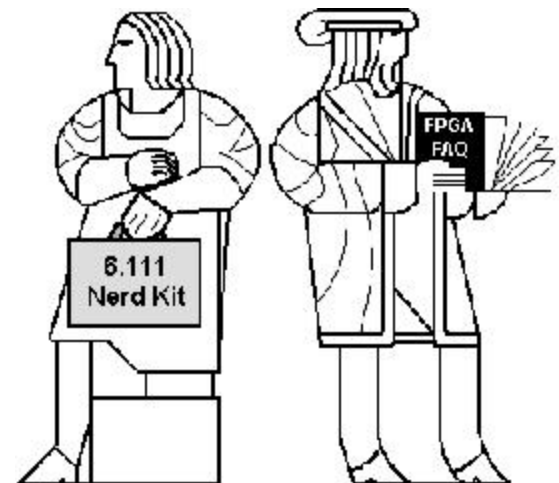


6.111 Lecture 10

Today: Memories

1. Static RAMs
2. Interfacing: Bus & Protocol
3. Synchronous Memories
4. EPROMs and DRAMs
5. Memory Mapped Peripherals



Acknowledgement: Nathan Ickes, Rex Min

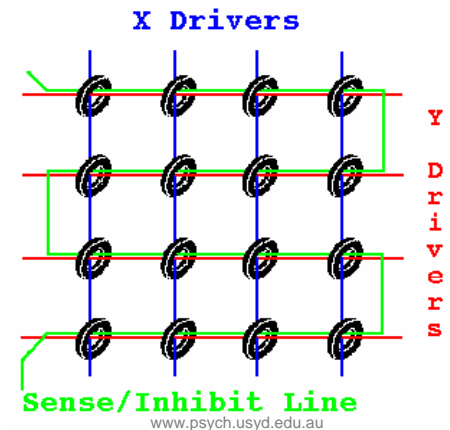
J. Rabaey, A. Chandrakasan, B. Nikolic, "Digital Integrated Circuits: A Design Perspective"
Prentice Hall, 2003 (Chapter 10)

Memories of a Digital World

Why need memories? State machines...

Memories:

- **Flip Flips, Registers, FIFO** (first-in-first-out)
- **Core memory!**
- **Random Access** (static/dynamic, read/write)
- **Slow / Non-volatile** (hard drive/eeprom/eprom)
- **Content-addressable**
- **Concept of a BUS and use of TRISTATE!**



Key Design Metrics:

1. **Memory Density** (number of bits/ μm^2) and **Size**
2. **Access Time** (time to read or write) and **Throughput**
3. **Power Dissipation**

Memory Classification & Metrics

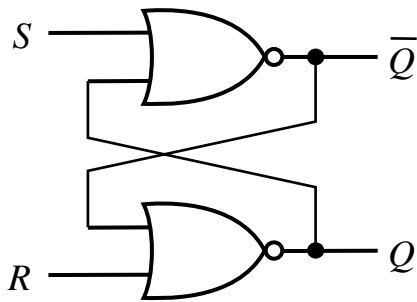
Read-Write Memory		Non-Volatile Read-Write Memory	Read-Only Memory (ROM)
Random Access	Non-Random Access	EPROM E ² PROM	Mask-Programmed
SRAM DRAM	FIFO LIFO	FLASH	

Key Design Metrics:

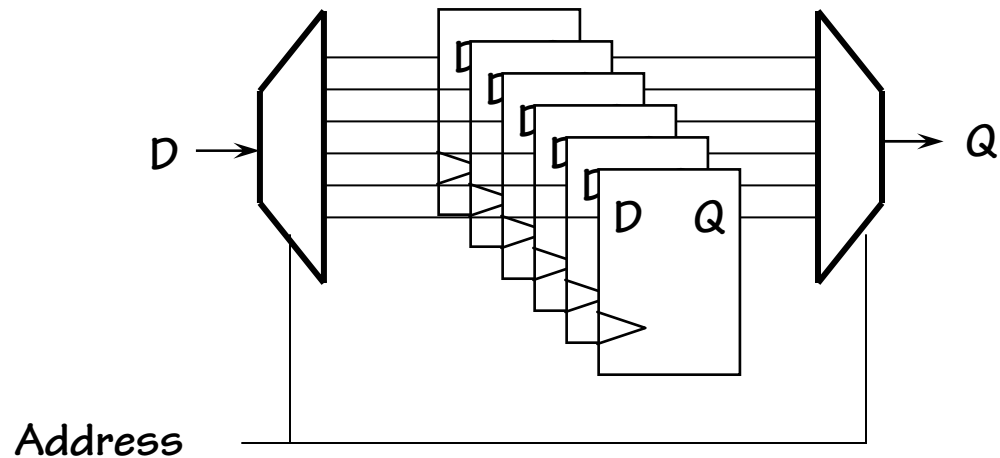
1. Memory Density (number of bits/ μm^2) and Size
2. Access Time (time to read or write) and Throughput
3. Power Dissipation

1. Static RAMs: Latch Based Memory

Set Reset Flip Flop



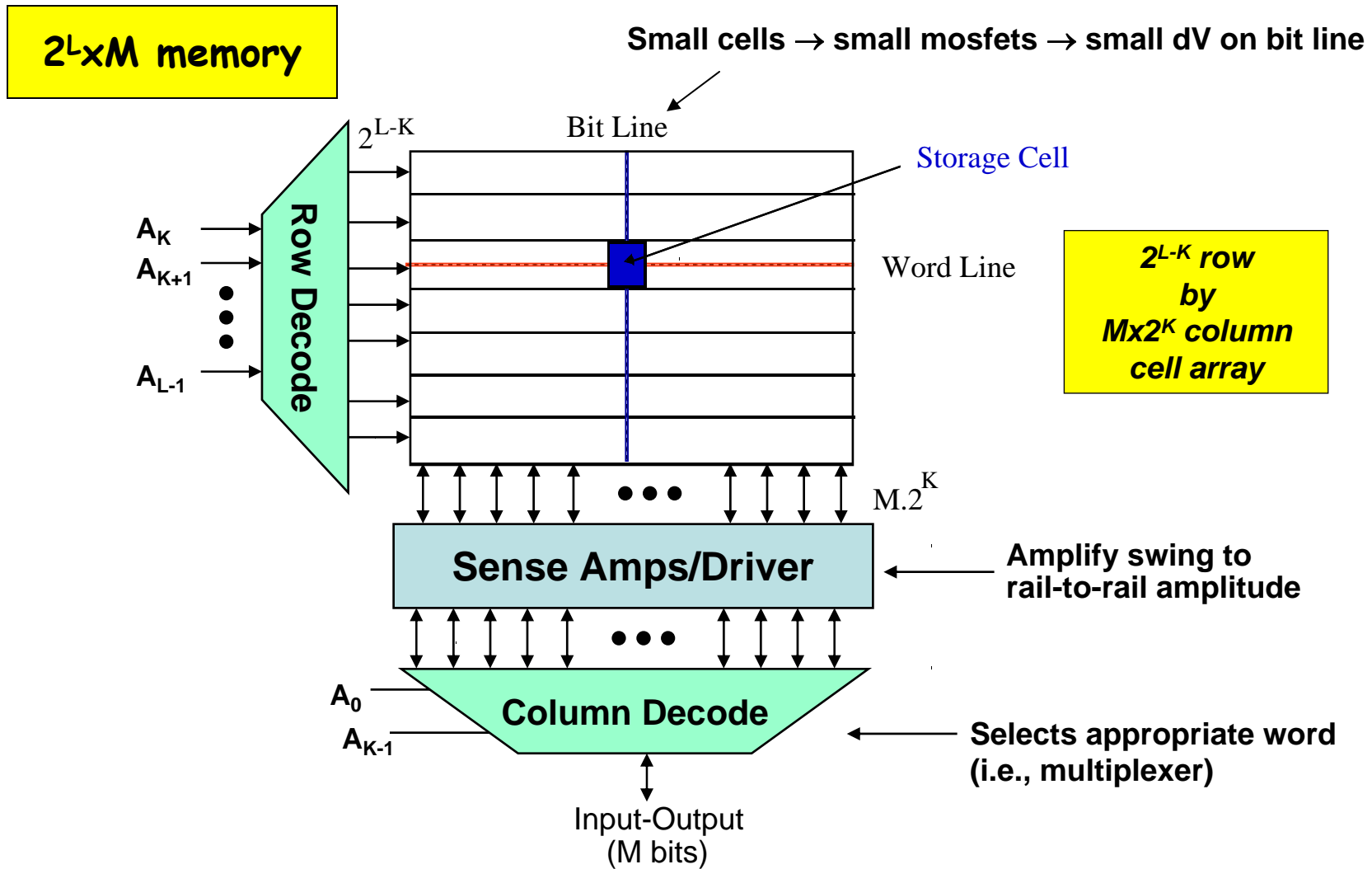
Register Memory



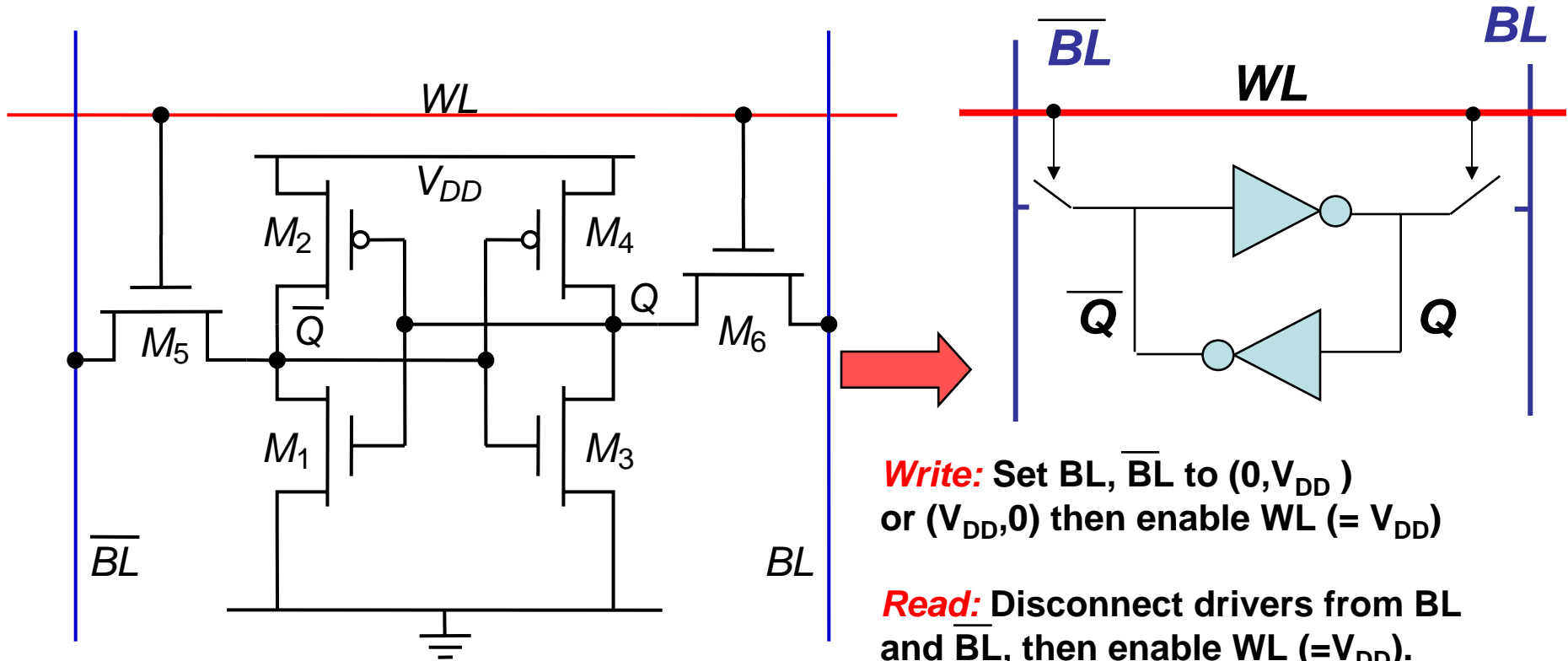
- Works fine for small memory blocks (e.g., small register files)
- Inefficient in area for large memories
- Density is the key metric in large memory circuits

How do we minimize cell size?

Memory Array Architecture



Static RAM (SRAM) Cell (The 6-T Cell)

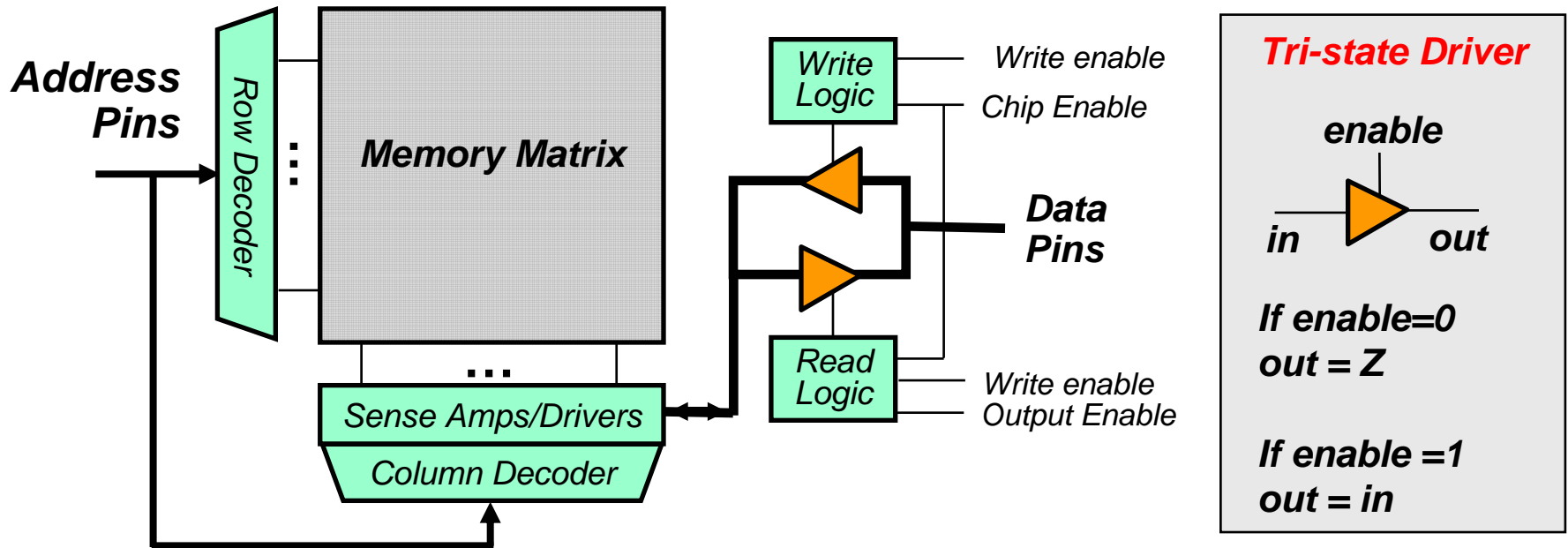


Write: Set BL, \overline{BL} to $(0, V_{DD})$ or $(V_{DD}, 0)$ then enable $WL (= V_{DD})$

Read: Disconnect drivers from BL and \overline{BL} , then enable $WL (= V_{DD})$. Sense a small change in BL or \overline{BL}

- **State held by cross-coupled inverters (M1-M4)**
- Retains state as long as power supply turned on
- Feedback must be overdriven to write into the memory

2. Using External Memory Devices



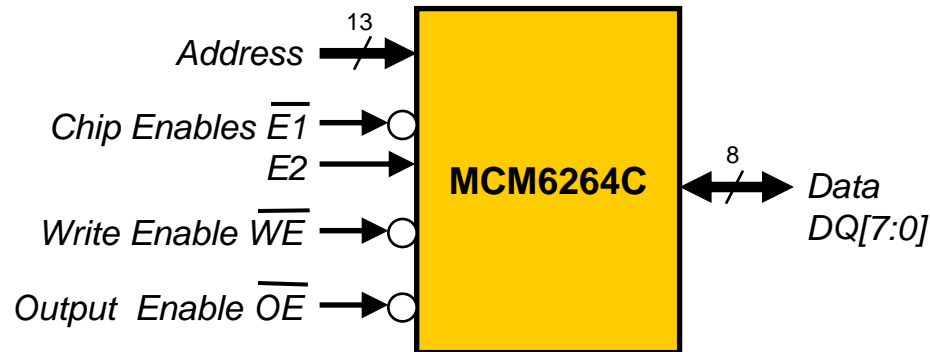
- **Address** pins drive row and column decoders
- **Data** pins are **bidirectional**: shared by reads and writes

Concept of "Data Bus"

- **Output Enable** gates the chip's tristate driver
- **Write Enable** sets the memory's read/write mode
- **Chip Enable/Chip Select** acts as a "master switch"

MCM6264C 8K x 8 Static RAM

On the outside:



Same (bidirectional) data bus used for reading and writing

Chip Enables ($\overline{E1}$ and $E2$)

$\overline{E1}$ must be low and $E2$ must be high to enable the chip

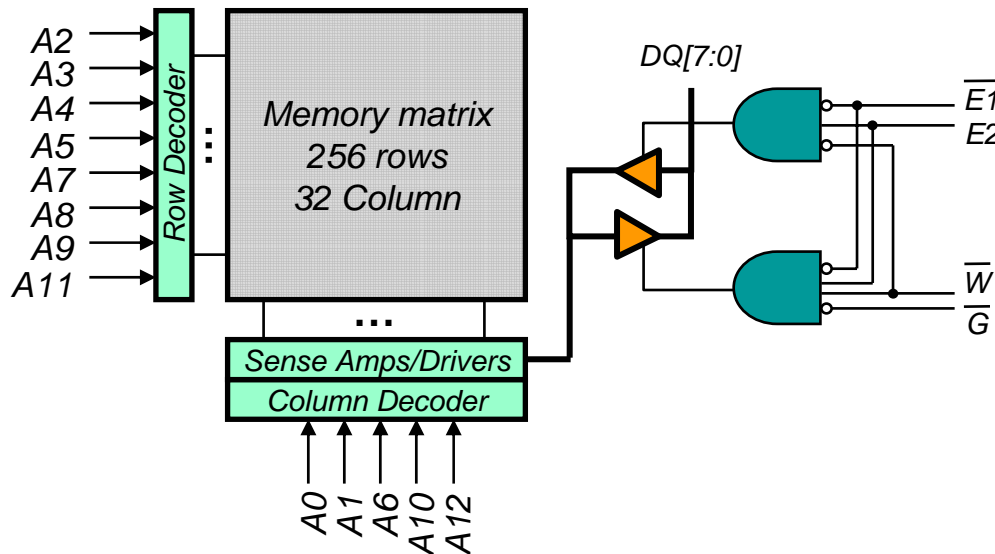
Write Enable (\overline{WE})

When low (and chip enabled), values on data bus are written to location selected by address bus

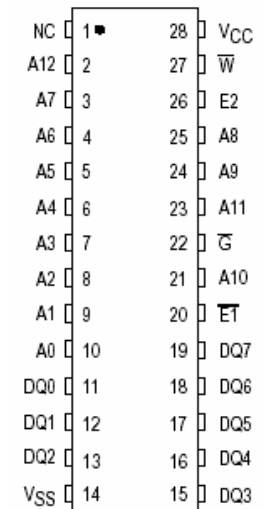
Output Enable (\overline{OE} or \underline{G})

When low (and chip is enabled), data bus is driven with value of selected memory location

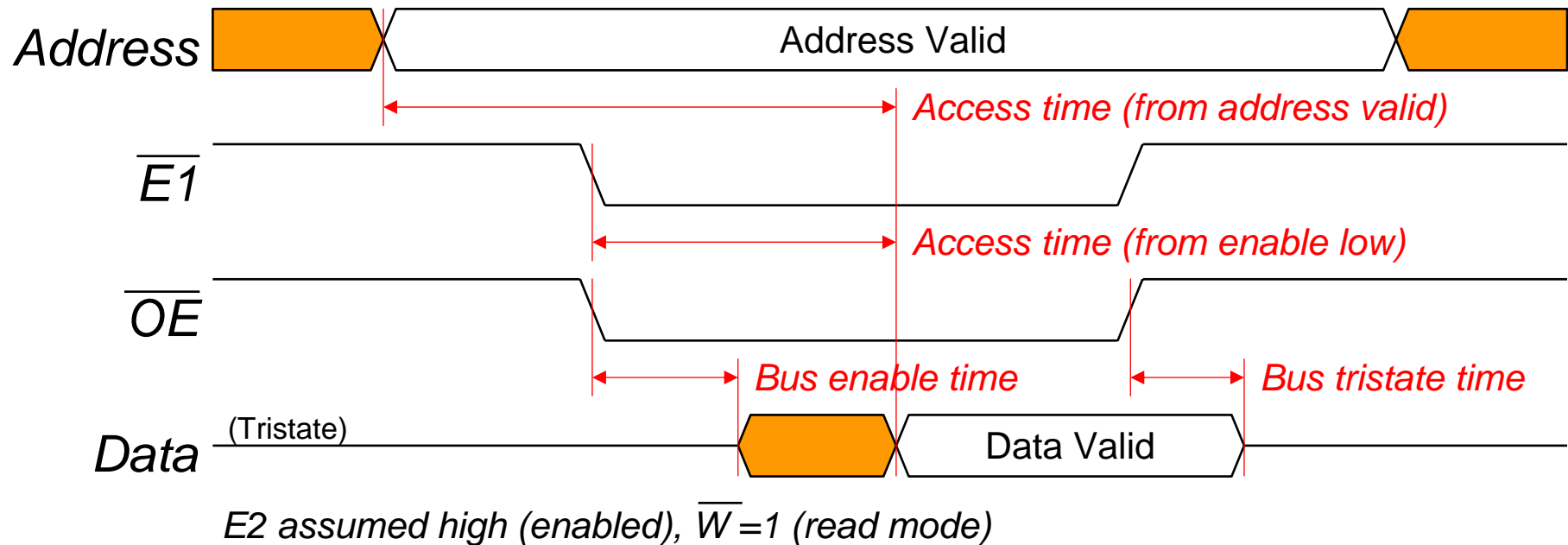
On the inside:



Pinout

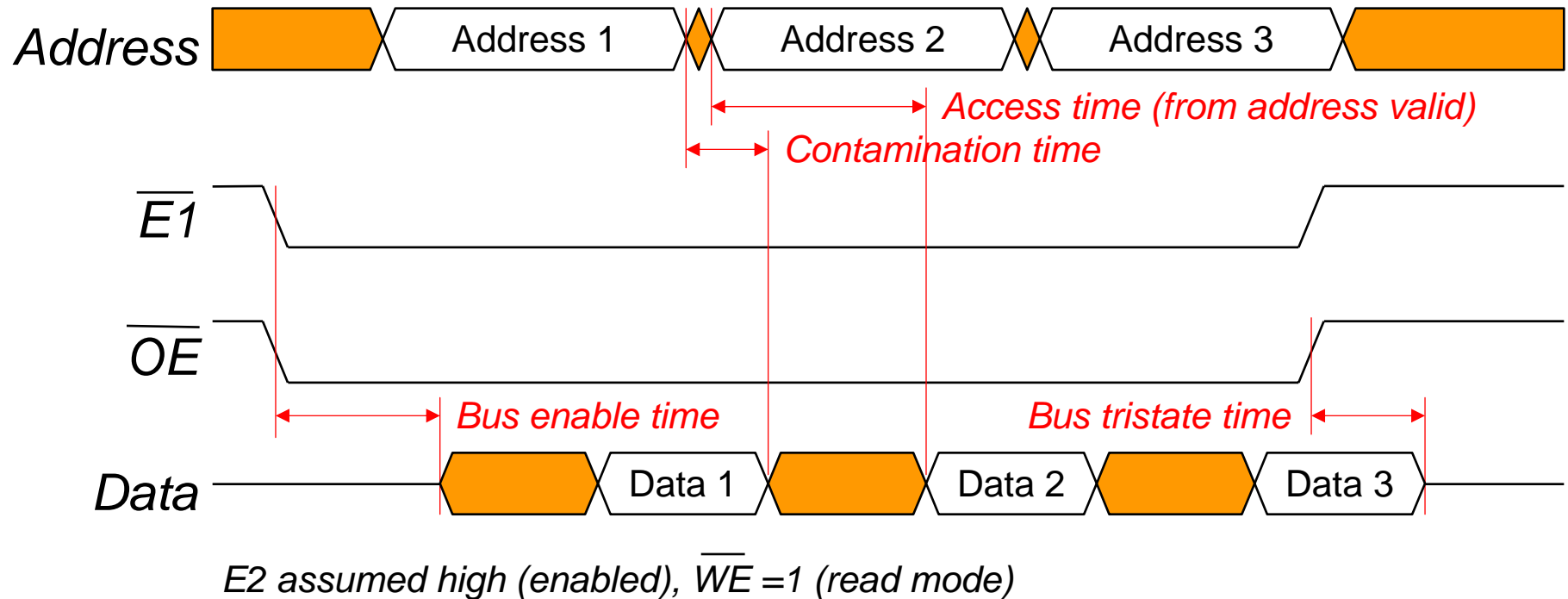


Reading an Asynchronous SRAM



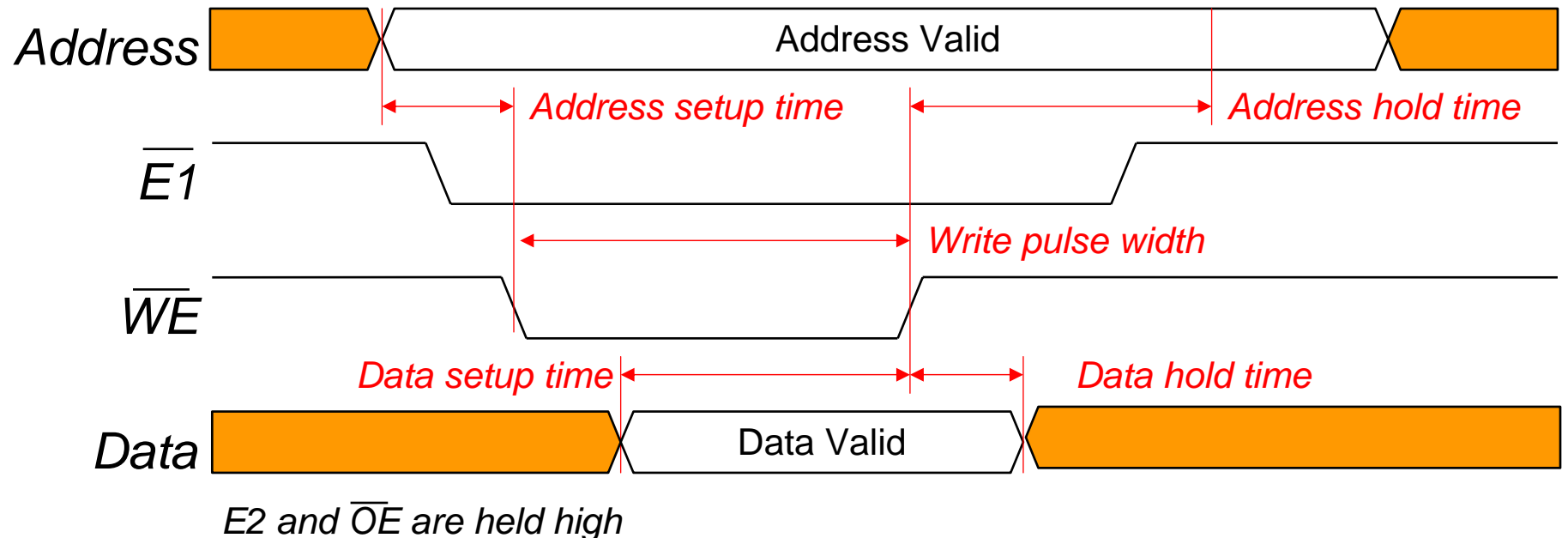
- Read cycle begins when all enable signals ($\overline{E1}$, $\overline{E2}$, \overline{OE}) are active
- Data is valid after read access time
 - Access time is indicated by full part number: *MCM6264CP-12* → 12ns
- Data bus is tristated shortly after \overline{OE} or $\overline{E1}$ goes high

Address Controlled Reads



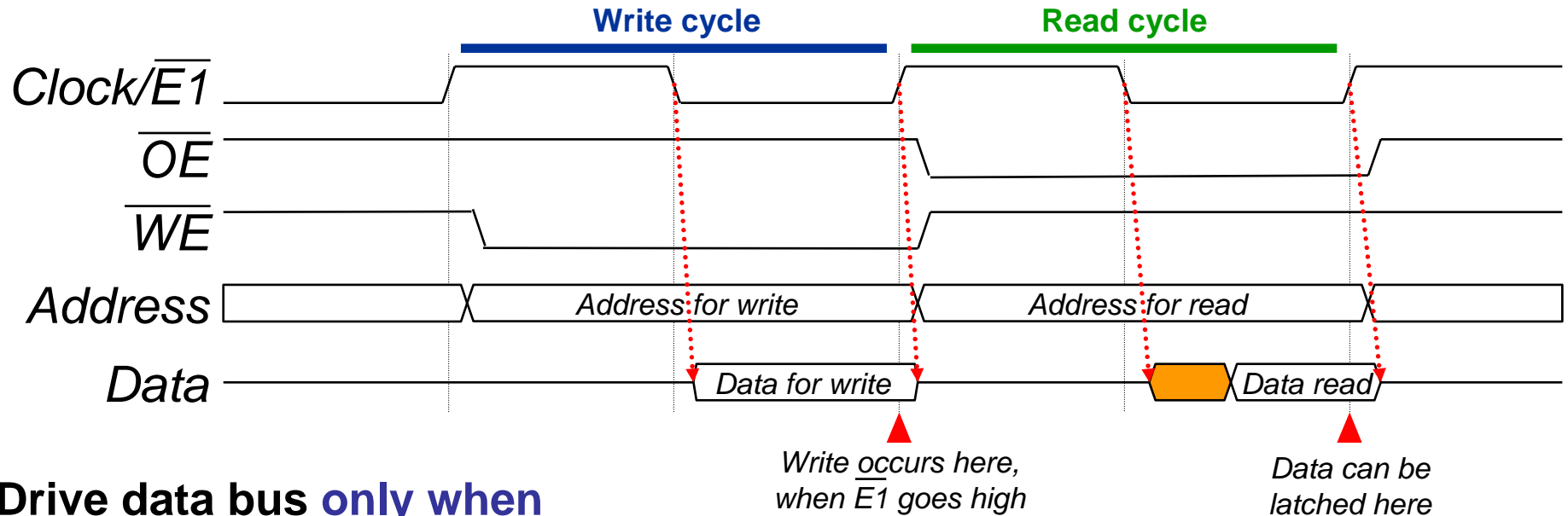
- Can perform multiple reads without disabling chip
- Data bus follows address bus, after some delay

Writing to Asynchronous SRAM



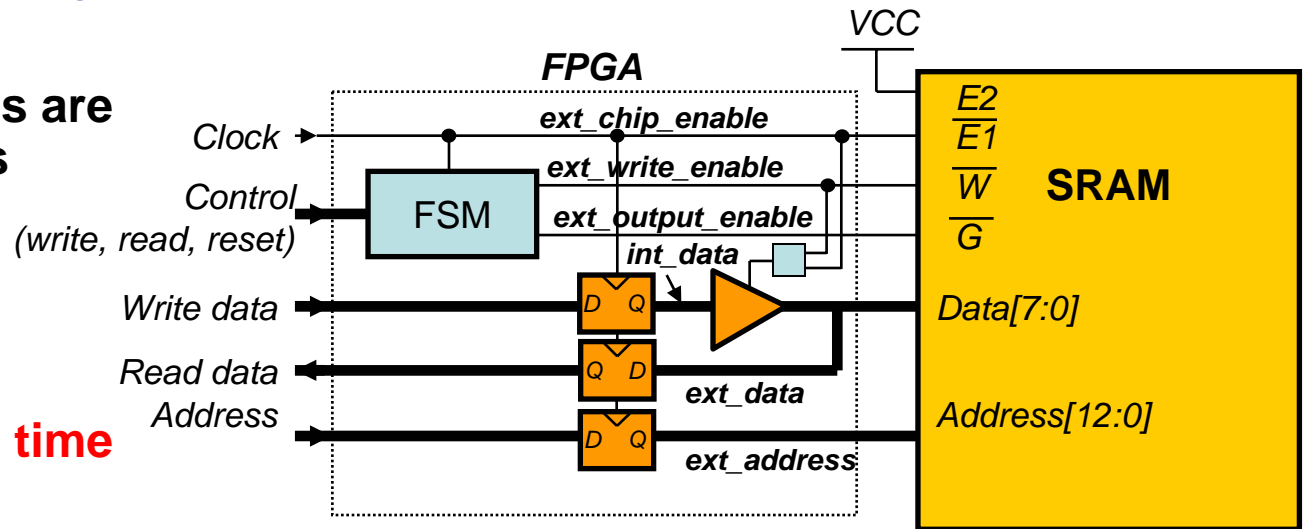
- **Data latched when \overline{WE} or $\overline{E1}$ goes high (or E2 goes low)**
 - Data must be stable at this time
 - Address must be stable before \overline{WE} goes low
- **Write waveforms are more important than read waveforms**
 - **Glitches to address can cause writes to random addresses!**

Sample Memory Interface Logic

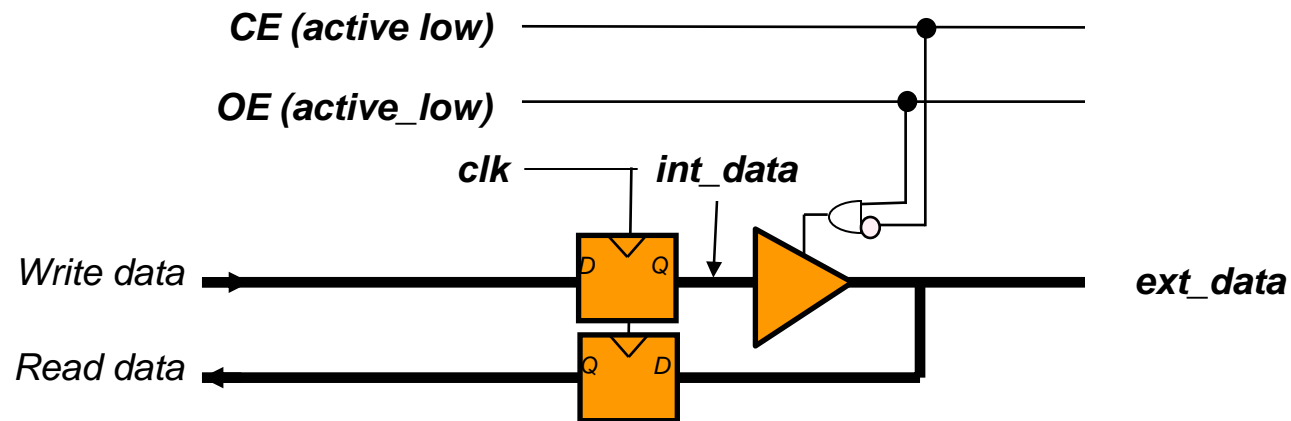


Drive data bus only when clock is low

- Ensures address are stable for writes
- Prevents bus contention
- **Minimum clock period is twice memory access time**



Tristate Data Buses in Verilog



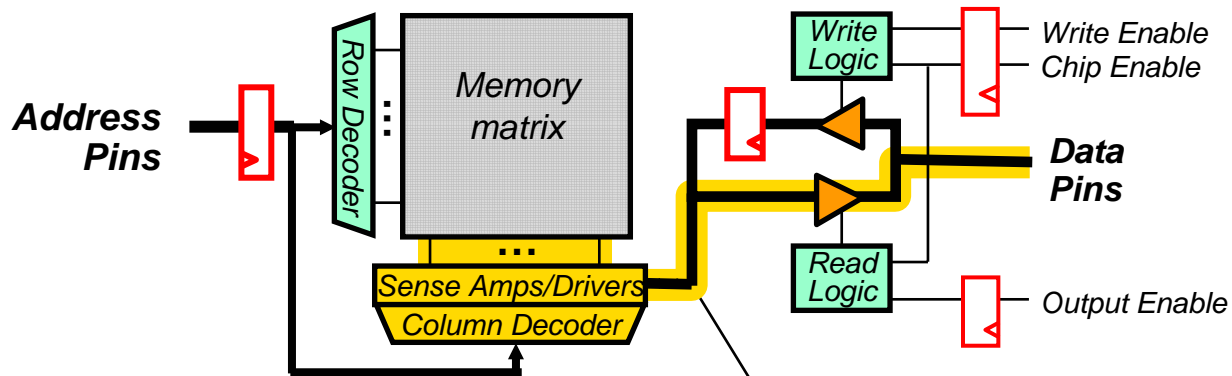
```
output CE,OE; // these signals are active low
inout [7:0] ext_data;
reg [7:0] read_data,int_data
wire [7:0] write_data;

always @ (posedge clk) begin
    int_data <= write_data;
    read_data <= ext_data;
end

// Use a tristate driver to set ext_data to a value
assign ext_data = (~CE & OE) ? int_data : 8'hZZ;
```

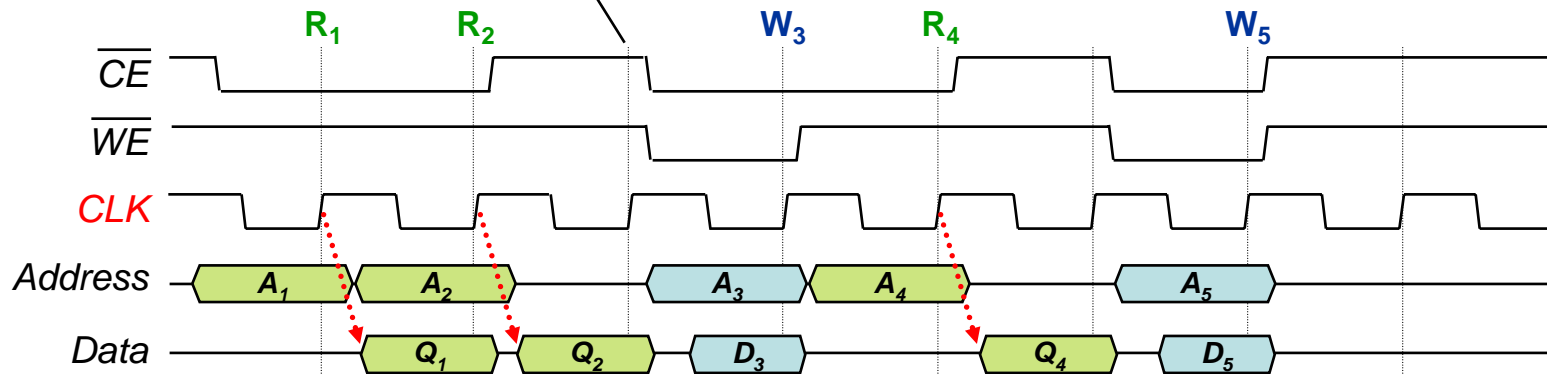
3. Synchronous SRAM Memories

- **Clocking** provides input synchronization and encourages more reliable operation at high speeds



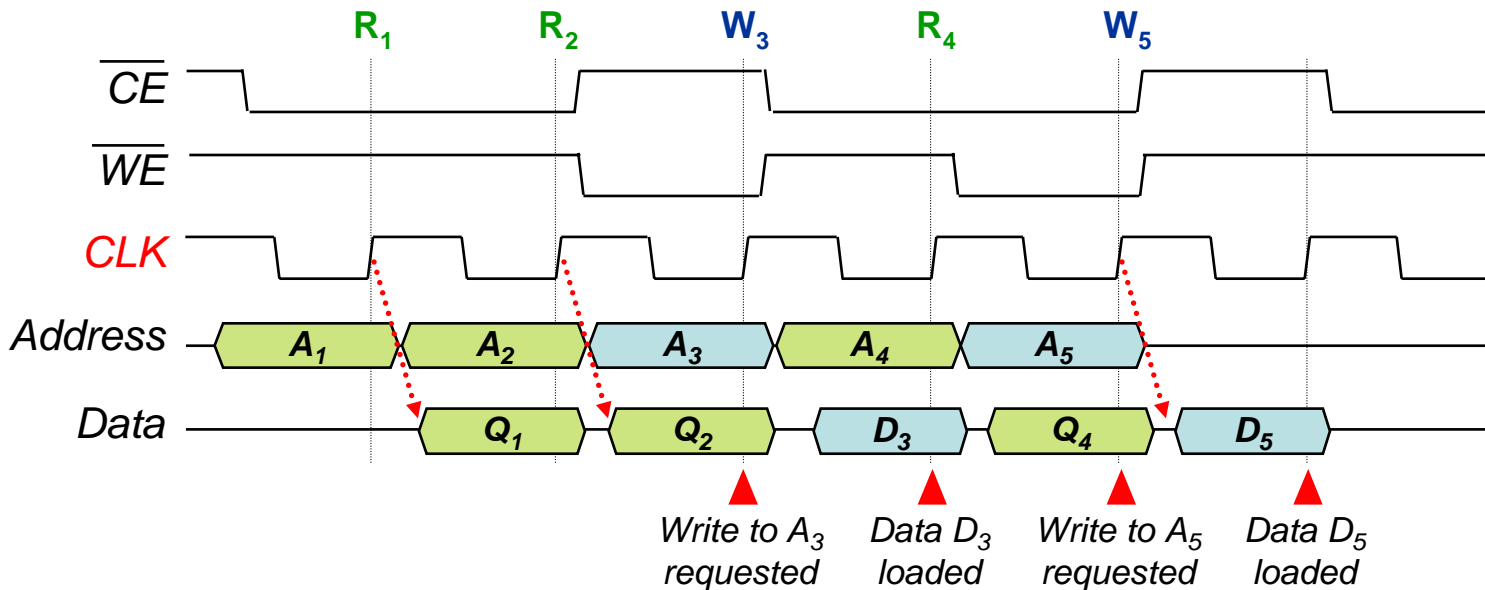
difference between read and write timings creates wasted cycles ("wait states")

long "flow-through" combinational path creates high CLK-Q delay



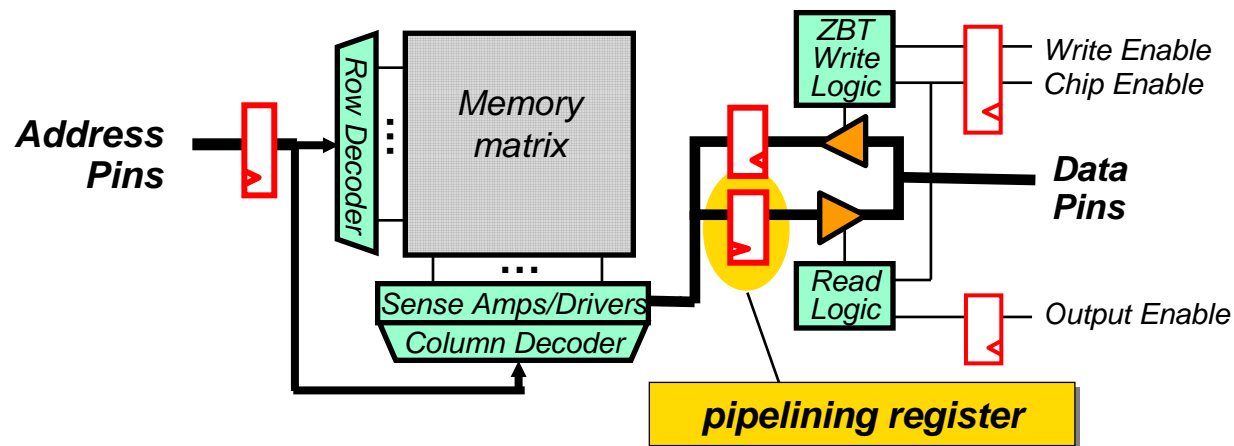
ZBT Eliminates the Wait State

- The wait state occurs because:
 - On a read, data is available *after* the clock edge
 - On a write, data is set up *before* the clock edge
- ZBT (“zero bus turnaround”) memories **change the rules for writes**
 - On a write, data is set up **after** the clock edge (so that it is read on the following edge)
 - Result: no wait states, higher memory throughput

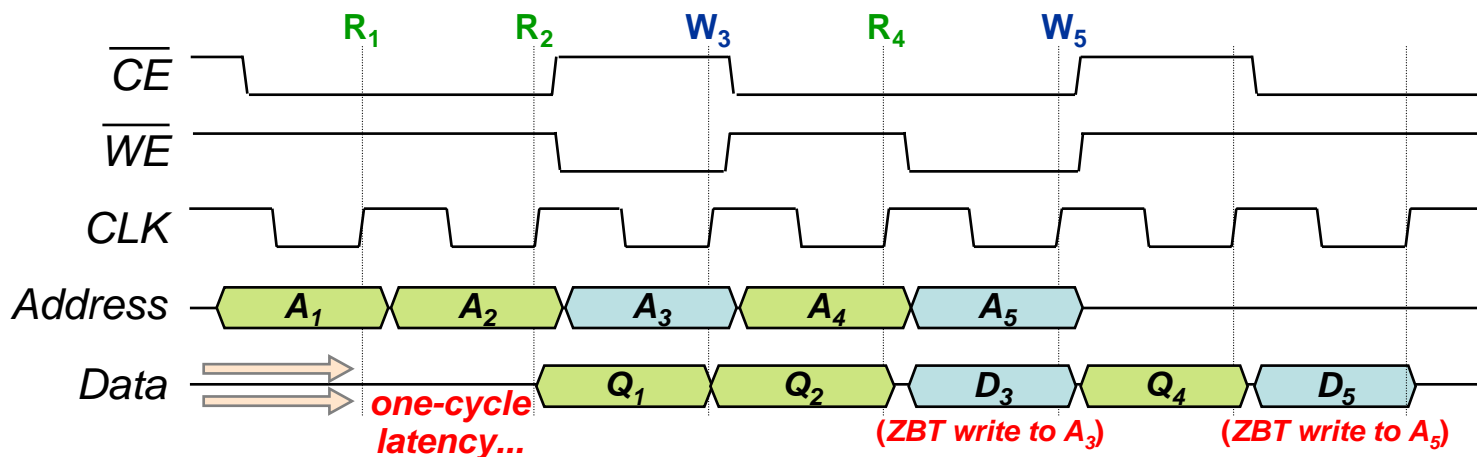


Pipelining Allows Faster CLK

- Pipeline the memory by registering its output
 - Good: Greatly reduces CLK-Q delay, allows higher clock (more throughput)
 - Bad: Introduces an extra cycle before data is available (more latency)



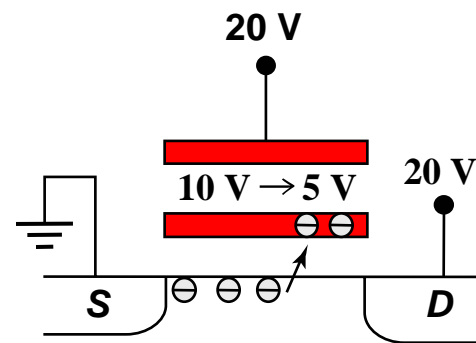
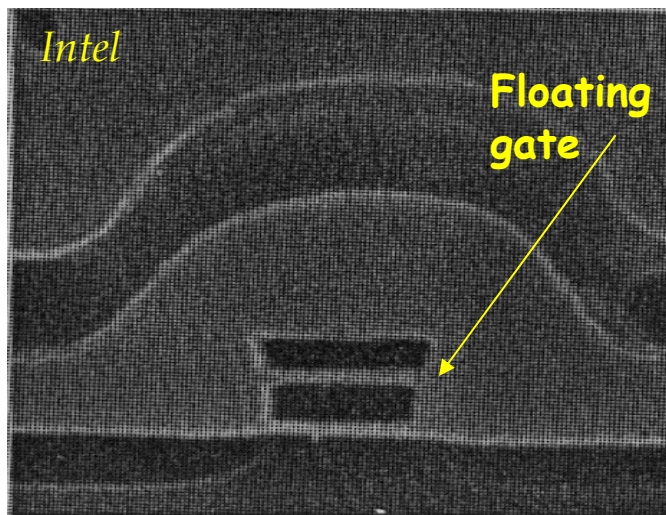
As an example,
see the **CY7C147X**
ZBT Synchronous
SRAM



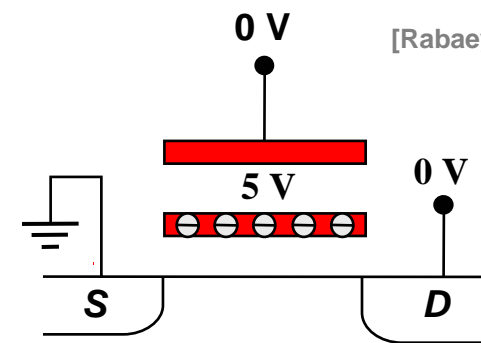
4. EPROMs and DRAMs

EEPROM - The Floating Gate Transistor

Electrically Erasable Programmable Read-Only Memory



Avalanche injection



Removing programming voltage leaves charge trapped

[Rabaey03]

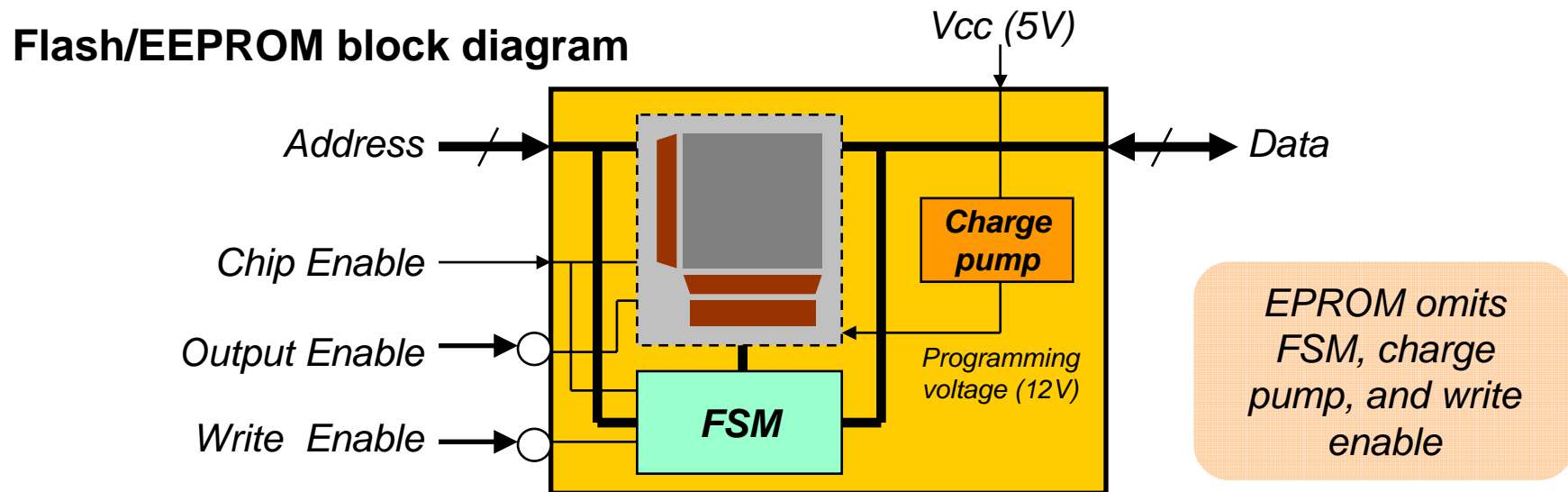
This is a non-volatile memory (retains state when supply turned off)

Usage: Just like SRAM, but **writes are much slower** than reads
(write sequence is controlled by an FSM internal to chip)

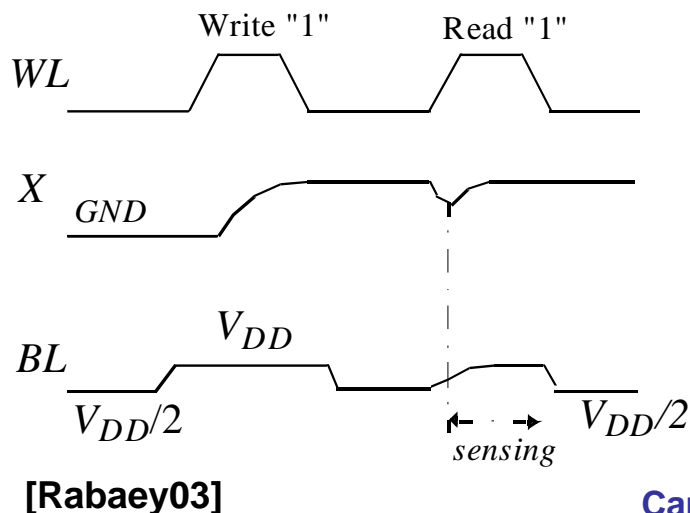
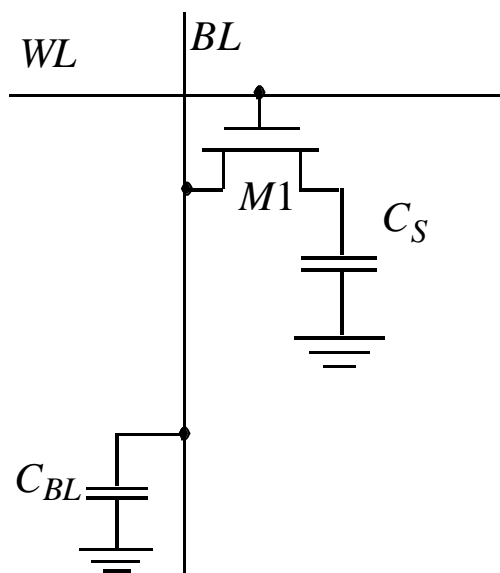
Common application: configuration data (serial EEPROM)

Interacting with Flash and (E)EPROM

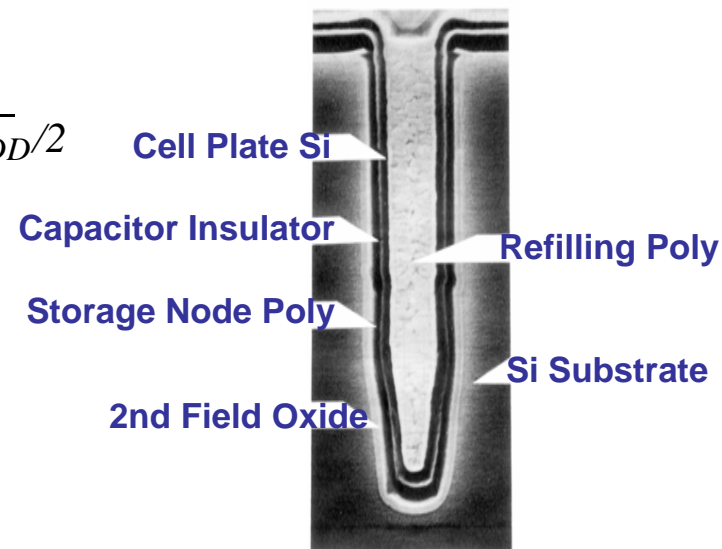
- Reading from flash or (E)EPROM is the same as reading from SRAM
- V_{pp} : input for programming voltage (12V)
 - EPROM: V_{pp} is supplied by programming machine
 - Modern flash/EEPROM devices generate 12V using an on-chip charge pump
- EPROM lacks a write enable
 - Not in-system programmable (must use a special programming machine)
- For flash and EEPROM, write sequence is controlled by an internal FSM
 - Writes to device are used to send signals to the FSM
 - Although the same signals are used, one can't write to flash/EEPROM in the same manner as SRAM



Dynamic RAM (DRAM) Cell



DRAM uses Special Capacitor Structures

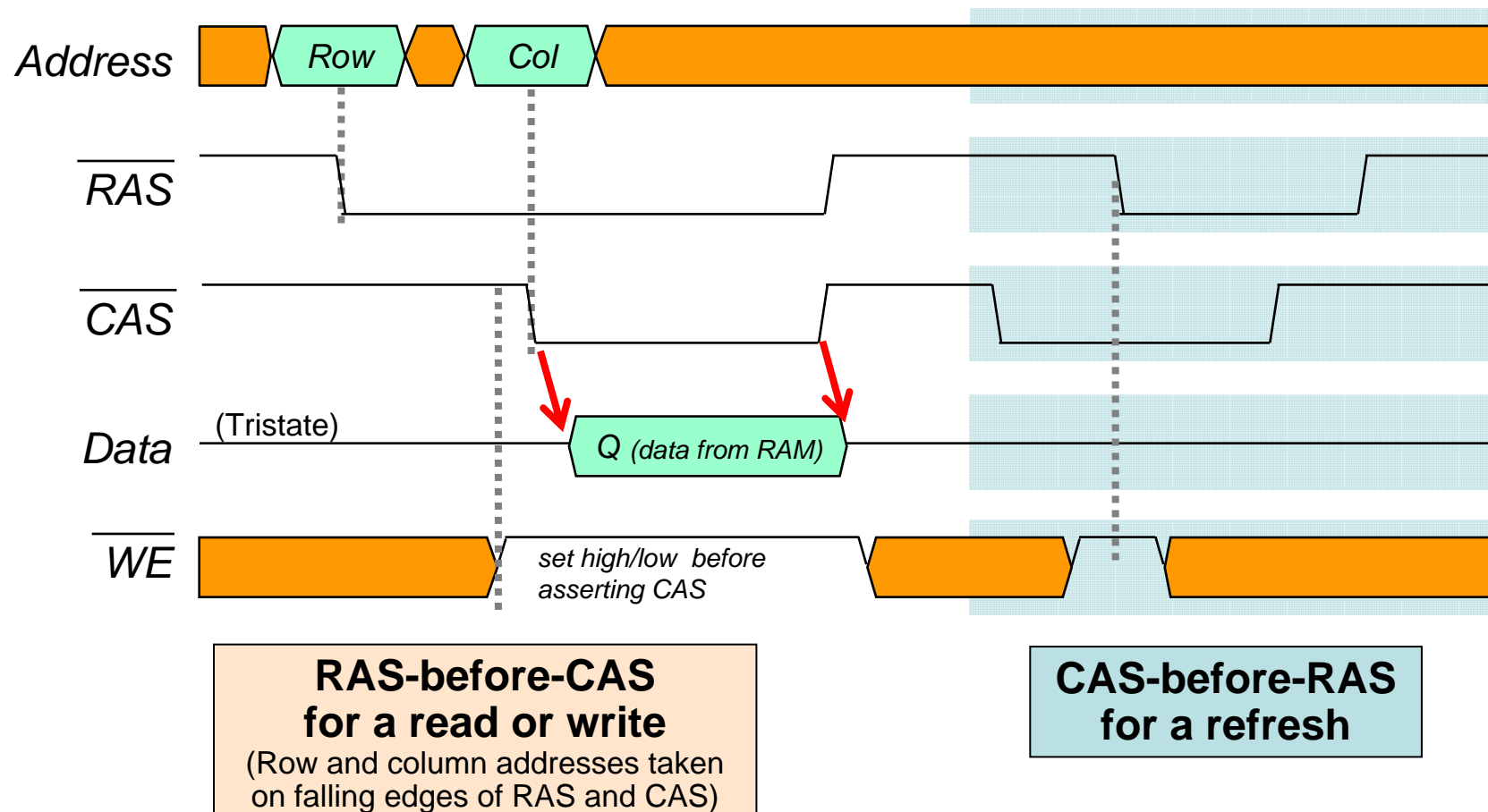


To Write: set Bit Line (BL) to 0 or V_{DD} & enable Word Line (WL) (i.e., set to V_{DD})

To Read: set Bit Line (BL) to $V_{DD}/2$ & enable Word Line (i.e., set it to V_{DD})

- **DRAM relies on charge stored in a capacitor to hold state**
- Found in all high density memories (one bit/transistor)
- Must be “refreshed” or state will be lost – high overhead

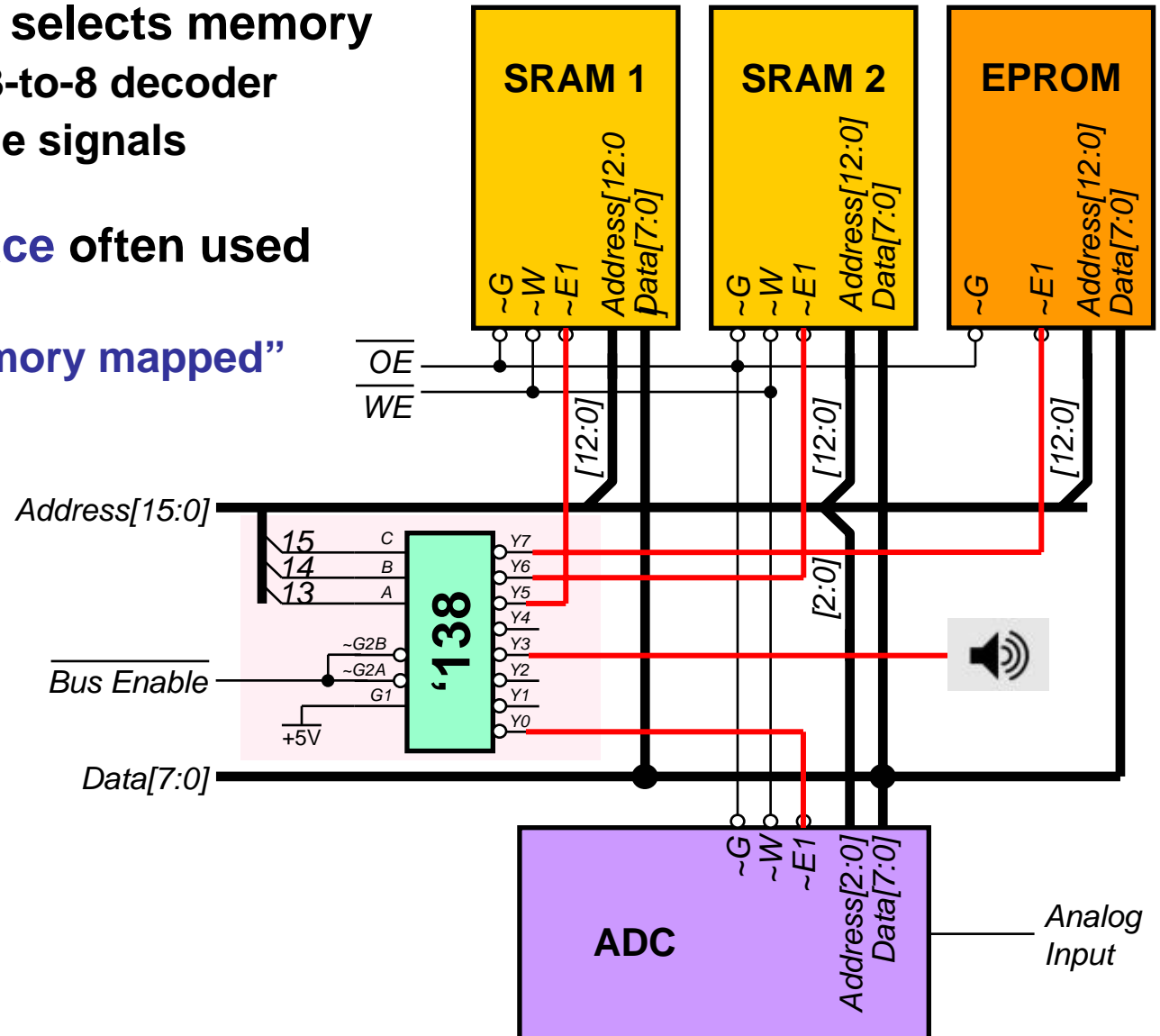
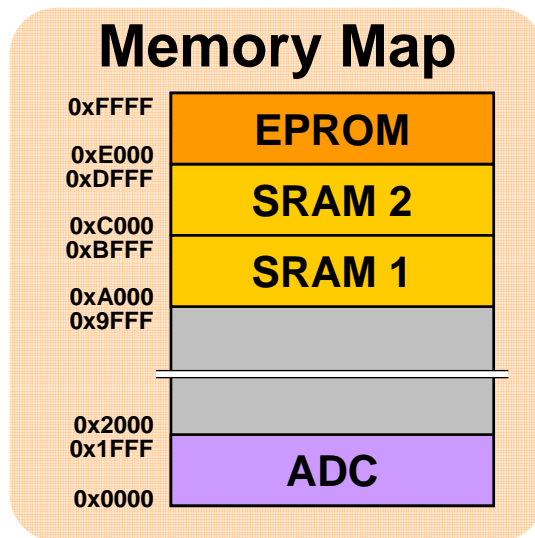
Asynchronous DRAM Operation



- **Clever manipulation of RAS and CAS after reads/writes provide more efficient modes: early-write, read-write, hidden-refresh, etc. (See datasheets for details)**

5. Addressing with Memory Maps

- Address decoder selects memory
 - Example: '138 3-to-8 decoder
 - Produces enable signals
- SRAM-like interface often used for peripherals
 - Known as “memory mapped” peripherals



Memory Devices: Helpful Knowledge

- **SRAM vs. DRAM**
 - SRAM holds state as long as power supply is turned on. DRAM must be "refreshed" - results in more complicated control
 - DRAM has much higher density, but requires special capacitor technology.
 - FPGA usually implemented in a standard digital process technology and uses SRAM technology
- **Non-Volatile Memory**
 - Fast Read, but very slow write (EPROM must be removed from the system for programming!)
 - Holds state even if the power supply is turned off
- **Memory Internals**
 - Has quite a bit of analog circuits internally -- pay particular attention to noise and PCB board integration
- **Device details**
 - Don't worry about them, wait until 6.012 or 6.374

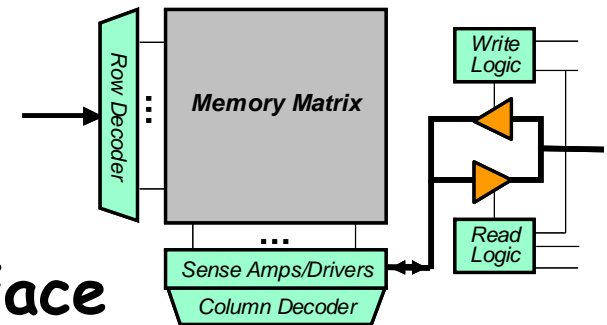
You Should Understand Why...

- control signals such as *Write Enable* should be registered
- a multi-cycle read/write is safer from a timing perspective than the single cycle read/write approach
- it is a bad idea to enable two tri-states driving the bus at the same time
- an SRAM does not need to be “refreshed” while a DRAM requires refresh
- an EPROM/EEPROM/FLASH cell can hold its state even if the power supply is turned off
- a synchronous memory can result in higher throughput

Summary

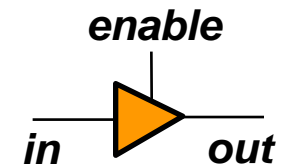
- **SRAMs:**

- Use synchronous FSM for interface
- Faster than DRAM, no refresh required



- **Data bus:**

- Tri-state (Z) used when not driving bus
- Multiple devices can share one bus



- **EEPROM:**

- Useful for config. data, long-term storage

- **Memory-Mapping:**

- Interact with peripheral as if it were an SRAM