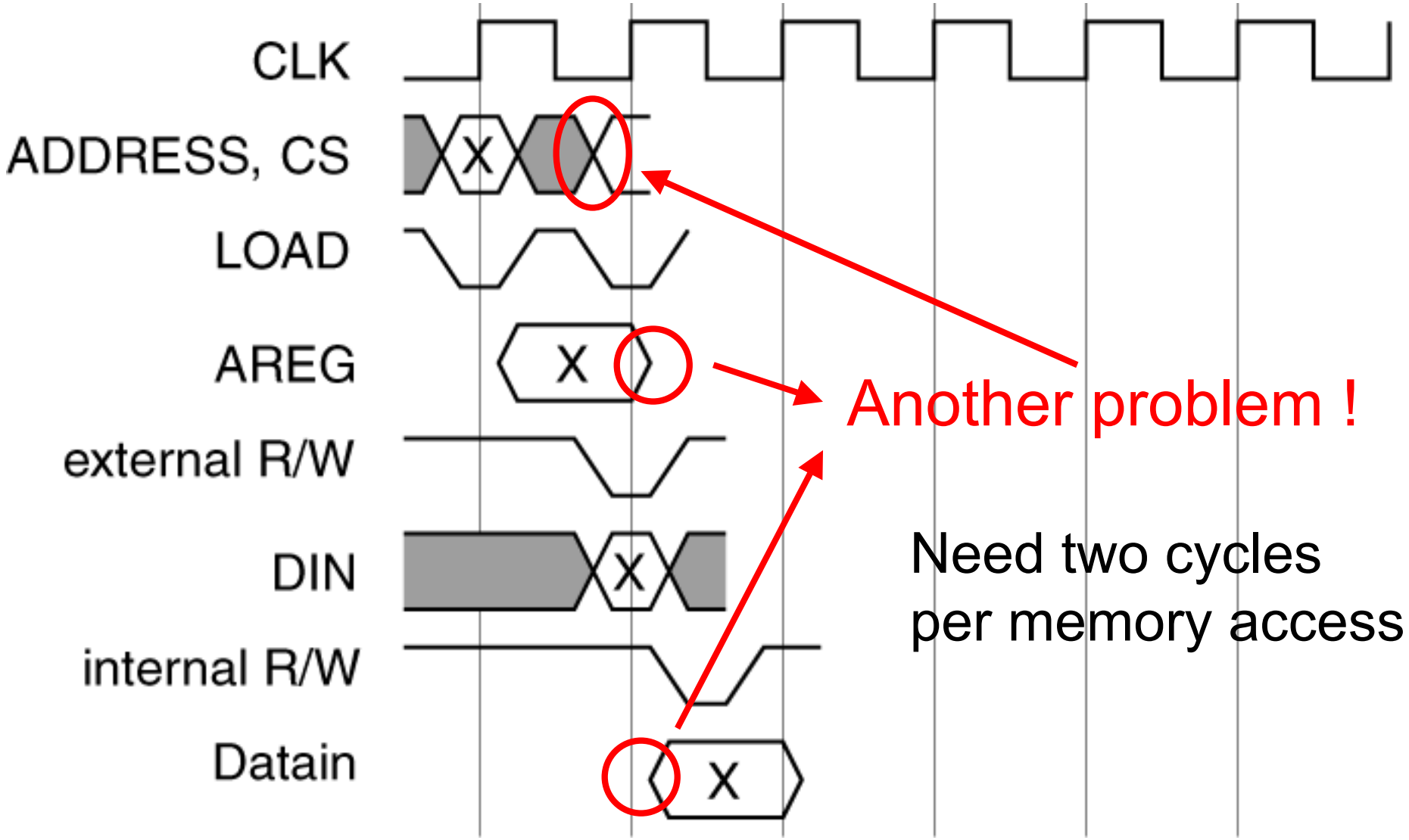




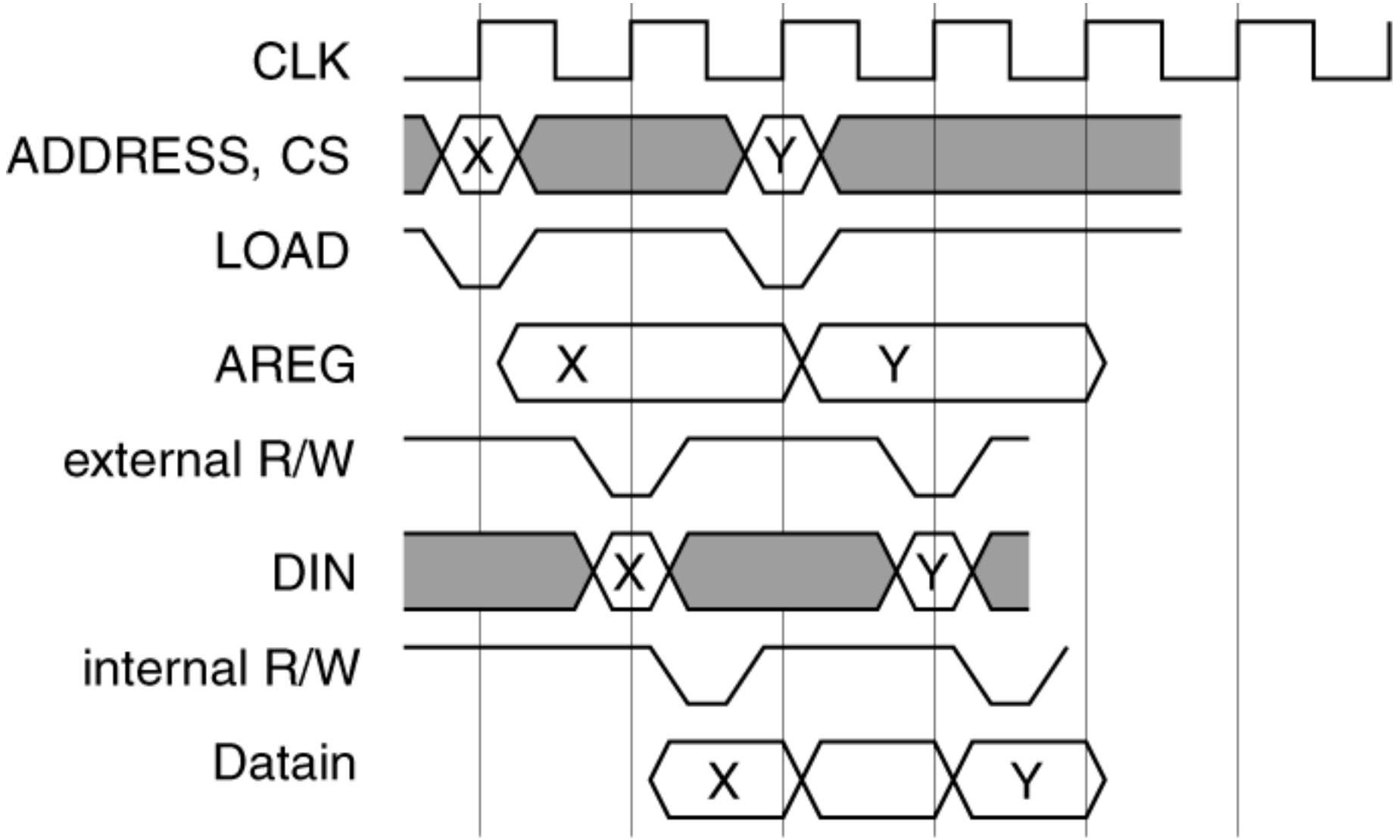


- Address delays are naturally longer than data delays.
- Need to add delay sections to keep the timing correct.
- Perfectly possible to do but in most memory systems not needed - two stage operation is usually helpful for the rest of the system.

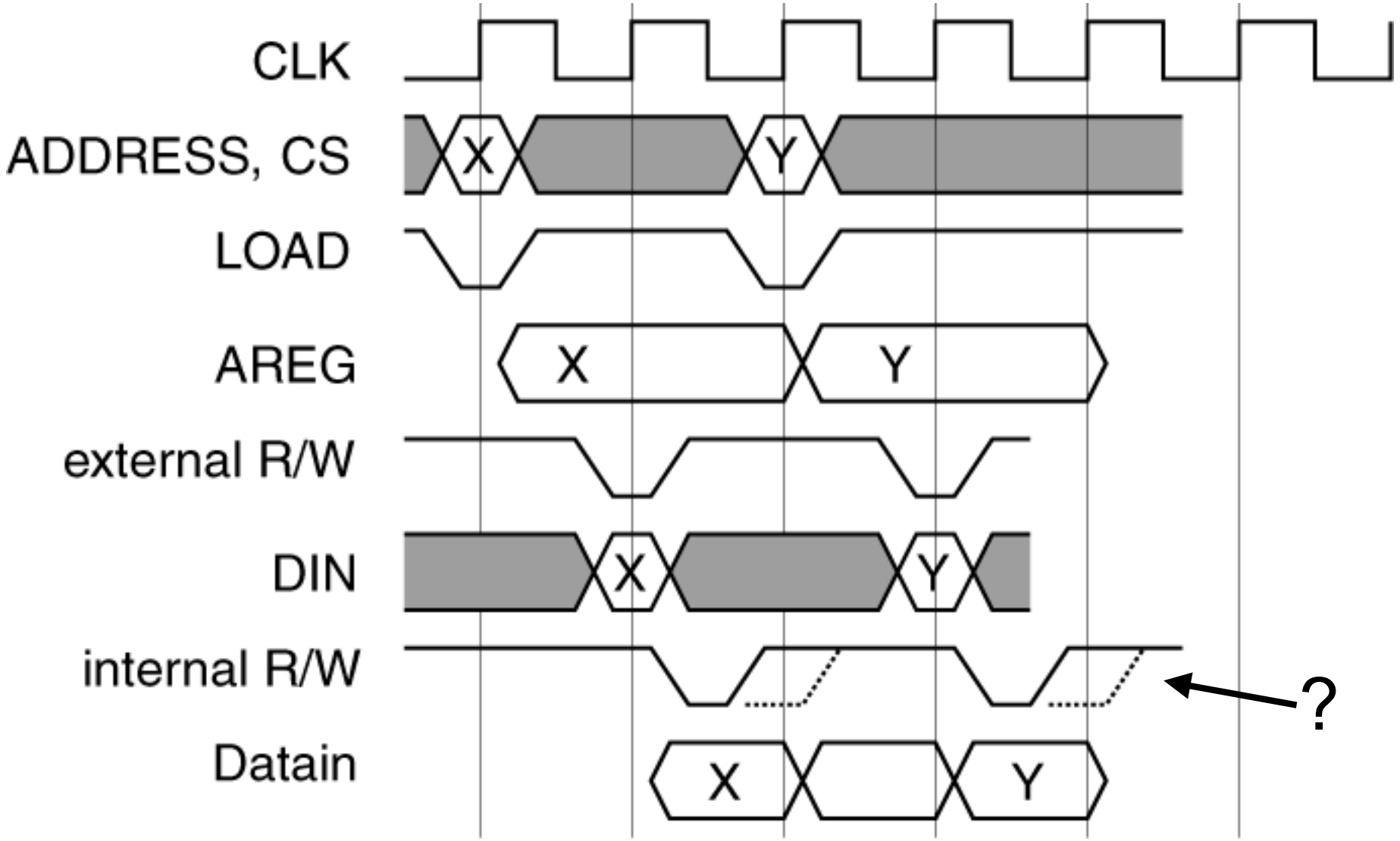
sync SRAM - write operations



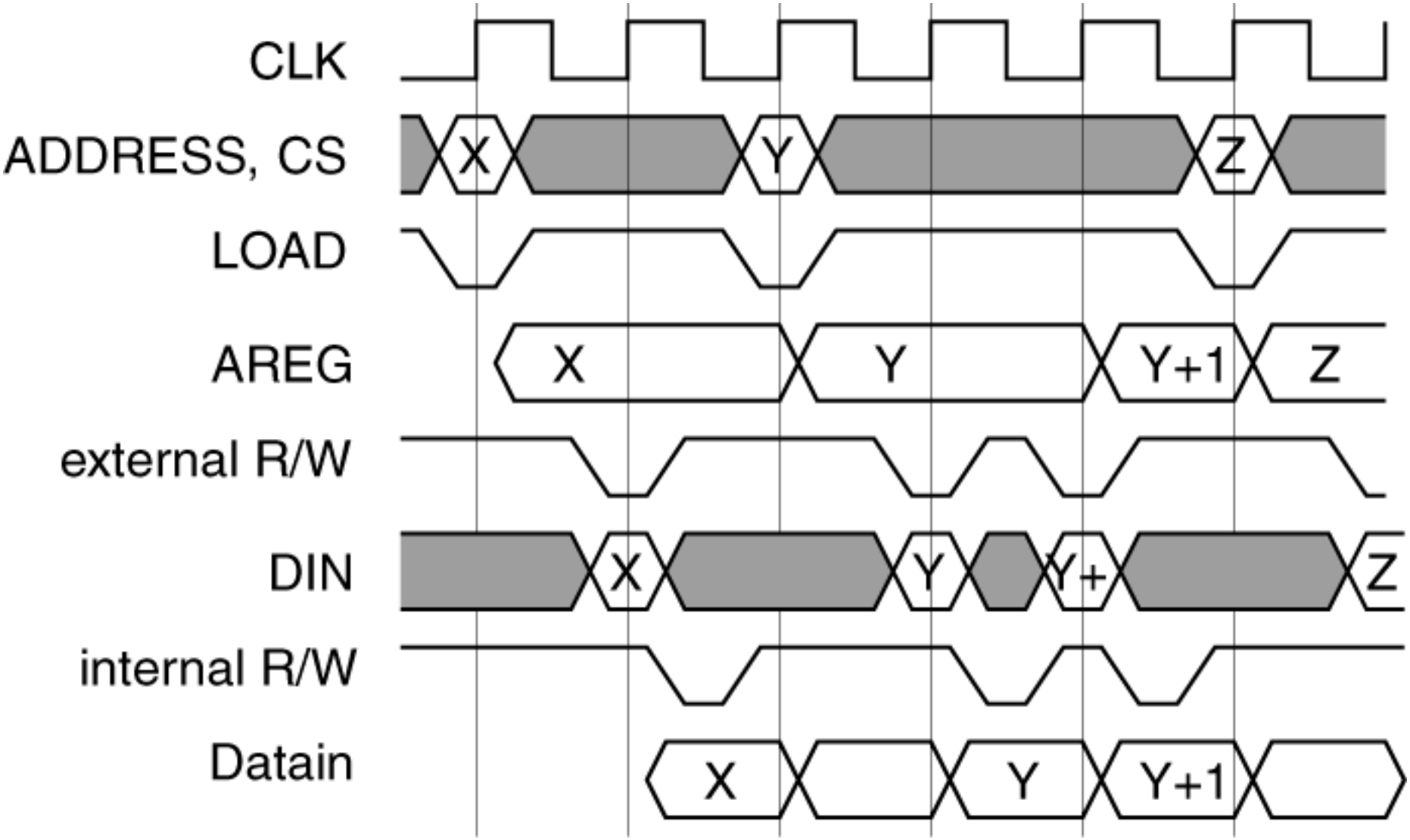
sync SRAM - write operations



sync SRAM - write operations



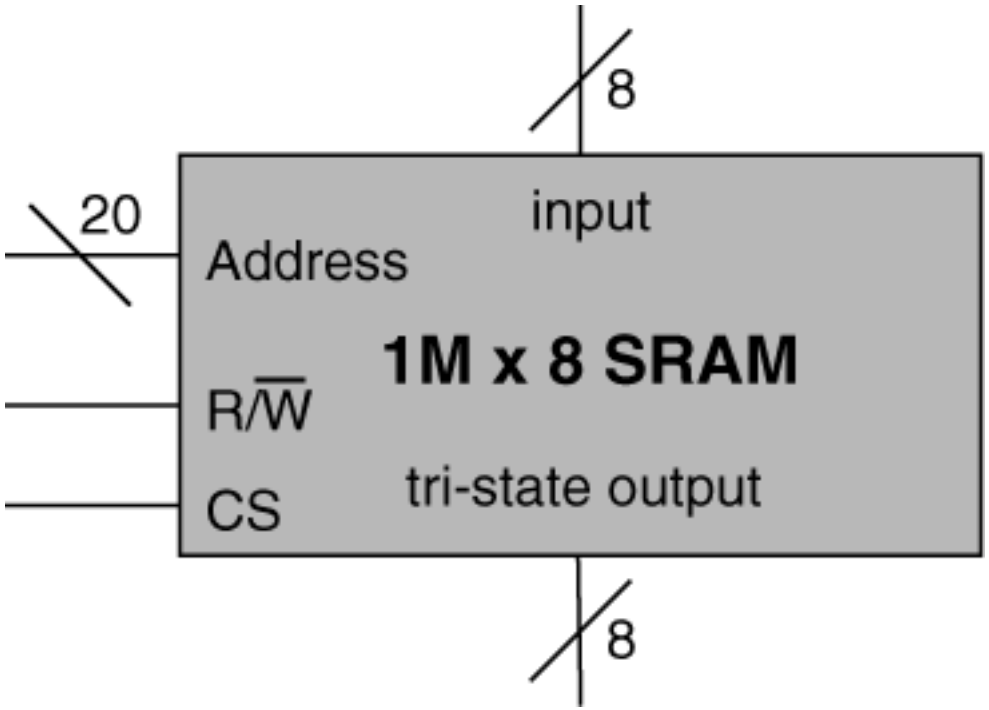
sync SRAM - write operations



sync SRAM - write operations

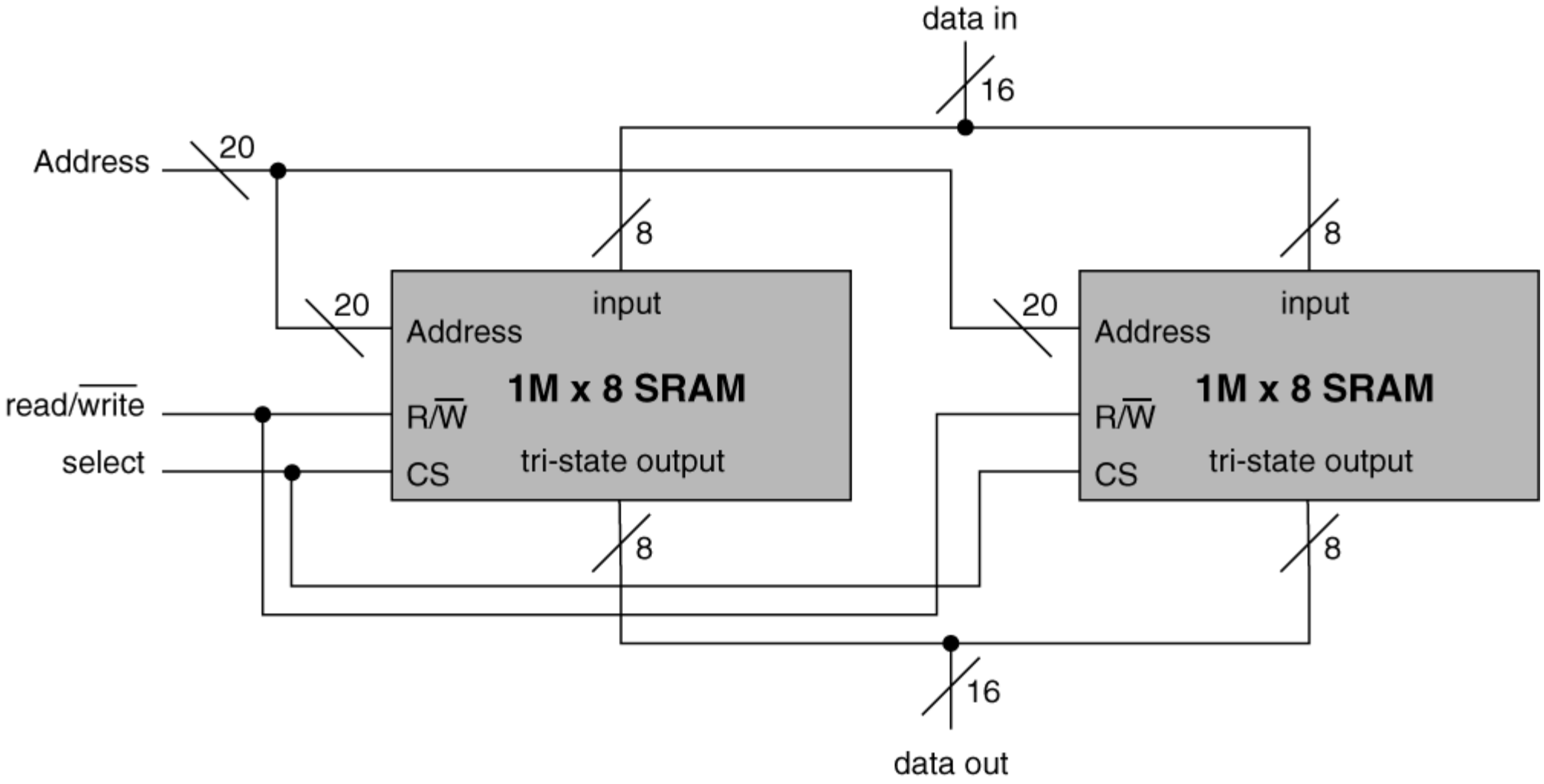
- ZBT SRAM (zero burst turnaround) shows an alternate method of solving the problem of the address needing to hang around for two clock cycles.
- Store it in another register
  - Downside, waste of floorplan area…..
  - Upside, can pipeline writes or interleave reads and writes.
- Wakerly - p865
- Is it worth it ?
  - IBM POWER architecture.

sync SRAM - 1M x 8 bits

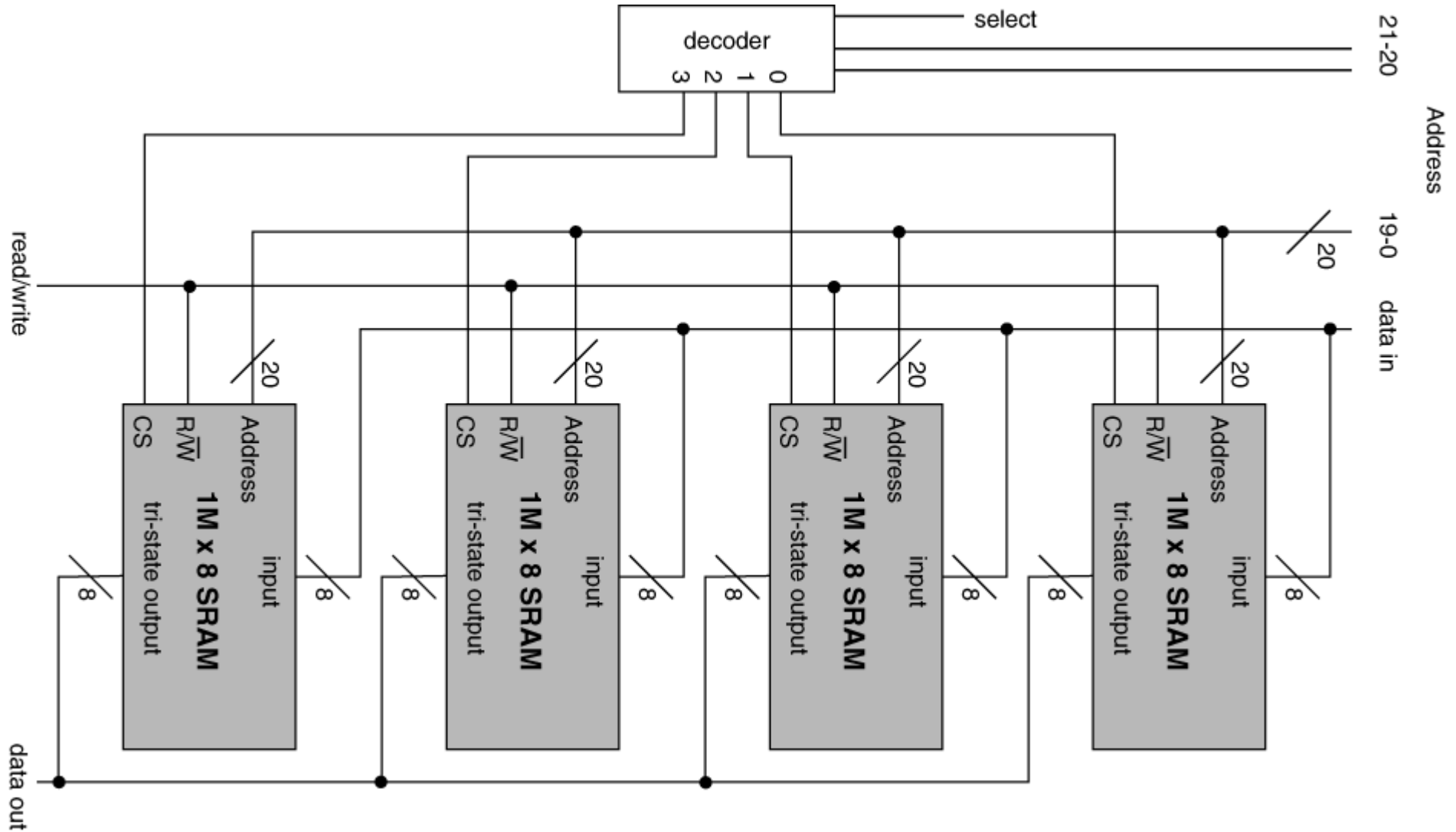


How do we link multiple memories together ?

sync SRAM - 1M x 16 bits

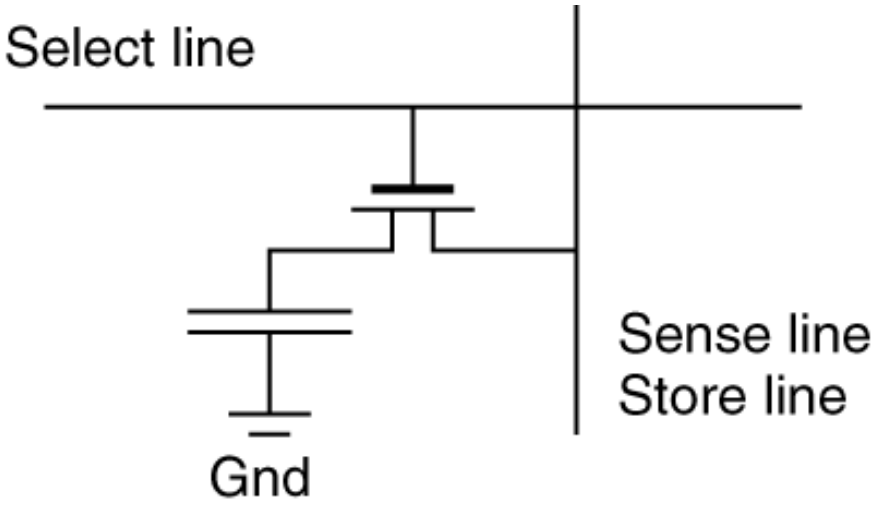


sync SRAM - 4M x 8 bits

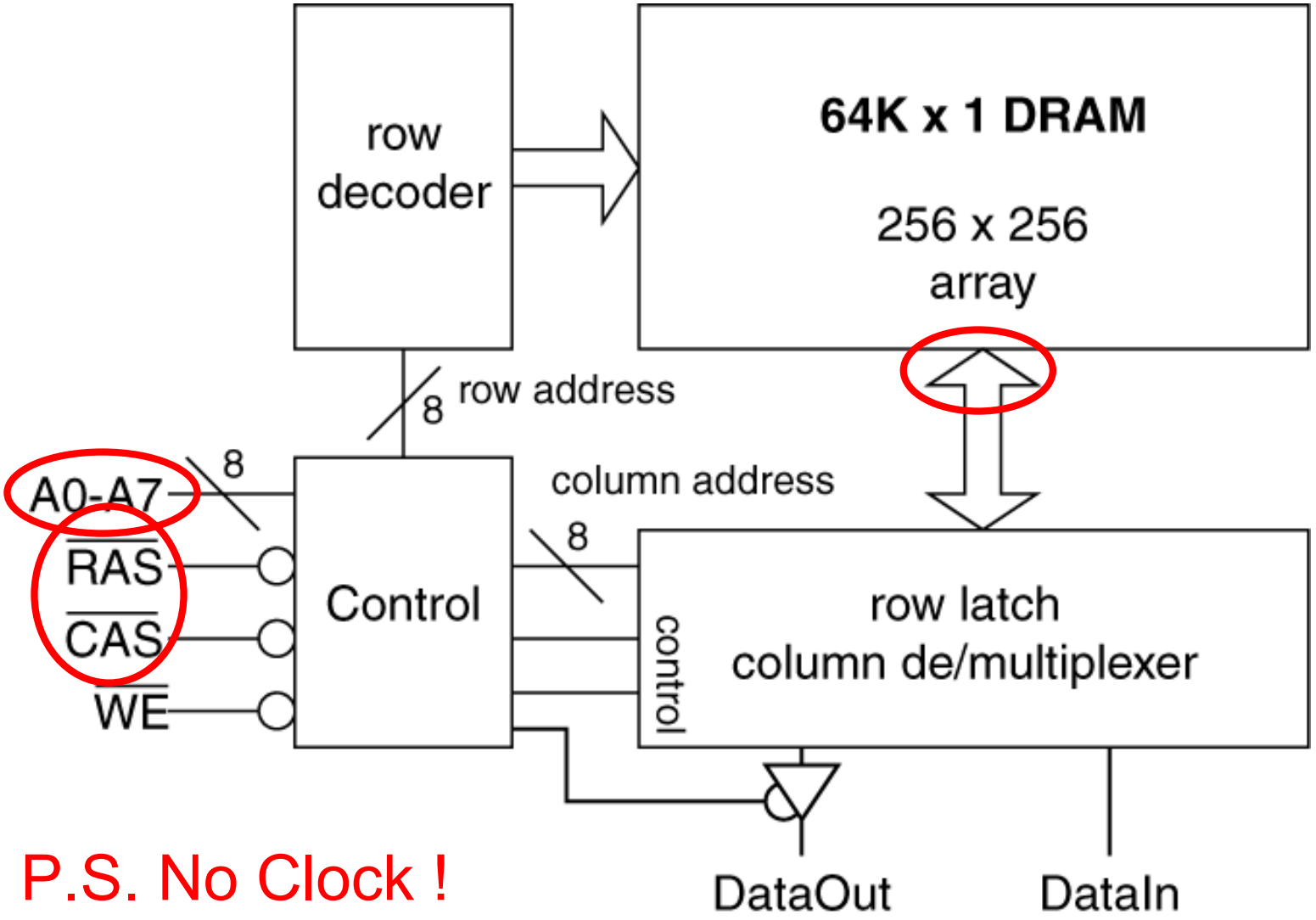


DRAM

- Memory element is entirely different in detail
- Destructive Read
- Leakage
- Need for ‘analogue’ row latch circuitry to amplify sense signal and feedback refresh.

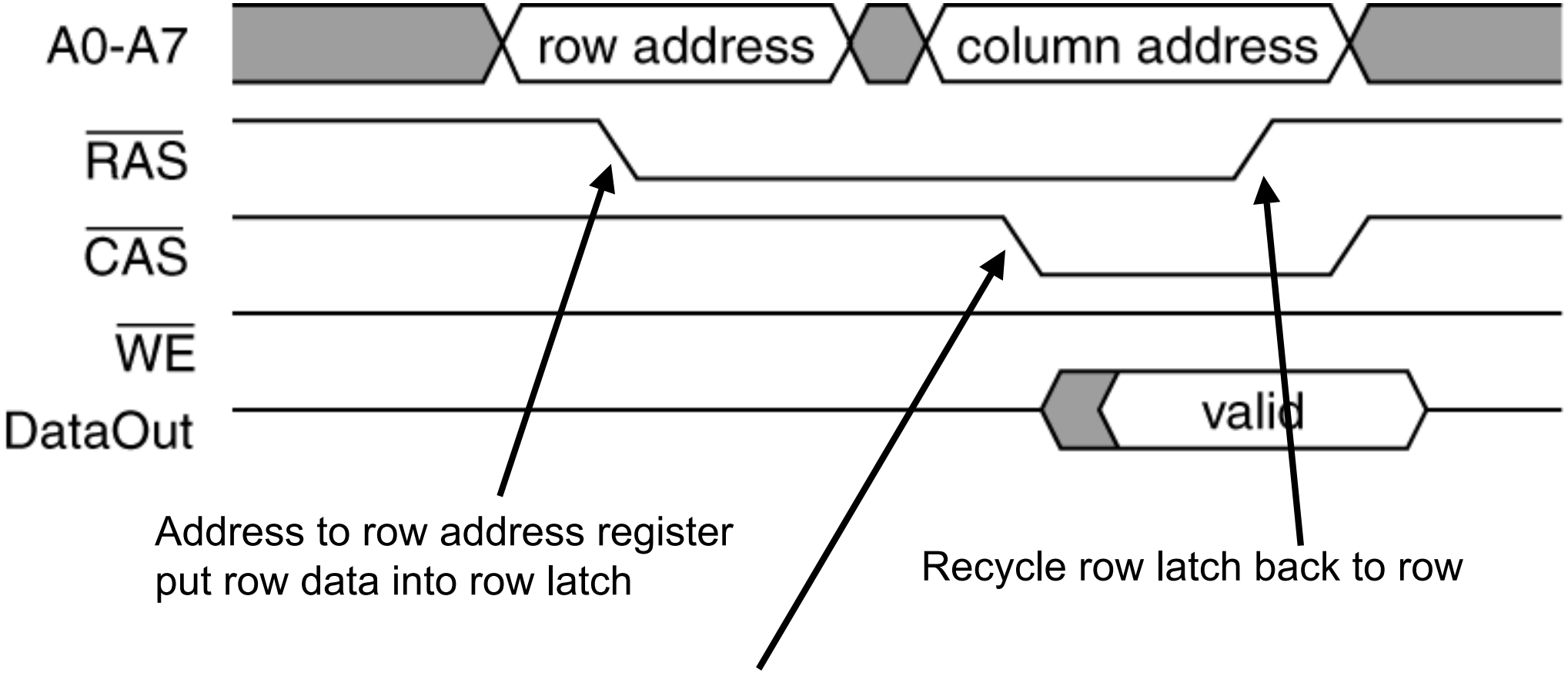


DRAM - overall structure

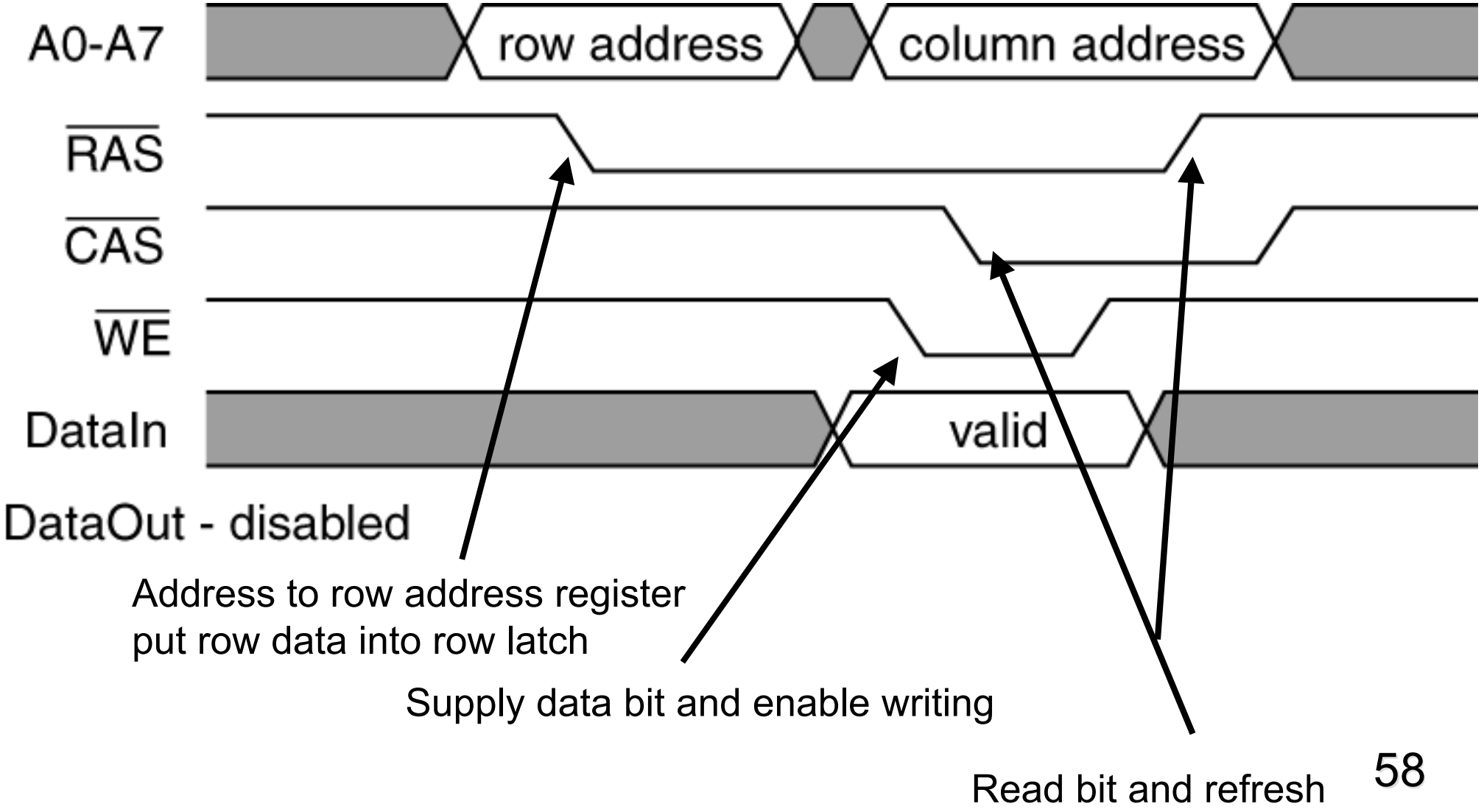


P.S. No Clock !

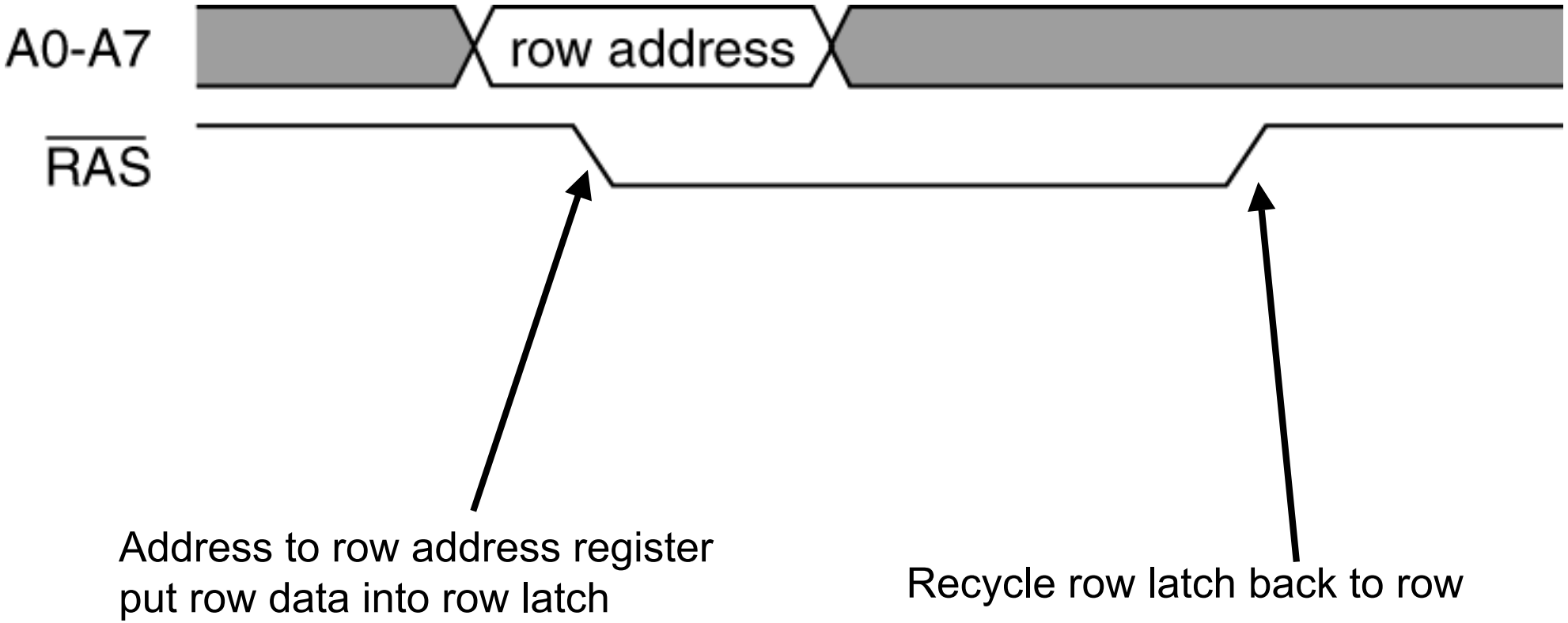
DRAM - read cycle



DRAM - write cycle



DRAM - refresh cycle



DRAM - refresh cycle

- 4Mx4bit DRAM requires each bit to be refreshed at least every 64 ms
- 4096 rows (1024 columns) so 4096 refreshes in 64 ms
- Available time therefore 15.625 μs
- Actual refresh time is 60 ns per row.
- Less than 1%, so negligible if distributed

DRAM - refresh cycle

- Refresh cycle shown needs an external DRAM controller to step through the rows and supply refresh signals at appropriate times
- More modern method is ‘CAS before RAS’ refresh (memory uses the otherwise unused CAS before RAS input to initiate a refresh cycle) which is associated with an internal refresh address counter. DRAM controller only chooses refresh times.

DRAM - other types

- EDO DRAM
  - Still used in older machines
  - Instead of output tri-state being controlled by CAS line becoming active on read cycle, an output enable signal is used.
  - CAS still controls output timing, OE controls output tri-state
  - Burst mode reads can happen much more quickly as no Z state between each one (extended data out)
- SDRAM
  - Discussed before, collar some DRAM with synchronisers as in SRAM above, and then play pipelining tricks internally.