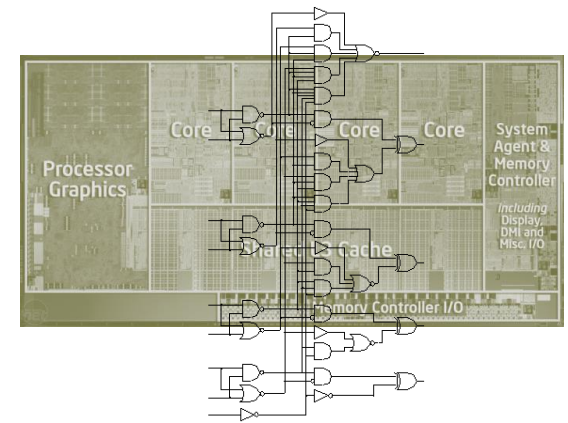




고려대학교
KOREA UNIVERSITY



COSE221, COMP211 Digital Logic Design

Lecture 10. Memory

Prof. Taeweon Suh

Computer Science & Engineering

Korea University

Memory

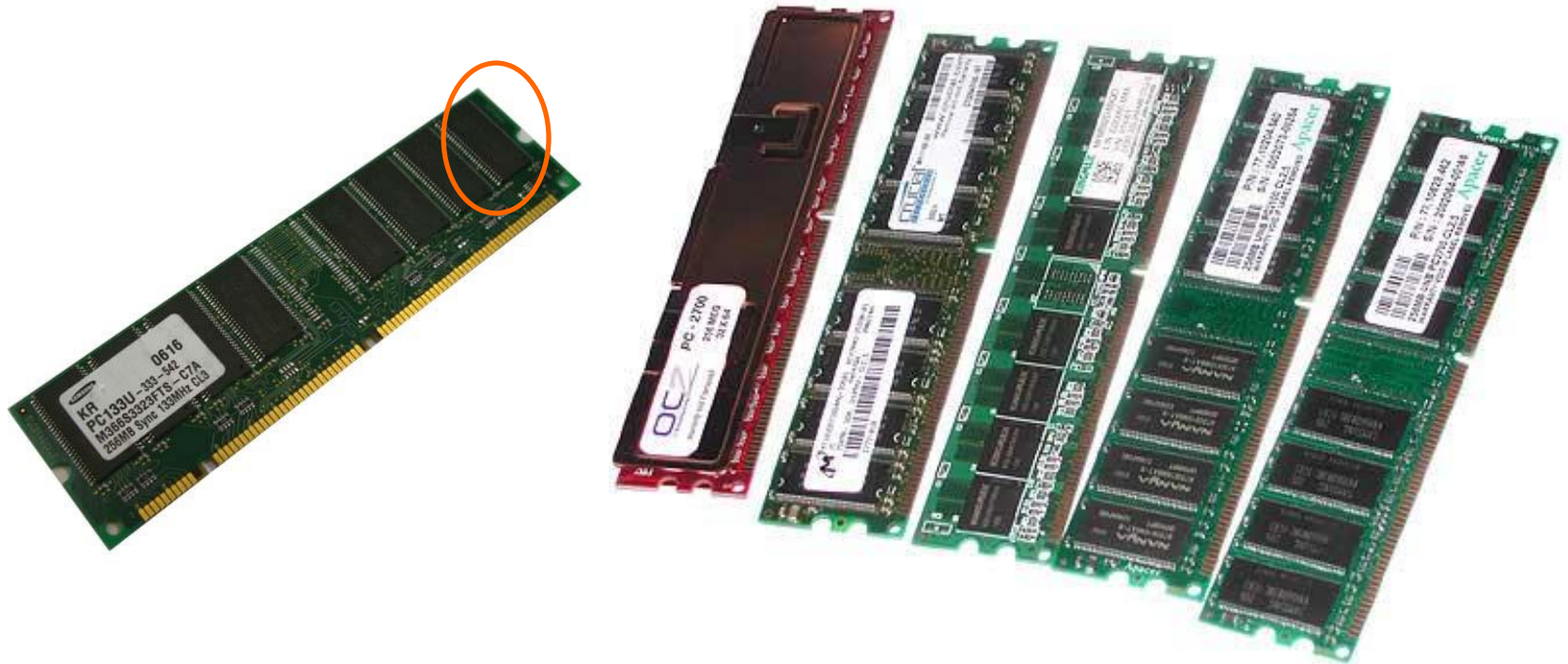
- Digital systems including computer systems require memories
 - Memory is used to store instructions and data
- Registers (made from flip-flops) are kind of memory
 - But, how many transistors need to build a 1-bit register (that is, to store 1-bit) with a flip-flop?
- Here we study memory arrays that can **efficiently** store large amounts of data
 - RAM and ROM

Memory

- Random Access Memory (RAM)
 - Static Random Access Memory (SRAM)
 - Data stored so long as power is applied
 - 6-transistors per cell
 - Faster
 - Differential
 - Dynamic Random Access Memory (DRAM)
 - Require periodic refresh
 - Smaller (can be implemented with 1 or 3 transistors)
 - Slower
 - Single-Ended
 - Can be read and written
- Read-Only Memory (ROM)

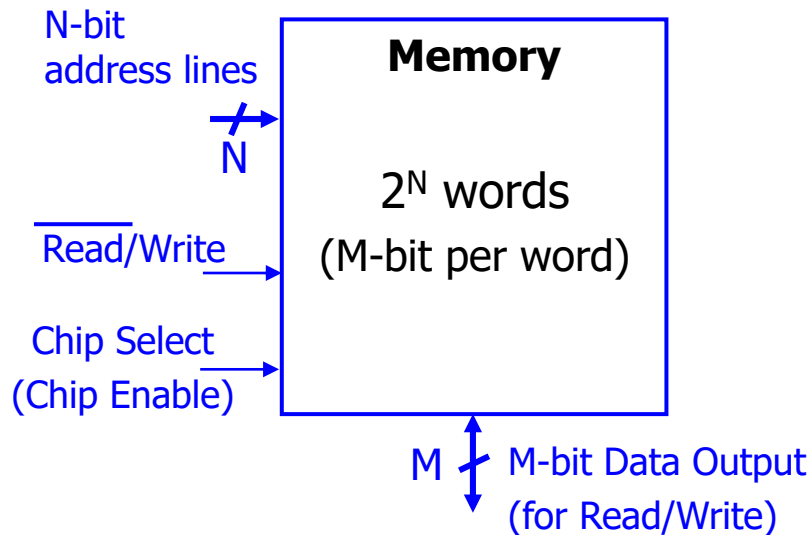
Examples

- DDR SDRAM (Double Data Rate Synchronous **DRAM**)
 - Widely being used for main memory in computers



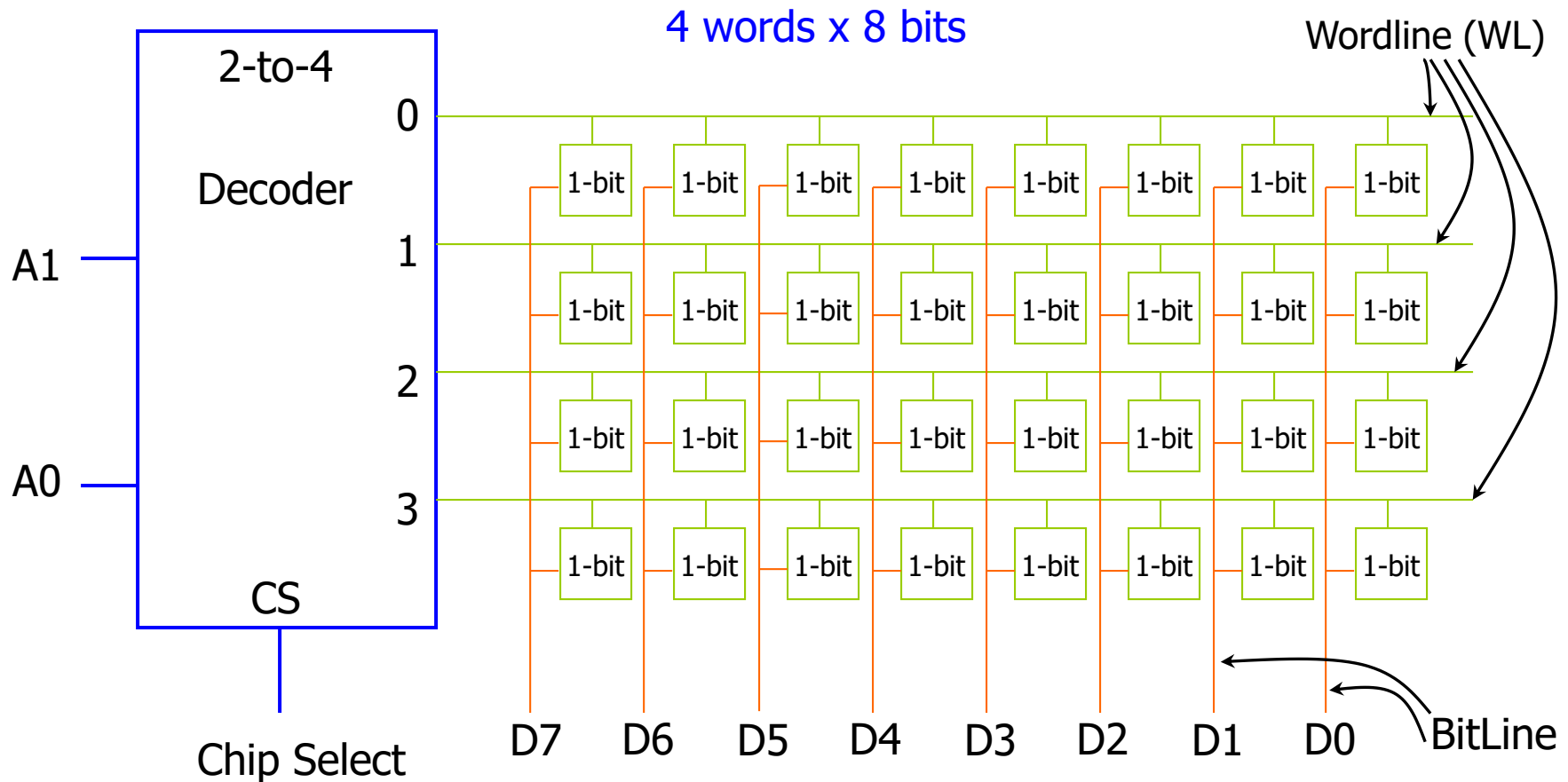
Block Diagram of Memory

- An M-bit data value can be read or written at each unique N-bit address

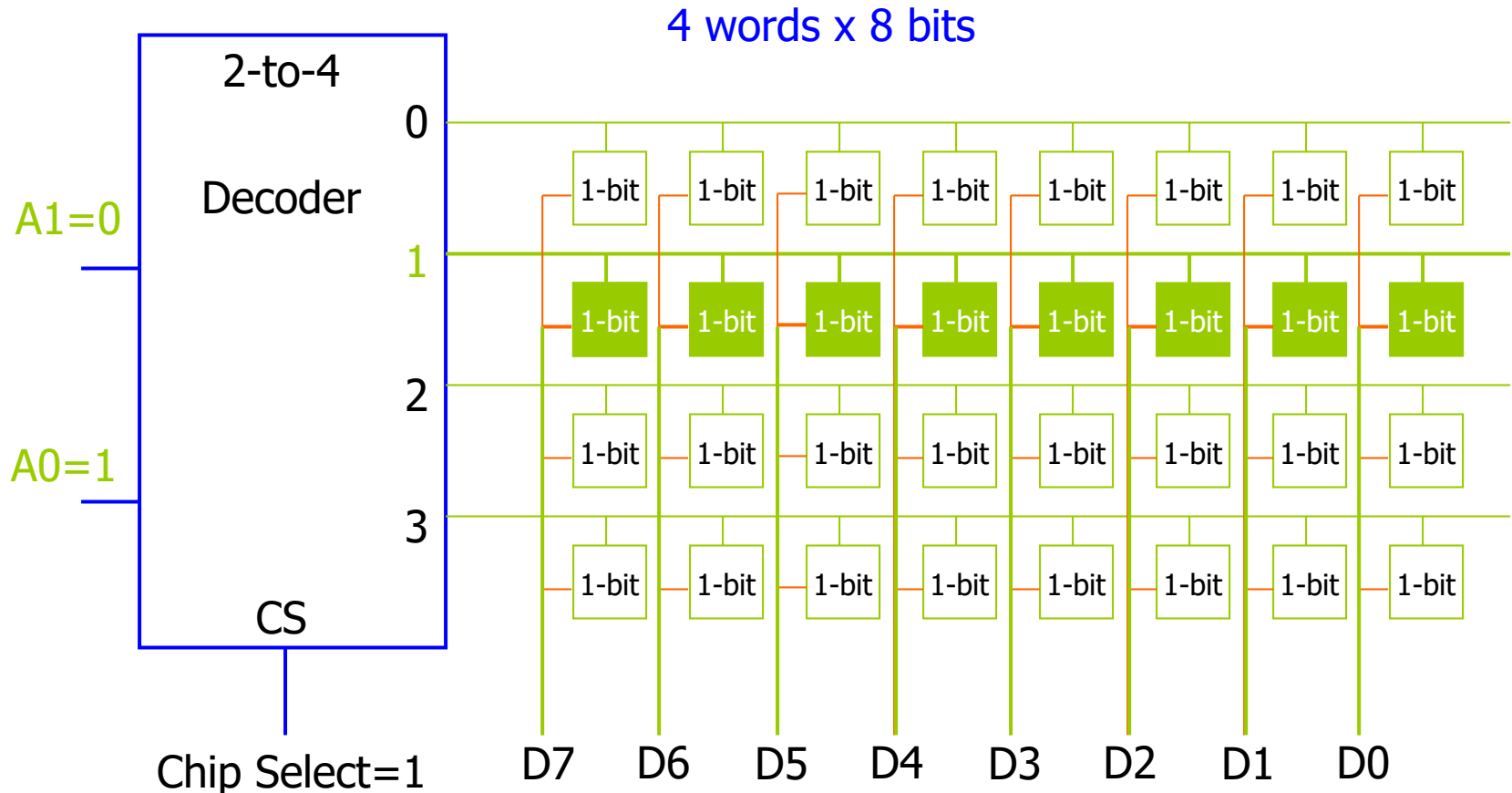


- Example: Byte-addressable 2MB memory
 - $M = 8$ (because of byte-addressability)
 - 1 word = 8-bit
 - $N = 21$
 - 2M locations

Memory Organization Example



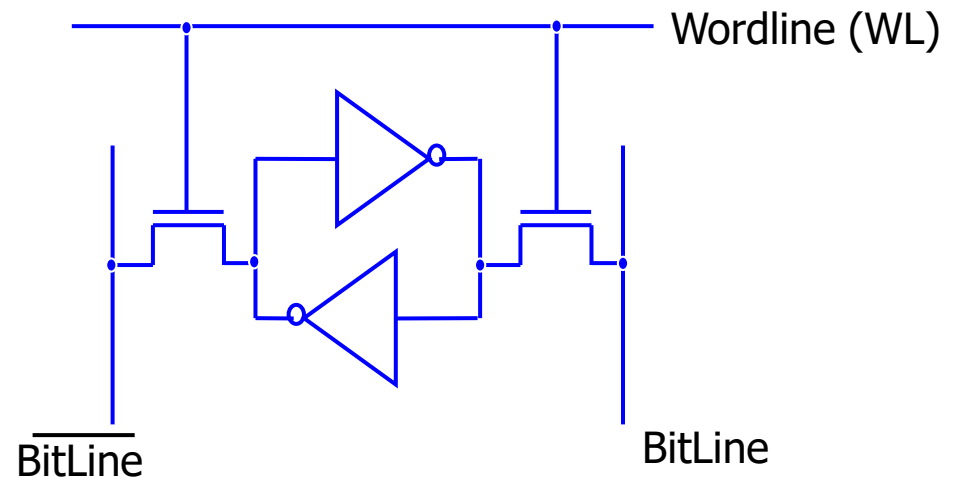
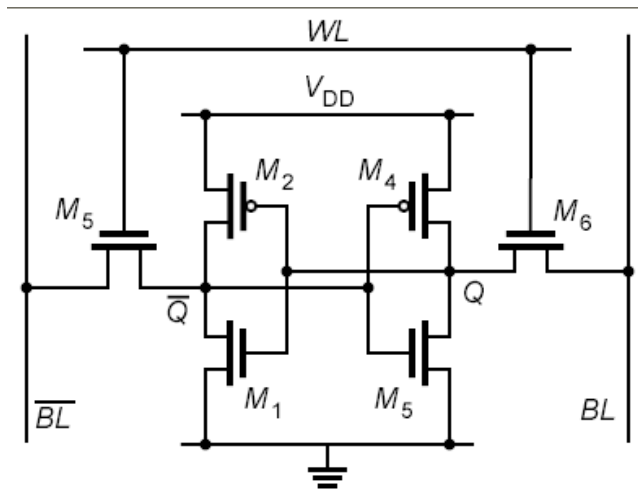
How to Address Memory



Access address = 0x1

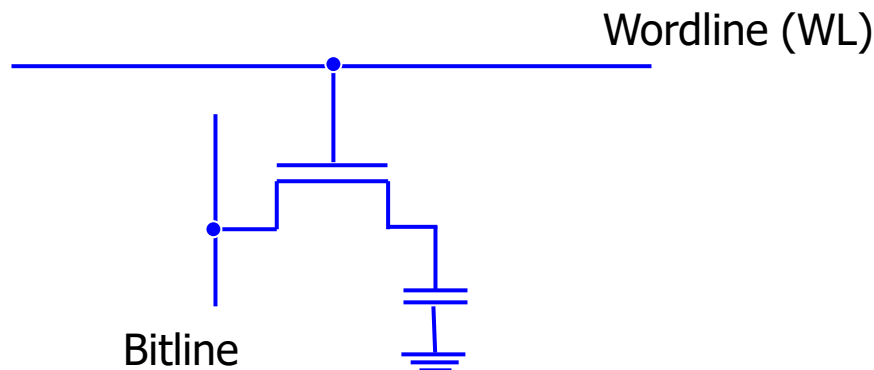
Static Random Access Memory (SRAM)

- Typically each bit is implemented with 6 transistors (6T)
- Read operation
 - The bitline and its inverse are precharged to V_{DD} (1)
 - Then set Wordline (WL) high
 - Depending on the value stored, either bitline or $\sim$ bitline goes low
- Write operation
 - Put the (strong) new value on Bitline and its inverse on $\sim$ Bitline
 - Then set the Wordline to high



Dynamic Random Access Memory (DRAM)

- 1-transistor DRAM cell
- Write operation
 - Put a (strong) new value on bitline and then set WL=1
- Read operation
 - Precharge bitline to Vdd (1)
 - Assert WL to 1
- Storage decays, thus requires **periodic refreshing**
 - Read bitline data and write it **periodically** to keep the value in the memory cell



Memory Description

- Capacity of a memory is described as
 - # addresses x Word size
- Examples:

Memory	# of addr	# of data lines	# of addr lines	# of total bytes
1M x 8	1,048,576	8	20	1 MB
2M x 4	2,097,152	4	21	1 MB
1K x 4	1024	4	10	512 B
4M x 32	4,194,304	32	22	16 MB
16K x 64	16,384	64	14	128 KB

Example



**2Gb: x4, x8, x16 DDR2 SDRAM
Features**

DDR2 SDRAM

MT47H512M4 – 64 Meg x 4 x 8 banks

MT47H256M8 – 32 Meg x 8 x 8 banks

MT47H128M16 – 16 Meg x 16 x 8 banks

Features

- Vdd = +1.8V ±0.1V, VddQ = +1.8V ±0.1V
- JEDEC-standard 1.8V I/O (SSTL_18-compatible)
- Differential data strobe (DQS, DQS#) option
- 4n-bit prefetch architecture
- Duplicate output strobe (RDQS) option for x8
- DLL to align DQ and DQS transitions with CK
- 8 internal banks for concurrent operation
- Programmable CAS latency (CL)

Options¹

- Configuration
 - 512 Meg x 4 (64 Meg x 4 x 8 banks)
 - 256 Meg x 8 (32 Meg x 8 x 8 banks)
 - 128 Meg x 16 (16 Meg x 16 x 8 banks)
- FBGA package (Pb-free) – x16
 - 84-ball FBGA (11.5mm x 14mm) Rev. A
- FBGA package (Pb-free) – x4, x8
 - 60-ball FBGA (11.5mm x 14mm) Rev. A
- Timing – cycle time
 - 3.0ns @ CL = 4 (DDR2-667)

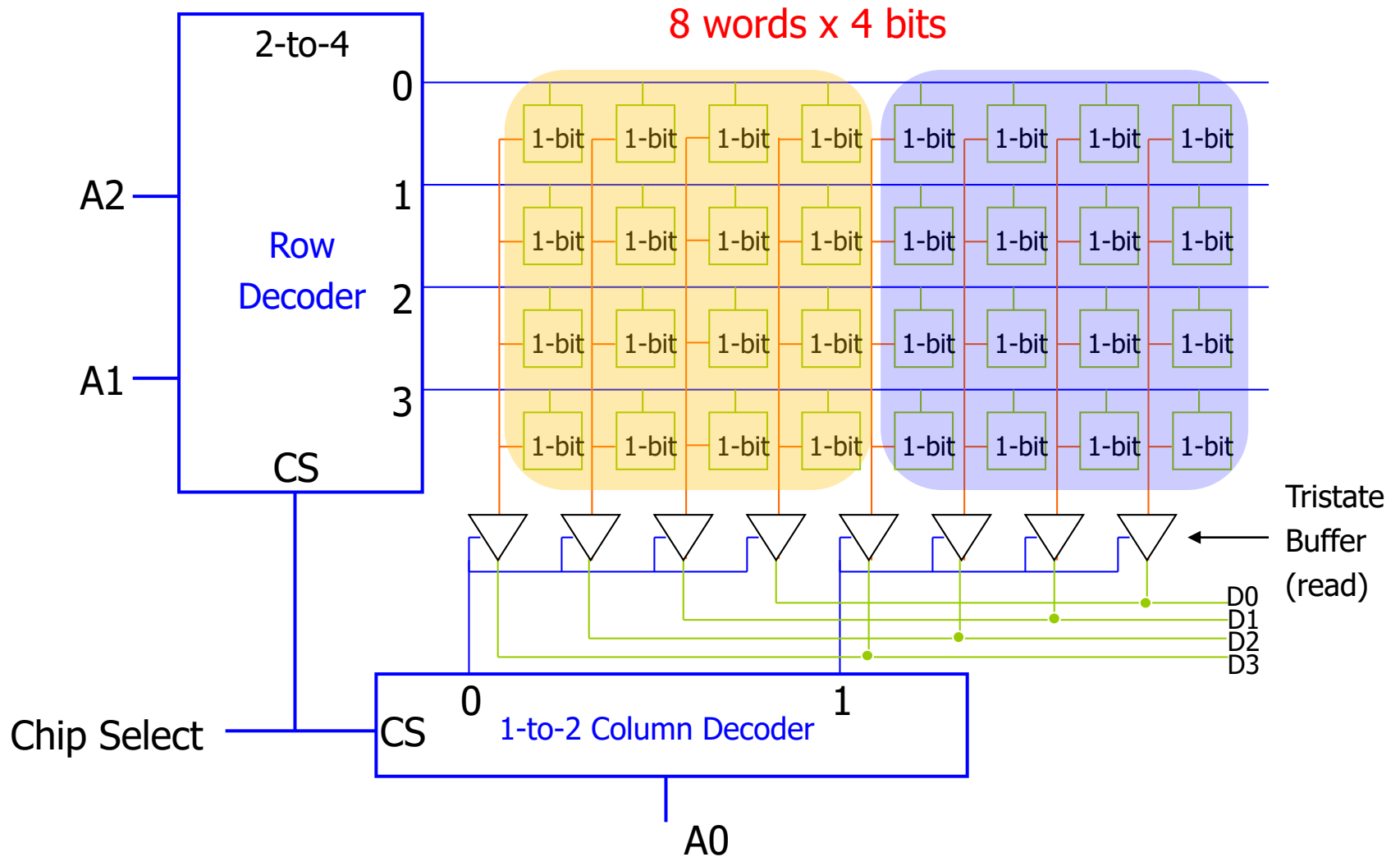
Marking

512M4
256M8
128M16

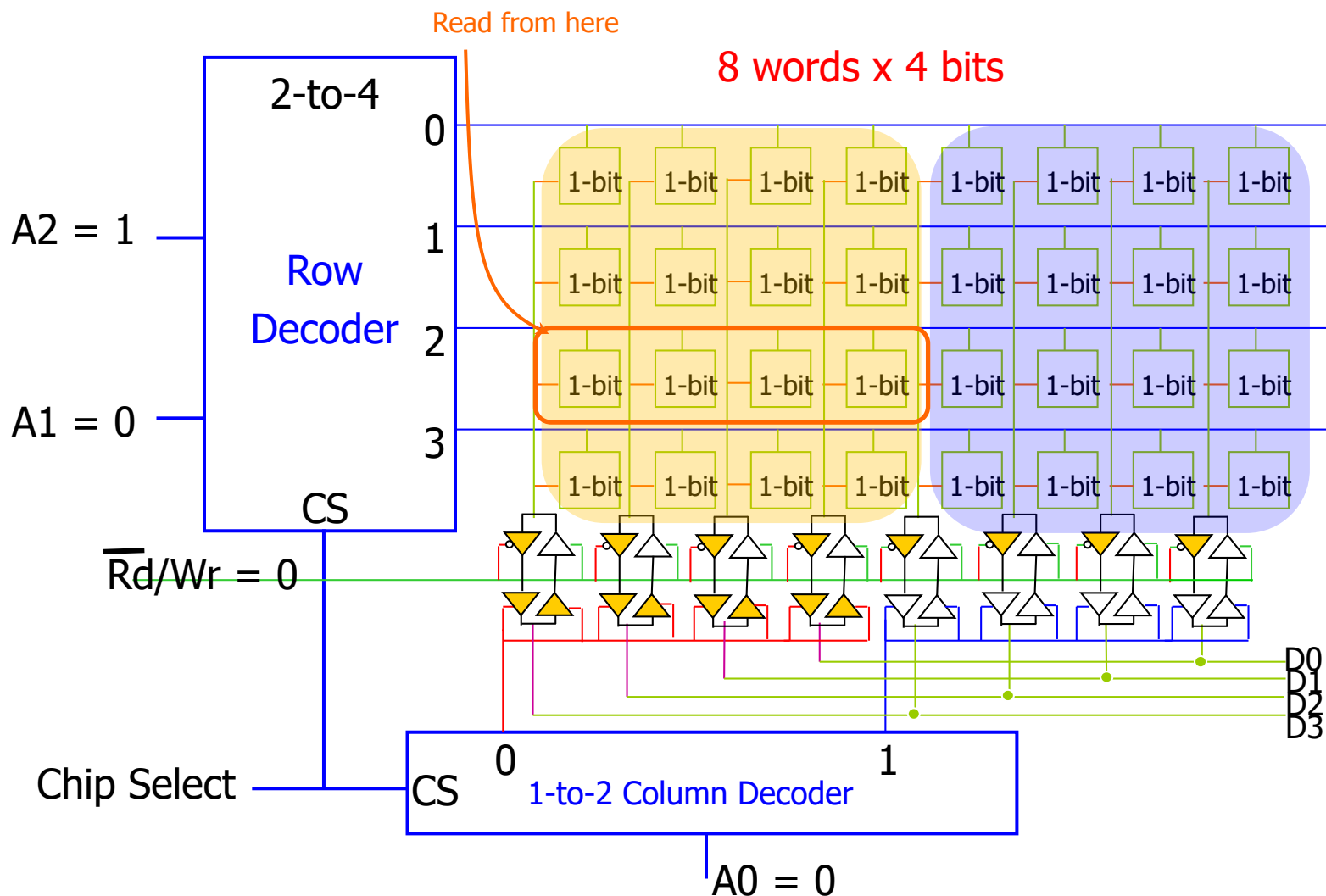
HG
HG

I -3E

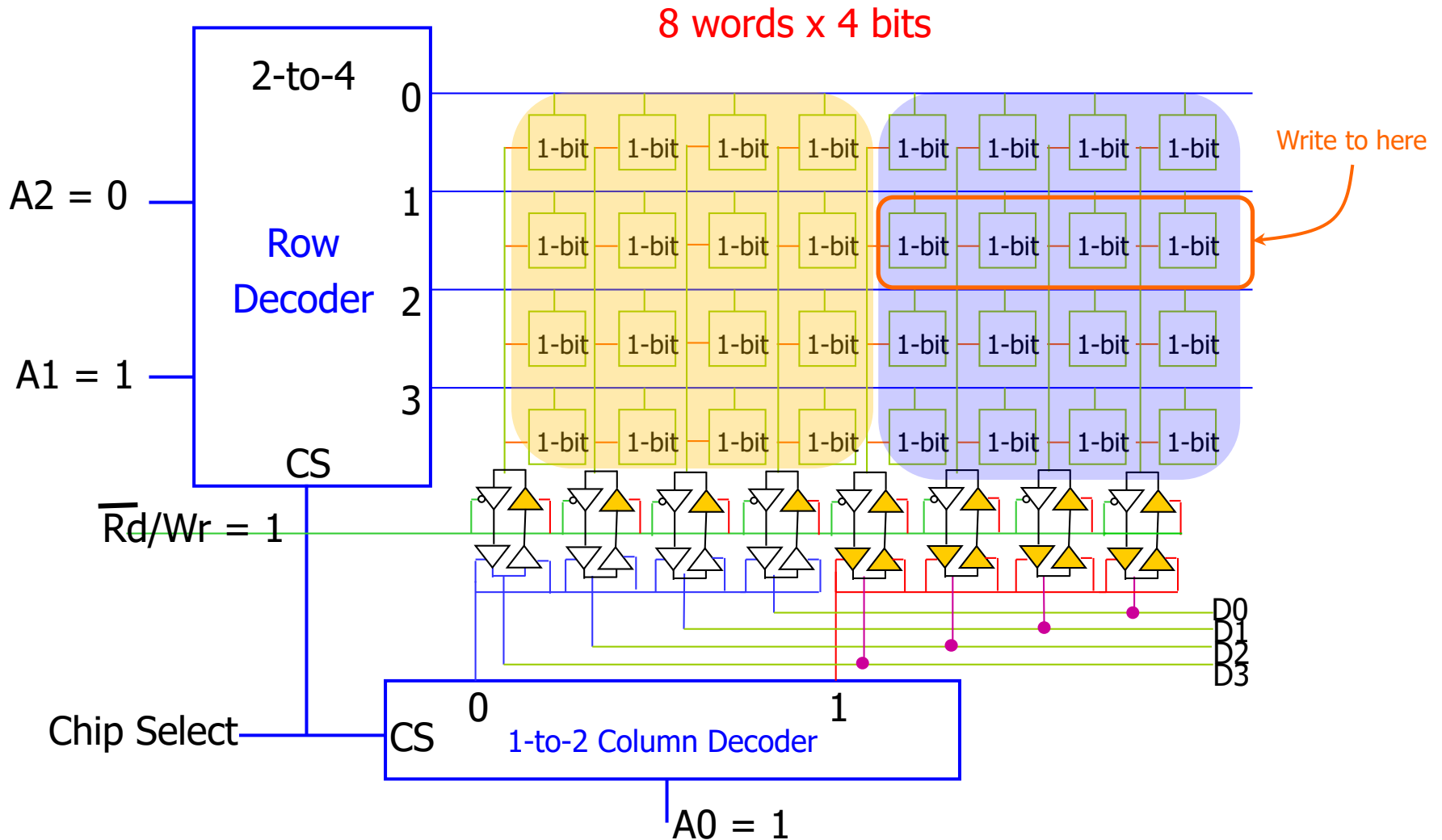
Memory with 2 Decoders



Read Operation

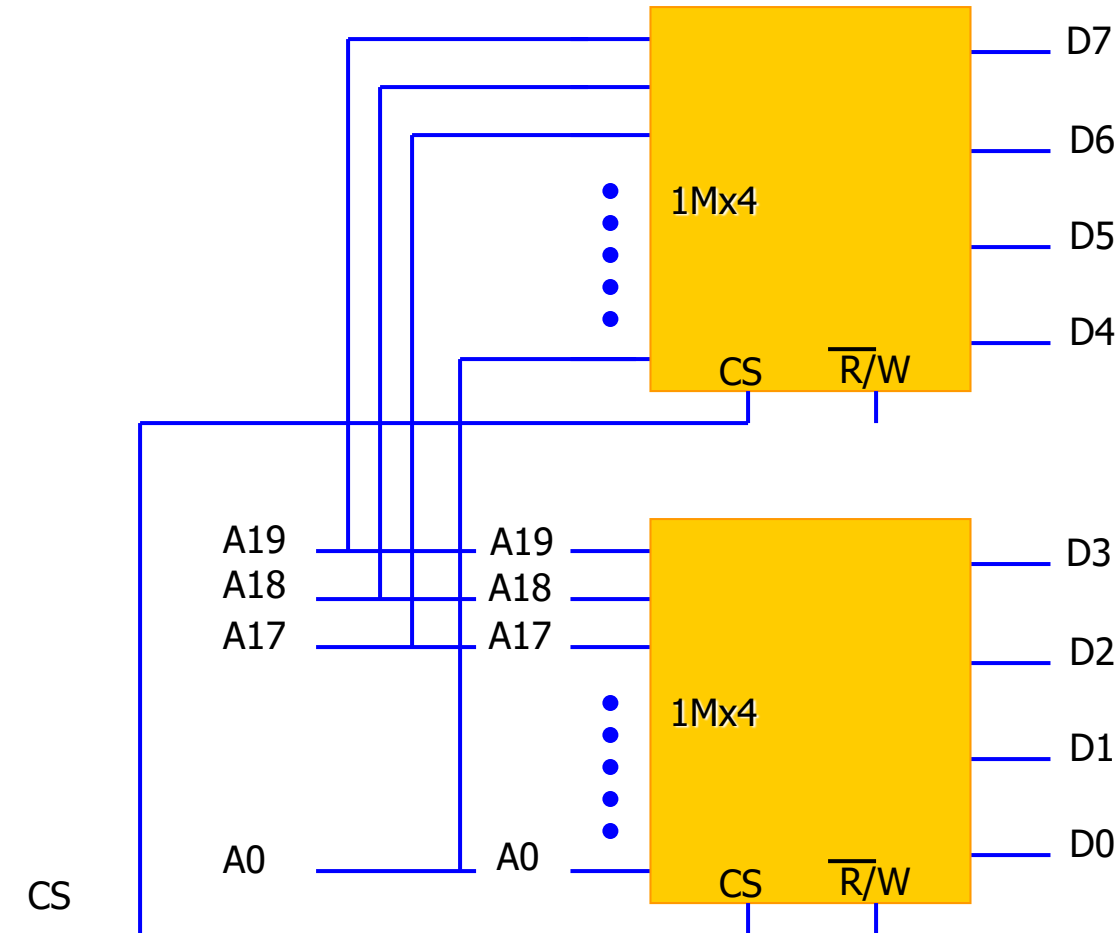


Write Operation



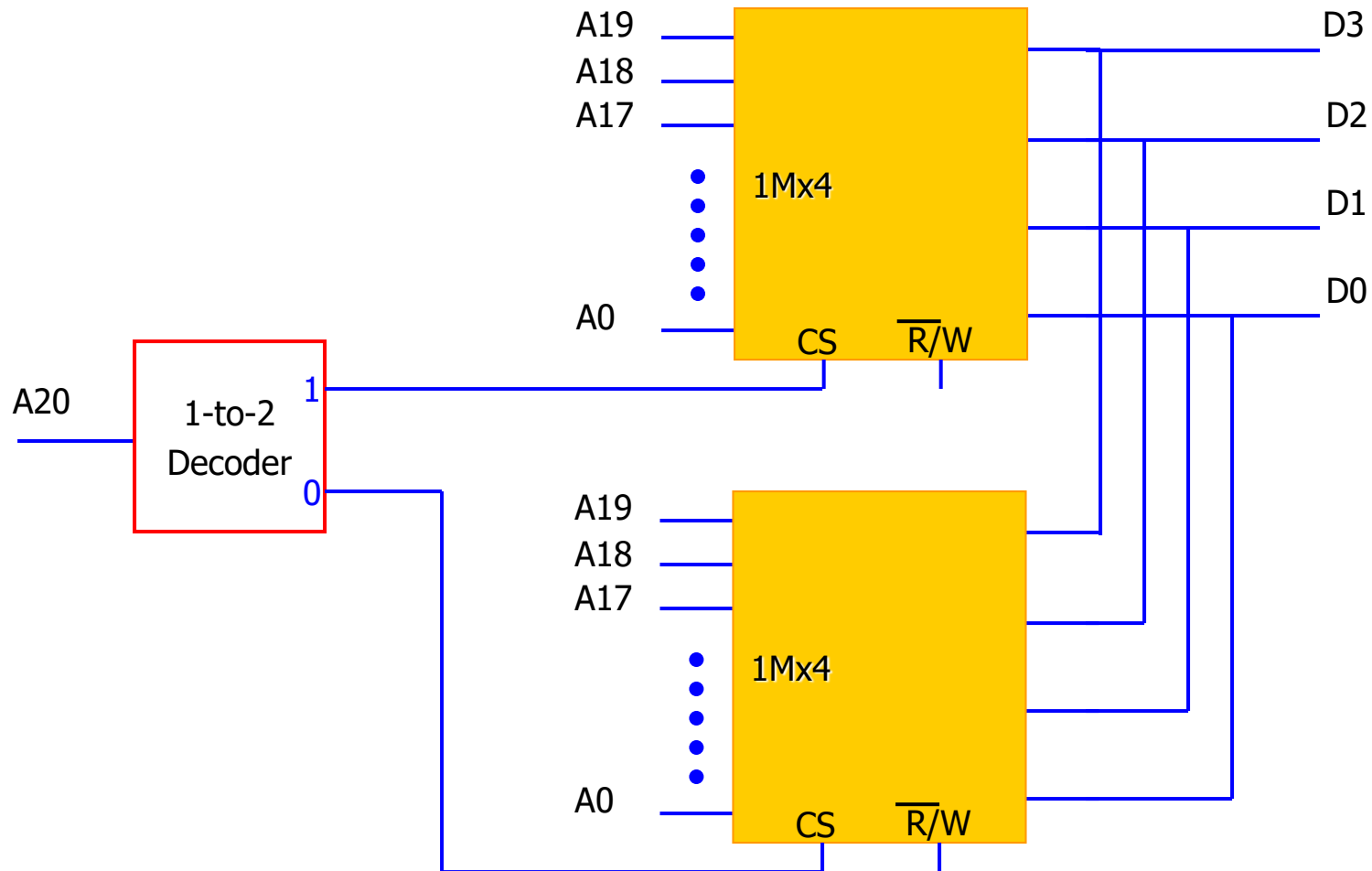
Building Memory in Hierarchy

- Design a 1Mx8 using 1Mx4 memory chips



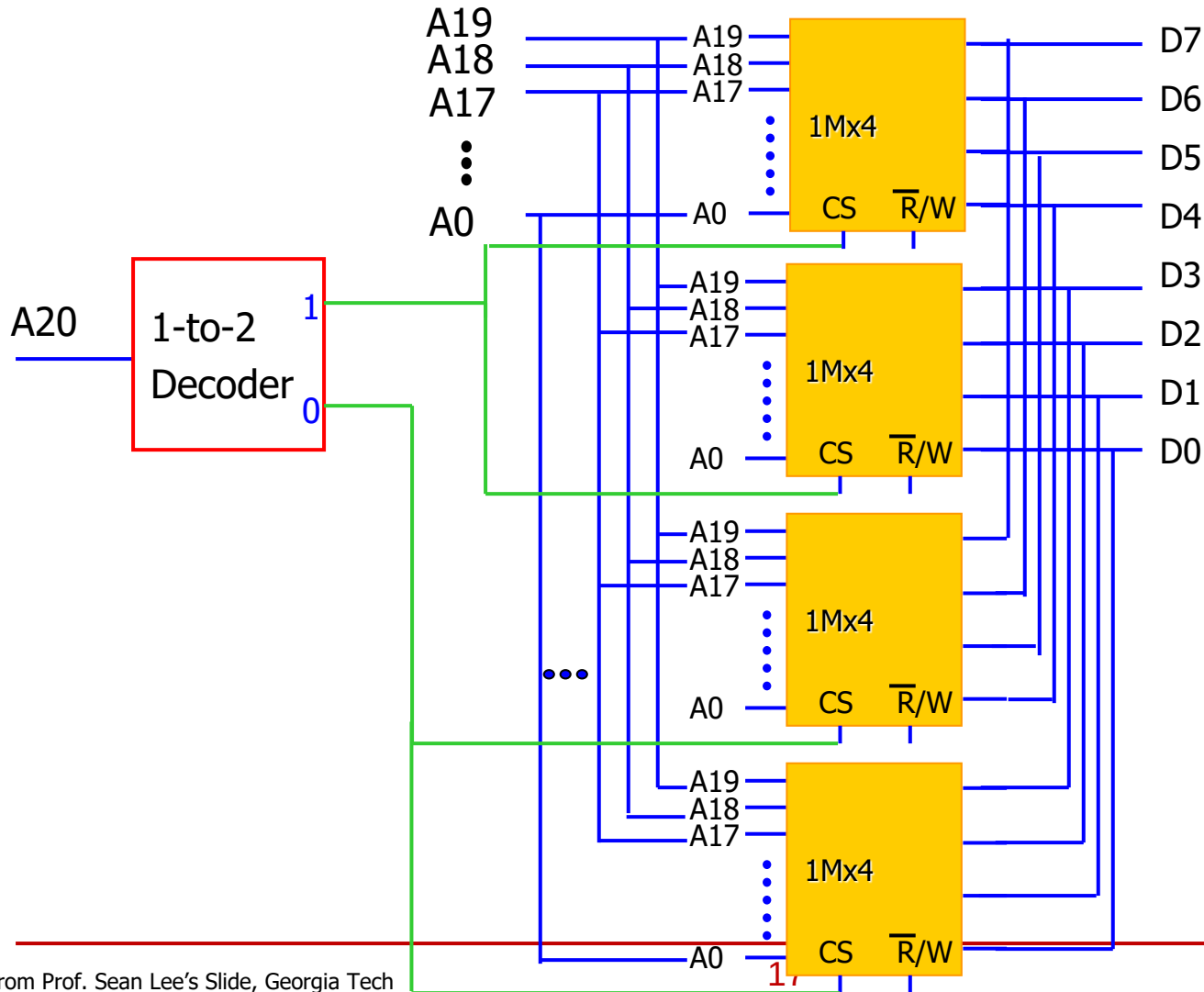
Building Memory in Hierarchy

- Design a 2Mx4 using 1Mx4 memory chips



Building Memory in Hierarchy

- Design a 2Mx8 using 1Mx4 memory chips



Verilog Representation of RAM

- RAM (synchronous with clock)

```
module ram #(parameter N = 8, M = 32)
    (input      clk,
     input      we,
     input [N-1:0] adr,
     input [M-1:0] din,
     output [M-1:0] dout);

    reg [M-1:0] mem[2*N-1:0];

    always @(posedge clk)
        if (we) mem[adr] <= din;

    assign dout = mem[adr];
endmodule
```

```
`timescale 1ns / 1ns
```

```
module ram_tb();
    reg  clk;
    reg  we;
    reg [7:0]  addr;
    reg [31:0] din;
    wire [31:0] dout;

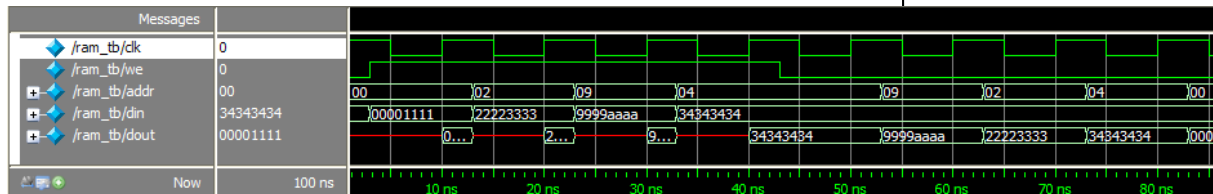
    parameter clk_period = 10;

    ram ram_uut(.clk (clk), .we (we), .adr (addr),
               .din (din), .dout (dout));

    always
    begin
        clk = 1;
        forever #(clk_period/2) clk = ~clk;
    end

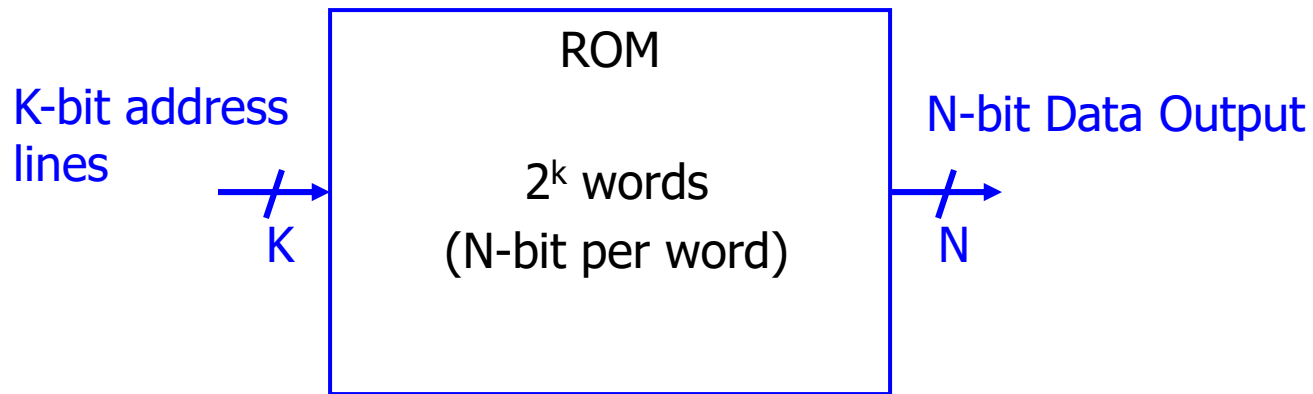
    initial
    begin
        addr = 4'h00; we = 1'b0; din = 32'h00000000; #3;
        addr = 4'h00; we = 1'b1; din = 32'h00001111; #(clk_period*1);
        addr = 4'h02; we = 1'b1; din = 32'h22223333; #(clk_period*1);
        addr = 4'h09; we = 1'b1; din = 32'h9999AAAA; #(clk_period*1);
        addr = 4'h34; we = 1'b1; din = 32'h34343434; #(clk_period*1);
        we = 1'b0;                                     #(clk_period*1);
        addr = 4'h09;                                     #(clk_period*1);
        addr = 4'h02;                                     #(clk_period*1);
        addr = 4'h34;                                     #(clk_period*1);
        addr = 4'h00;                                     #(clk_period*1);
    end

endmodule
```

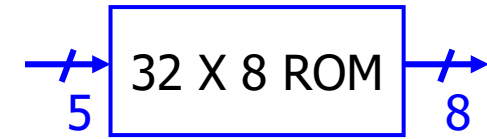


Read-Only Memory (ROM)

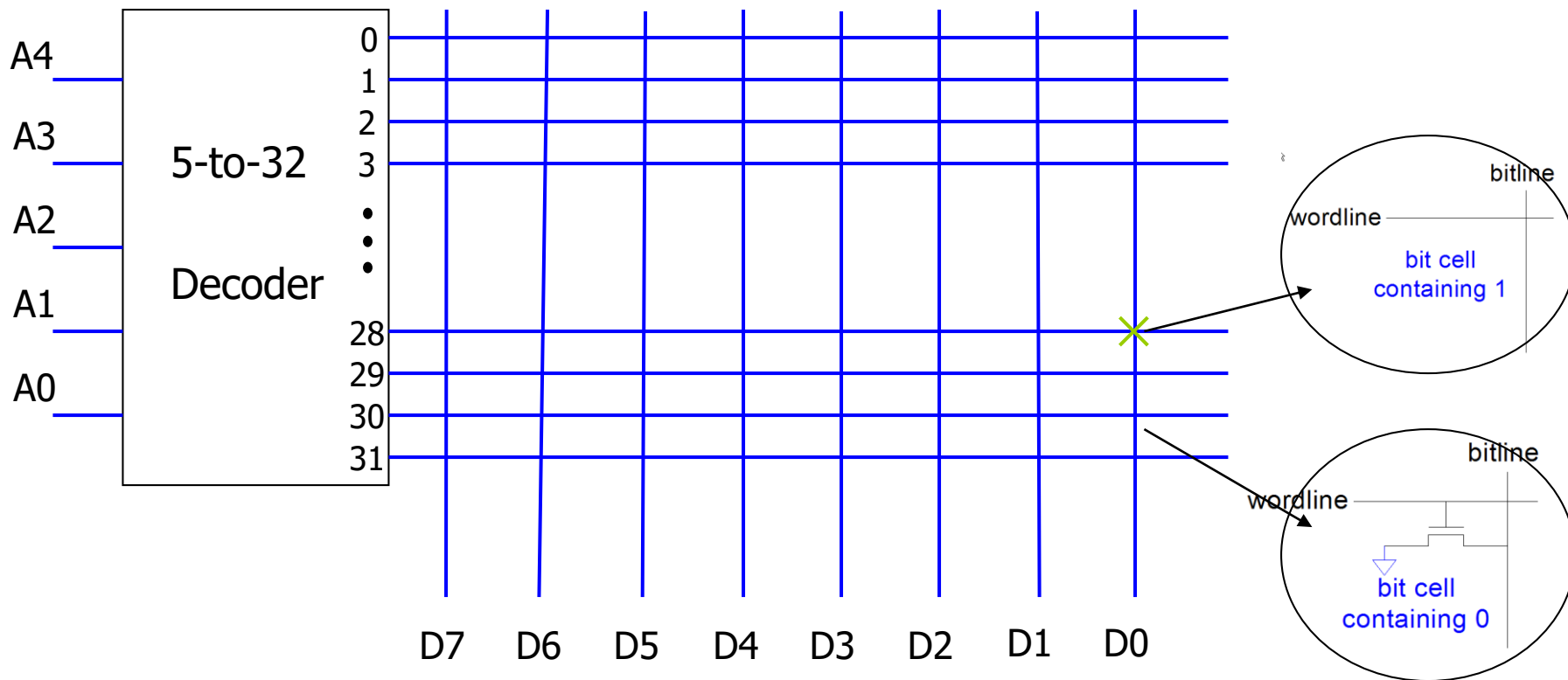
- Non-volatile memory
 - Permanent binary information is stored
 - Power off does not erase information stored



32 x 8 (32 words x 8 bits) ROM

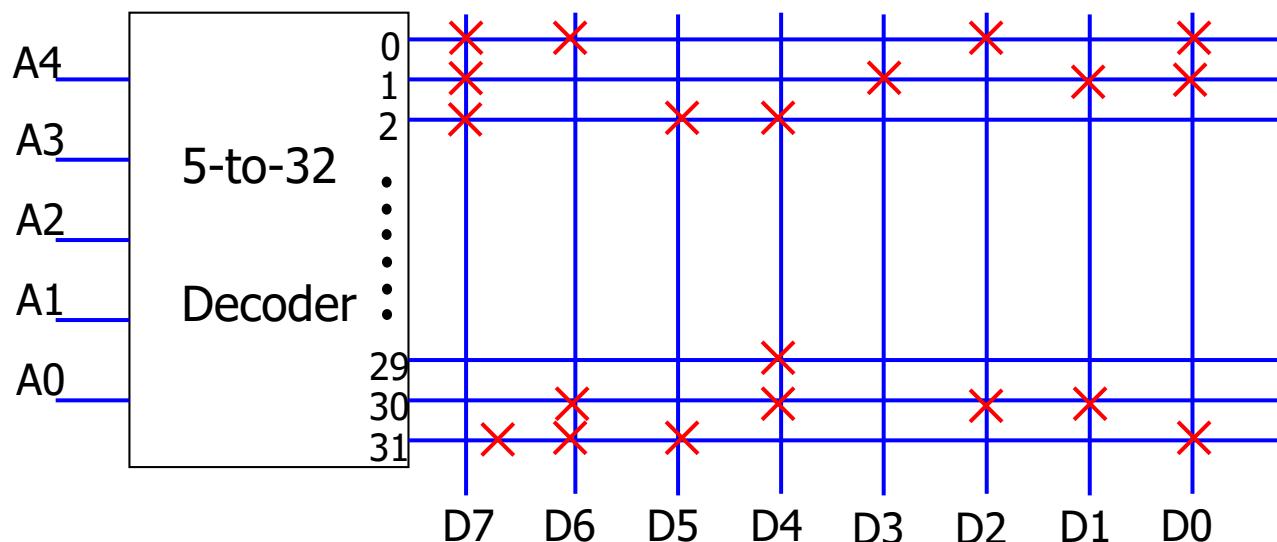


- To read the cell, the bitline is weakly pulled HIGH. Then, the wordline is turned ON
 - If the transistor is present, it pulls the bitline LOW
 - If the transistor is not present, the bitline remains HIGH



Programming the 32x8 ROM

A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	1	0	0	0	1	0	1
0	0	0	0	1	1	0	0	0	1	0	1	1
0	0	0	1	0	1	0	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1	1	1	0	1	0	0	0	1	0	0	0	0
1	1	1	1	0	0	1	0	1	0	1	1	0
1	1	1	1	1	1	1	1	0	0	0	0	1



Verilog Representation of ROM

- 4 X 3 (4 words x 3 bits) ROM

```
module rom(input      [1:0] adr,
            output reg [2:0] dout);

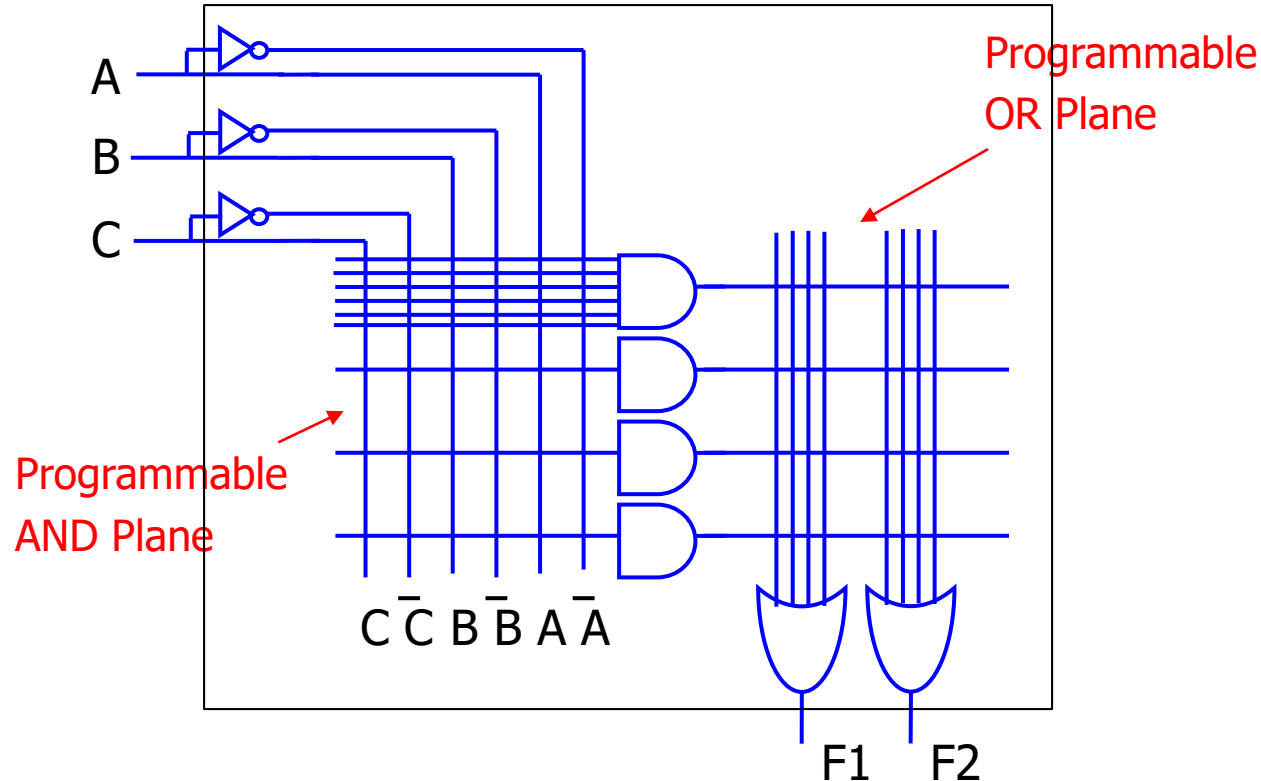
    always @(adr)
        case(adr)
            2'b00: dout <= 3'b011;
            2'b01: dout <= 3'b110;
            2'b10: dout <= 3'b100;
            2'b11: dout <= 3'b010;
        endcase
endmodule
```

Other Flavors of ROMs

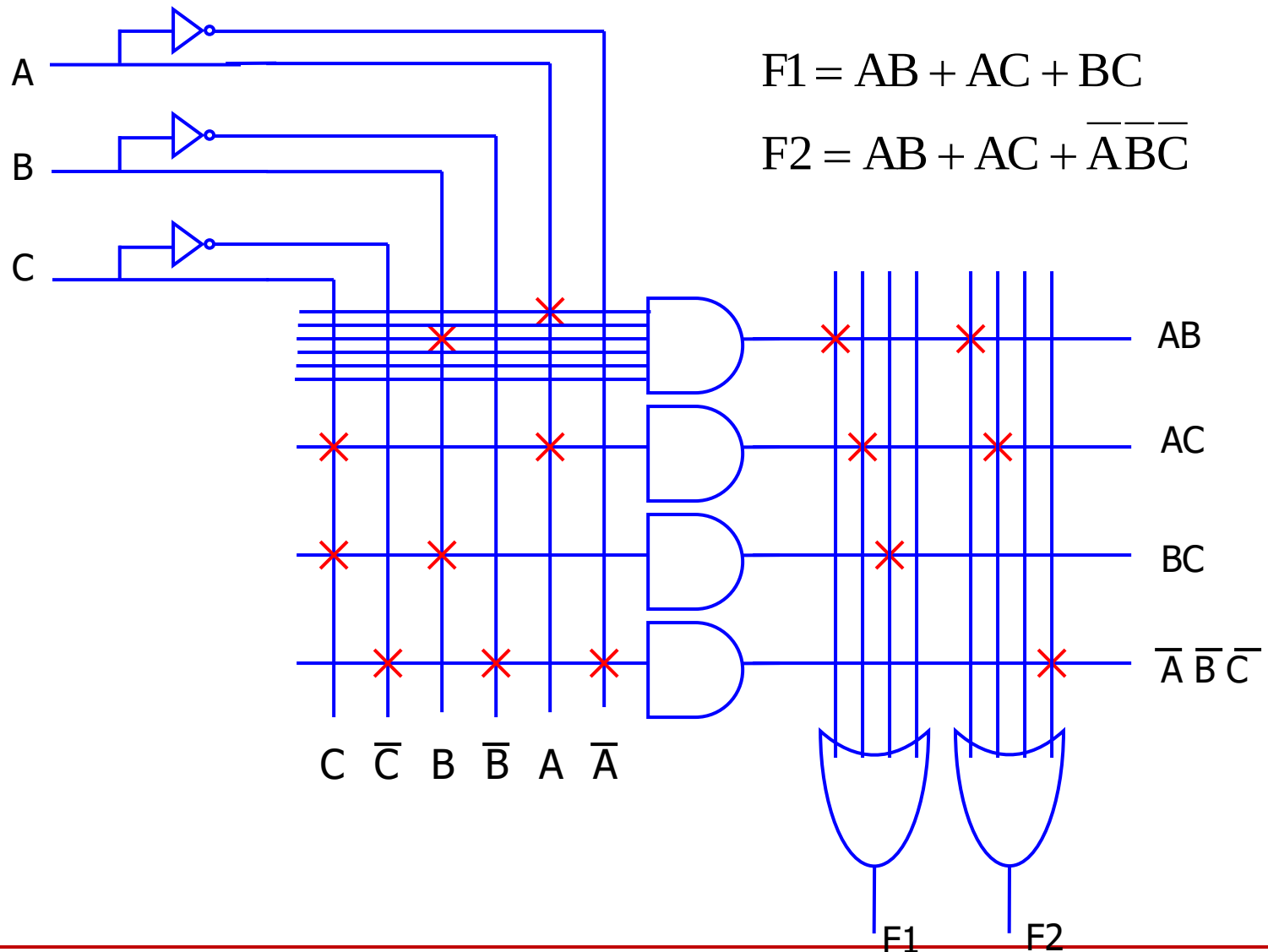
- Reprogrammable ROMs
 - EPROM (Erasable Programmable ROM)
 - Use UV (Ultra Violet) light for erasing
 - EEPROM (Electrically Erasable Programmable ROM)
- Flash memory
 - Read and Writable
 - Non-volatile
 - Power off does not erase information stored
- Modern ROMs are not really read-only
 - They can be reprogrammed as well
- Flash memory has become extremely popular to store large amounts of data in portable systems such as cameras and music players

Programmable Logic Array (PLA)

- PLA implement two-level combinational logic in SOP form
- Concept of PLA is used in the decoding stage of CPU design
- Older technology



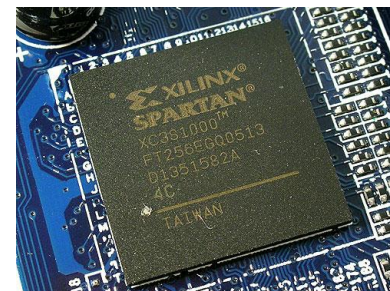
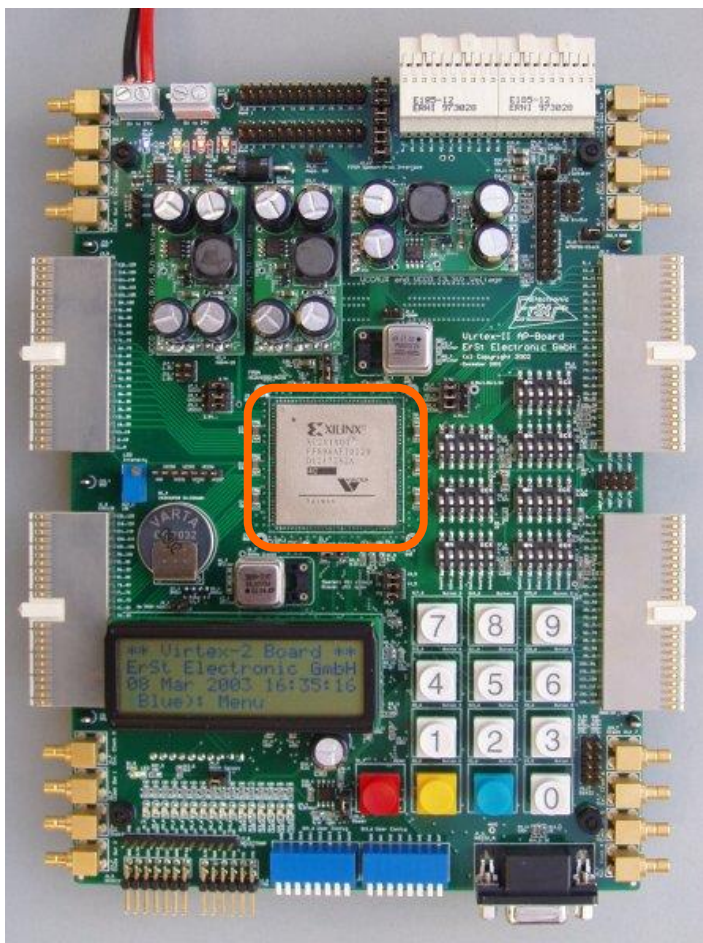
Example of using PLA



FPGA

- Field Programmable Gate Array (FPGA) can implement both combinational and sequential logic
 - PLA can implement only combinational logic
- Using CAD tools, a user can implement hardware design using either HDL and/or schematic and download it onto the FPGA
- It is programmable in the **field**
 - FPGA is reprogrammable (reconfigurable) after a system is deployed
- 2 Major FPGA companies
 - Xilinx: www.xilinx.com
 - Altera: www.altera.com

FPGA Examples



The End

- That's it! Folks
- I hope you have enjoyed this class
- You just got one step closer to understanding computers
- See you next semester in the computer architecture class

Backup Slides

DDR, DDR2, DDR3

With a 64-bit transferred at a time, $100\text{MHz} \times 2 \text{ (DDR)} \times 8\text{Bytes} = 1600 \text{ MB/sec}$

- DDR

Standard name	Memory clock	Cycle time	I/O Bus clock	Data transfers per second	JEDEC standard VDDQ voltage	Module name	Peak transfer rate
DDR-200	100 MHz	10 ns [1]	100 MHz	200 Million	2.5v +/- 0.2v	PC-1600	1600 MB/s
DDR-266	133 MHz	7.5 ns	133 MHz	266 Million	2.5v +/- 0.2v	PC-2100	2100 MB/s
DDR-300	150 MHz	6.67 ns	150 MHz	300 Million	Not a JEDEC standard	PC-2400	2400 MB/s
DDR-333	166 MHz	6 ns	166 MHz	333 Million	2.5v +/- 0.2v	PC-2700	2700 MB/s
DDR-400	200 MHz	5 ns	200 MHz	400 Million	2.6v +/- 0.1v	PC-3200	3200 MB/s

- DDR2

Standard name	Memory clock	Cycle time	I/O Bus clock	Data transfers per second	Module name	Peak transfer rate
DDR2-400	100 MHz	10 ns	200 MHz	400 Million	PC2-3200	3200 MB/s
DDR2-533	133 MHz	7.5 ns	266 MHz	533 Million	PC2-4200 PC2-4300 ¹	4266 MB/s
DDR2-667	166 MHz	6 ns	333 MHz	667 Million	PC2-5300 PC2-5400 ¹	5333 MB/s
DDR2-800	200 MHz	5 ns	400 MHz	800 Million	PC2-6400	6400 MB/s
DDR2-1066	266 MHz	3.75 ns	533 MHz	1066 Million	PC2-8500 PC2-8600 ¹	8533 MB/s

- DDR3

Standard name	Memory clock	Cycle time	I/O Bus clock	Data transfers per second	Module name	Peak transfer rate
DDR3-800	100 MHz	10 ns	400 MHz	800 Million	PC3-6400	6400 MB/s
DDR3-1066	133 MHz	7.5 ns	533 MHz	1066 Million	PC3-8500	8533 MB/s
DDR3-1333	166 MHz	6 ns	667 MHz	1333 Million	PC3-10600	10667 MB/s [1]
DDR3-1600	200 MHz	5 ns	800 MHz	1600 Million	PC3-12800	12800 MB/s

ROM Implementation

- To minimize the size and cost, ROMs and PLAs commonly use pseudo-nMOS

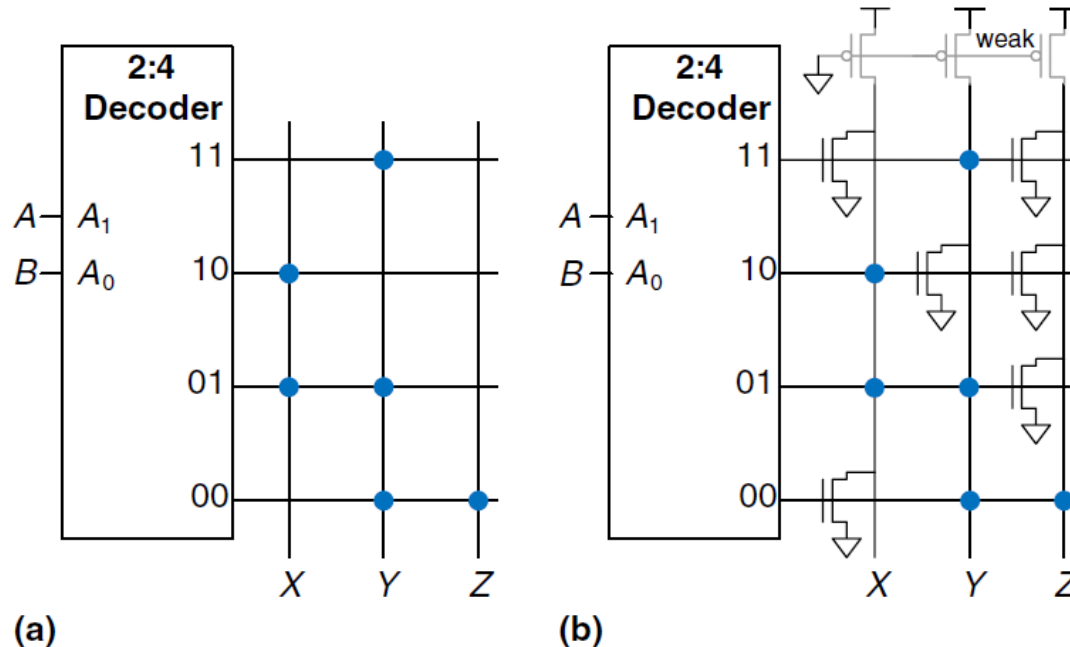
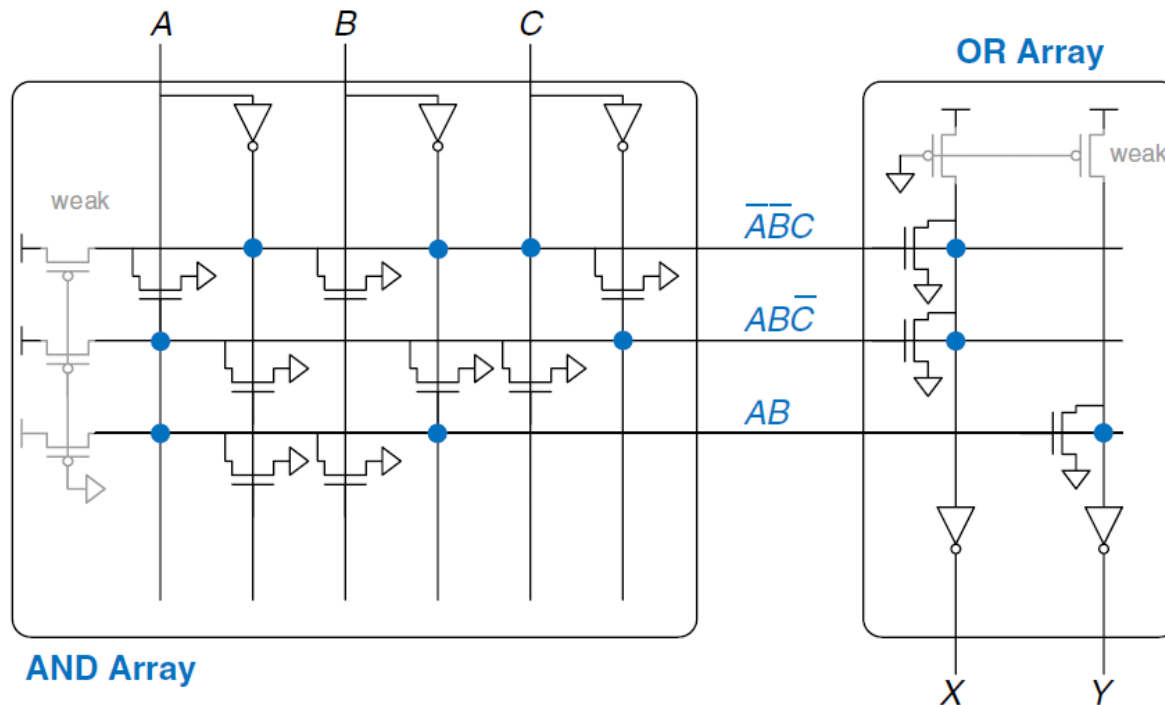


Figure 5.62 ROM implementation: (a) dot notation, (b) pseudo-nMOS circuit

PLA Implementation

- To minimize the size and cost, ROMs and PLAs commonly use pseudo-nMOS



$$X = \overline{A}\overline{B}C + A\overline{B}\overline{C} \quad Y = \overline{A}\overline{B}$$