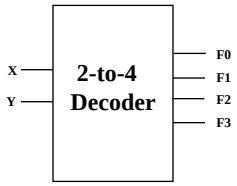
Decoders

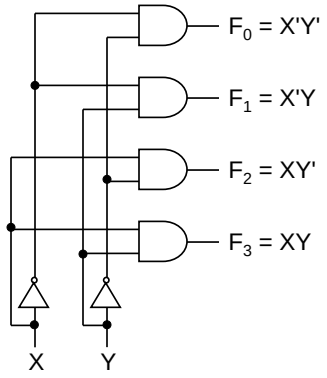
- A *decoder* is a combinational circuit that converts binary information from *n* inputs to a maximum of 2^n outputs.
- A decoder is called *n*-to-*m* decoder, where $m \leq 2^n$
- Consider 3-8 decoder, truth table with 3 inputs *x,y,z* and 8 outputs D_7 .. D_0 the circuit is shown in Figure 4-18

Decoders

X	Y	F_0	F_1	F_2	F_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

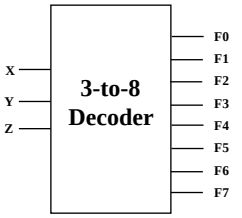
- From truth table, circuit for 2x4 decoder is:
- Note: Each output is a 2-variable minterm (**X'Y', X'Y, XY'** or **XY**)

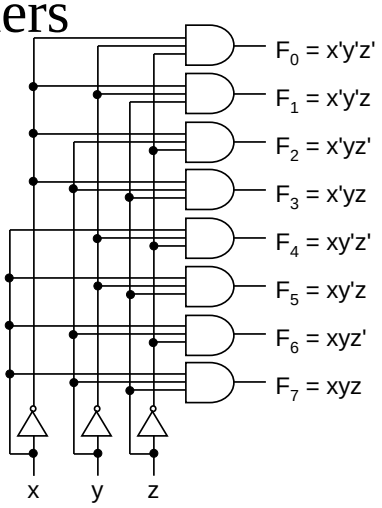




Decoders

x	y	z	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1





Decoders

- Some decoders are implemented using NAND gates, in this case it will be more economical to produce the output in their complemented form.
- 2-4 decoder
- Circuit 4-19
- When E is 1, non

Of the outputs I 0

- Decoder may be

Activated With E=0 or 1

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(a) Conventional symbol (b) Array logic symbol

Fig. 7-1 Conventional and Array Logic Diagrams for OR Gate

Memory address		Memory contest
Binary	decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

Fig. 7-3 Content of a 1024 × 16 Memory

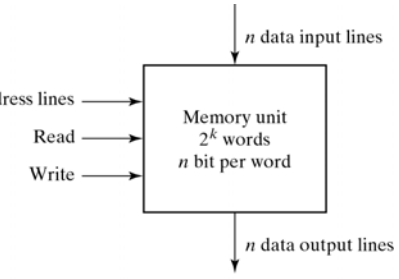


Fig. 7-2 Block Diagram of a Memory Unit

```
//Read and write operation of memory
Module memory (Enable, ReadWrite, Address, DataIn, Dataout);
input Enable, ReadWrite;
input [3:0] DataIn;
input [5:0] Address;
Output [3:0] Dataout;
reg [3:0] Dataout;
reg [3:0] Mem[0:63];
always @(Enable or ReadWrite)
        if(Enable)
                if(ReadWrite)
                Dataout=Mem[Address];
                else
                Mem[Address]=DataIn;
        else
        Dataout=4'bz;
endmodule
```


Memory enable and the read/write signal must be activated after the address lines have stabilized, otherwise we may destroy the contents of other memory locations.

Address and data signal must remain stable for a short time after the control signals are activated.

The 2 control signals must stay active for 50μ sec (in this example).

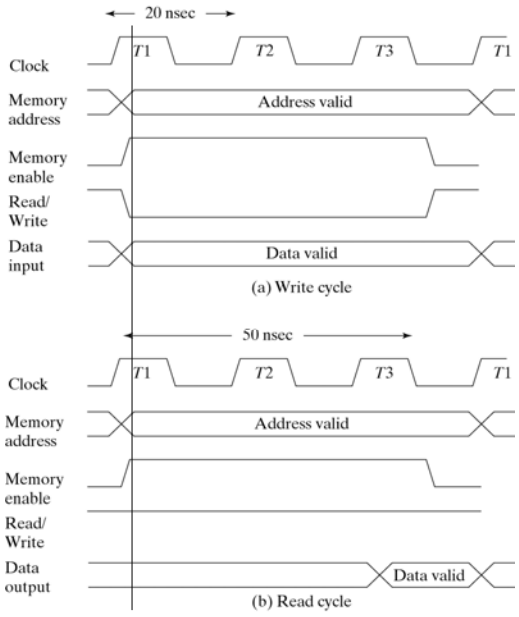


Fig. 7-4 Memory Cycle Timing Waveforms

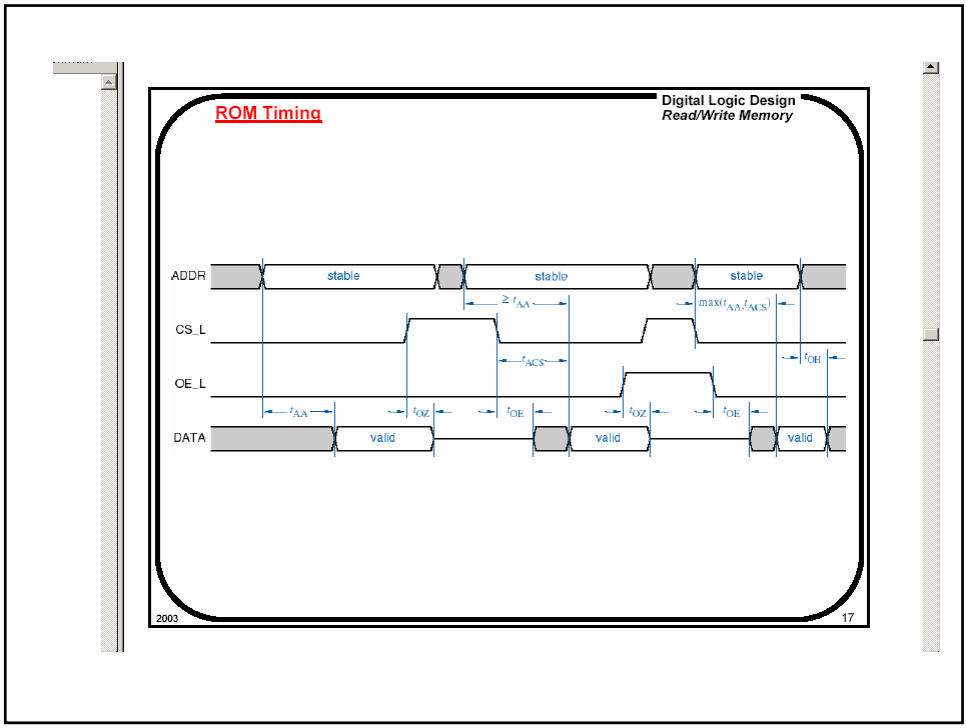
Types of Memory

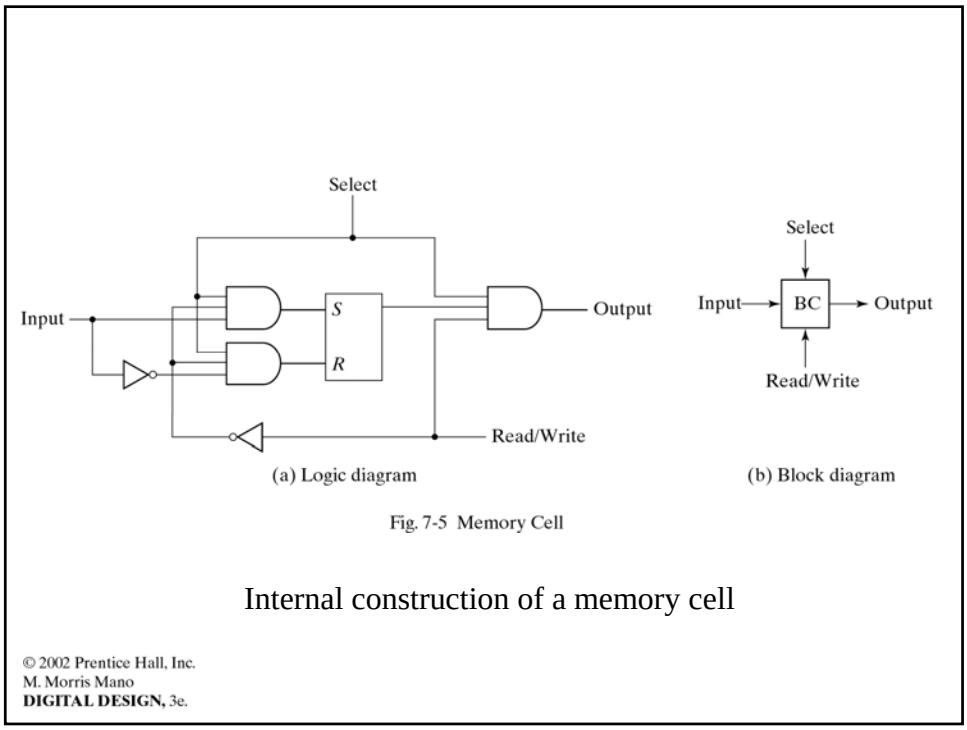
- ROM, PROM, EPROM, EEPROM
- RAM (SRAM, DRAM)
- SRAM store information in latches, while DRAM stores information as electric charges on capacitors (needs refreshing).

Figure 10-14

ROM Access

- t_{AA} Access time from address: delay from stable address input to stable data output.
- t_{ACS} Access time from chip select: delay from CS being asserted until data output are valid.
- t_{OE} Output enable time: propagation delay from OE and CS both asserted until the three state output drives have left the Hi-Z state.
- t_{OZ} Output-hold time: propagation delay from the time OE and CS is negated until the three-state output drives have entered the Hi-Z state
- t_{OH} hold output time: The length of time that the outputs remain valid after a change in the address inputs (or after OE of CS are negated).





Internal construction of a memory cell

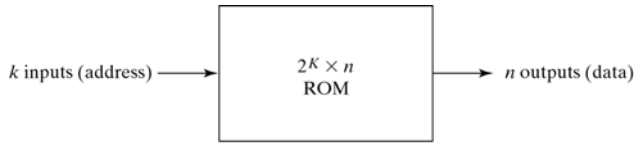


Fig. 7-9 ROM Block Diagram

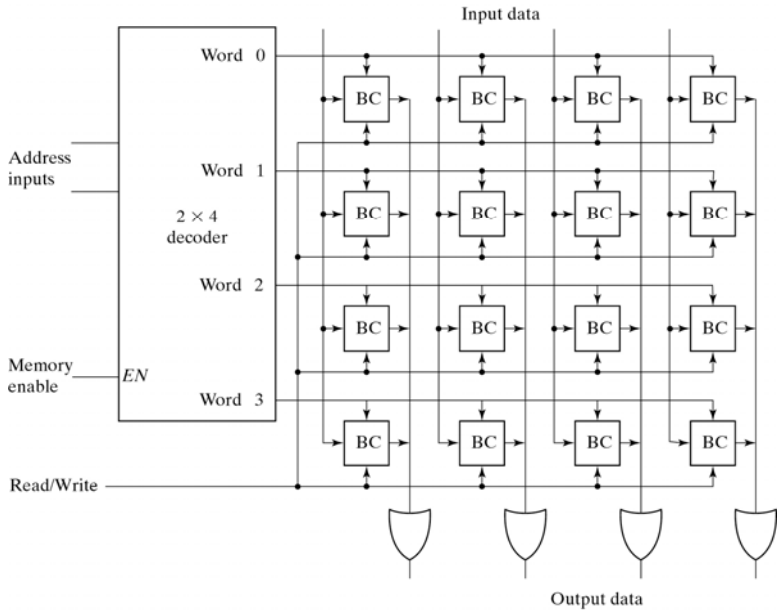


Fig. 7-6 Diagram of a 4 × 4 RAM

Coincident decoding, notice that we need $2(2^5)$ AND gate instead of 2^{10} AND gates

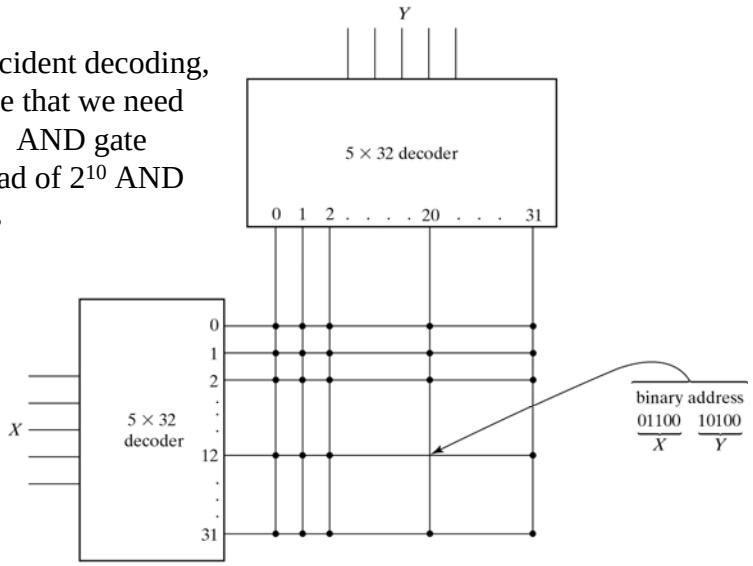


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

To reduce the number of pins in an IC

The 8 bits row address is applied to the input, and RAS changed to 0. The column address is applied, and CAS set to 0

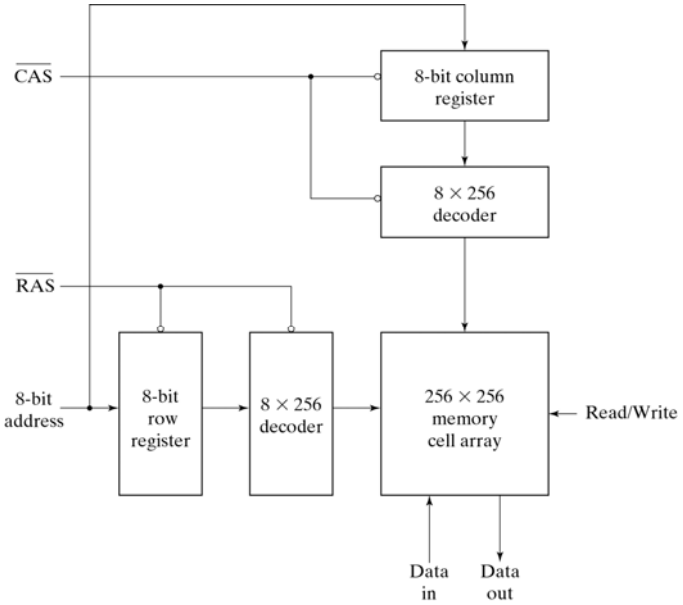
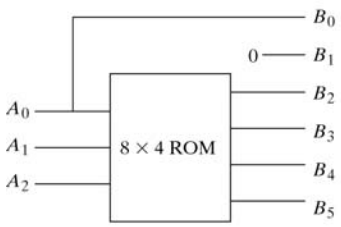


Fig. 7-8 Address Multiplexing for a 64K DRAM



(a) Block diagram

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

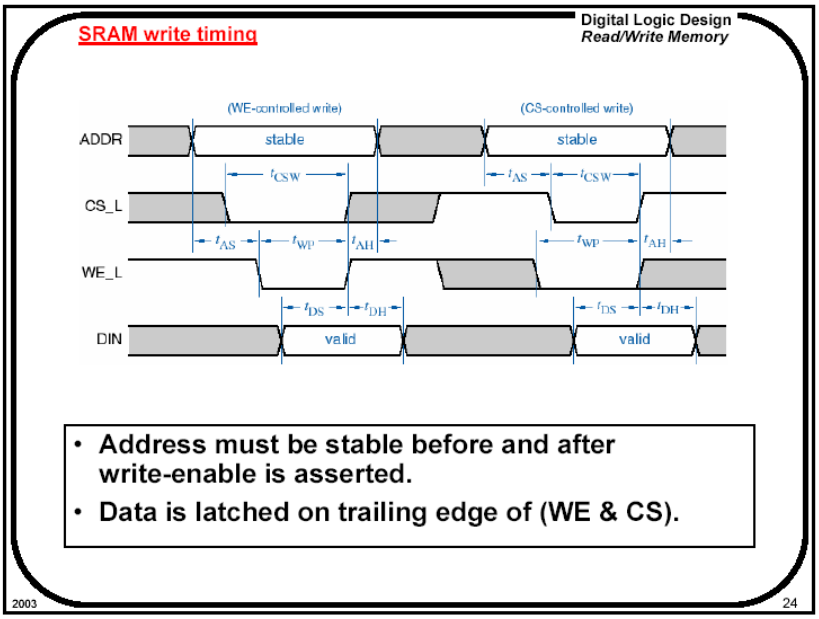
(b) ROM truth table

Fig. 7-12 ROM Implementation of Example 7-1

10-23

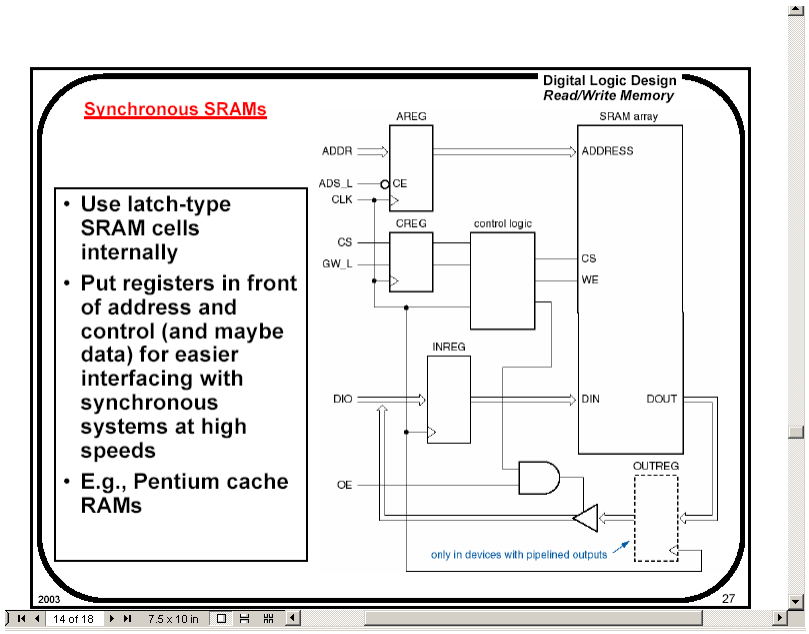
RAM (Write)

- t_{AS} Address setup time before write: (address inputs must be stable before CS and WE are asserted)
- t_{AH} Address hold time after write: Address hold time after write
- t_{CSW} CS setup before end of write: CS must be asserted that long.
- t_{WP} Write cycle width: W-enable must be asserted that long
- t_{DS} Data setup time before end of write: Data must be stable that long before end of write cycle
- t_{DH} Data hold time after end of write:



Synchronous SRAM

- SSRAM
- An operation that is setup before the rising edge of the clock is performed internally during a subsequent clock period.
- INREG captures the input data for write operation.
- Depending on whether the device is *pipelined* or *flow-through* OUTREG may or may not be present.
- Supports *burst* mode, in which data in a sequence of addresses are read, in this case AREG behaves like a counter.

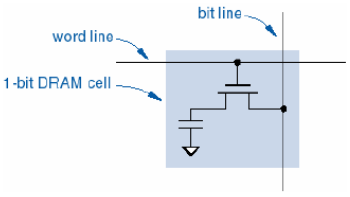


Dynamic RAM

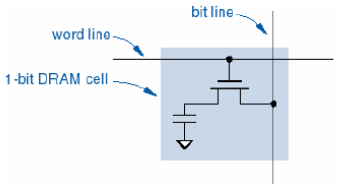
- The basic cell in SRAM is the D latch, 4-6 transistor per bit.
- DRAM stores information in a 1 transistor per bit

DRAM (Dynamic RAMs)

- SRAMs typically use six transistors per bit of storage.
- DRAMs use only one transistor per bit.
- 1/0 = capacitor charged/discharged

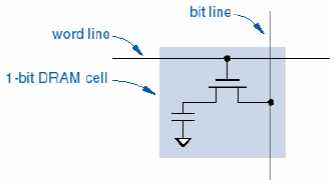


DRAM read operations



- Precharge bit line to $V_{DD}/2$.
- Take the word line HIGH.
- Detect whether current flows into or out of the cell.
- Note: cell contents are destroyed by the read!
- Must write the bit value back after reading.

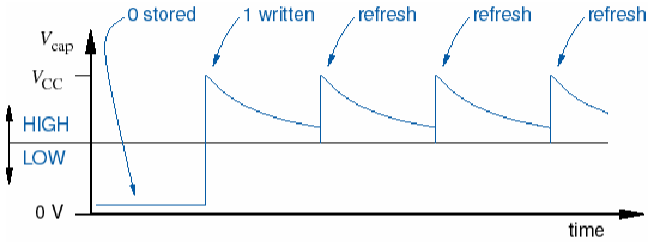
DRAM write operations



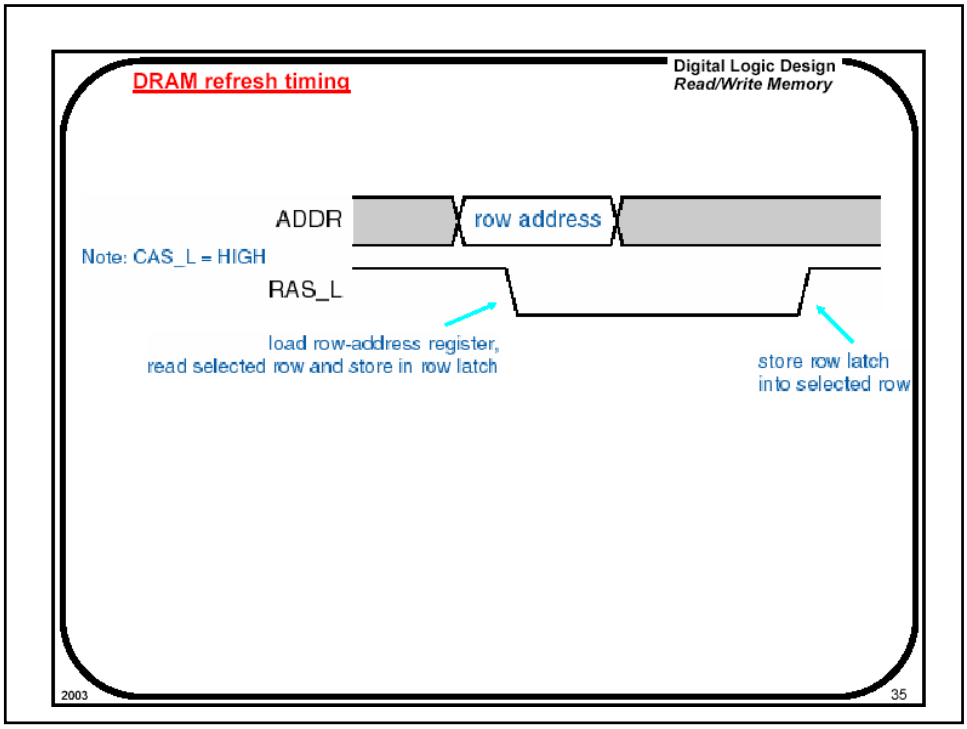
- Take the word line HIGH.
- Set the bit line LOW or HIGH to store 0 or 1.
- Take the word line LOW.

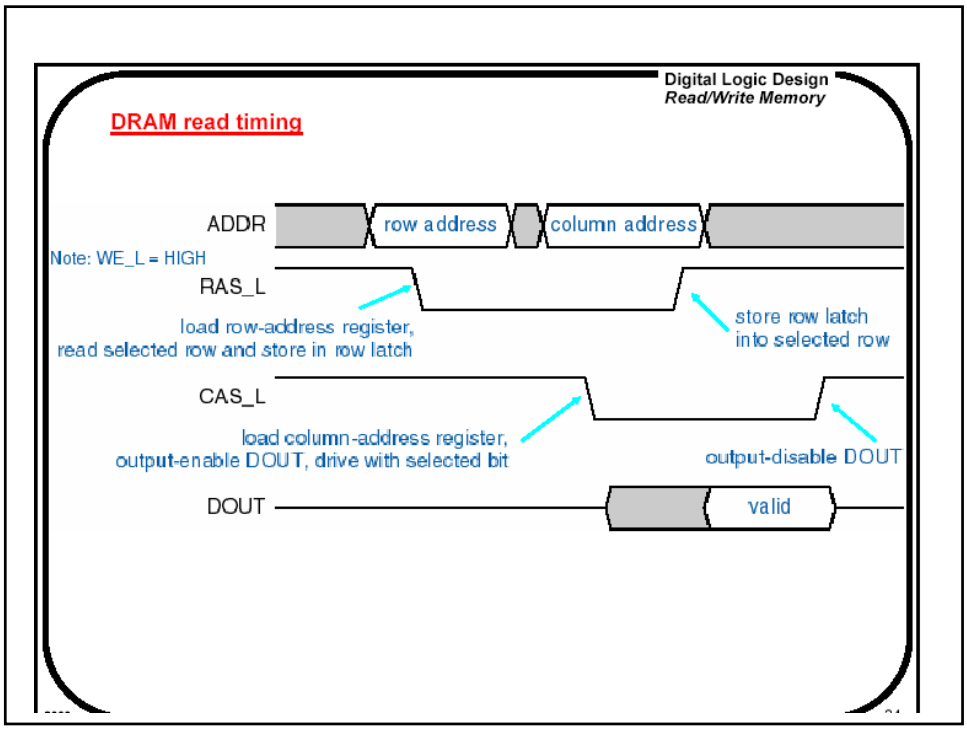
- Note: The stored charge for a 1 will eventually leak off.

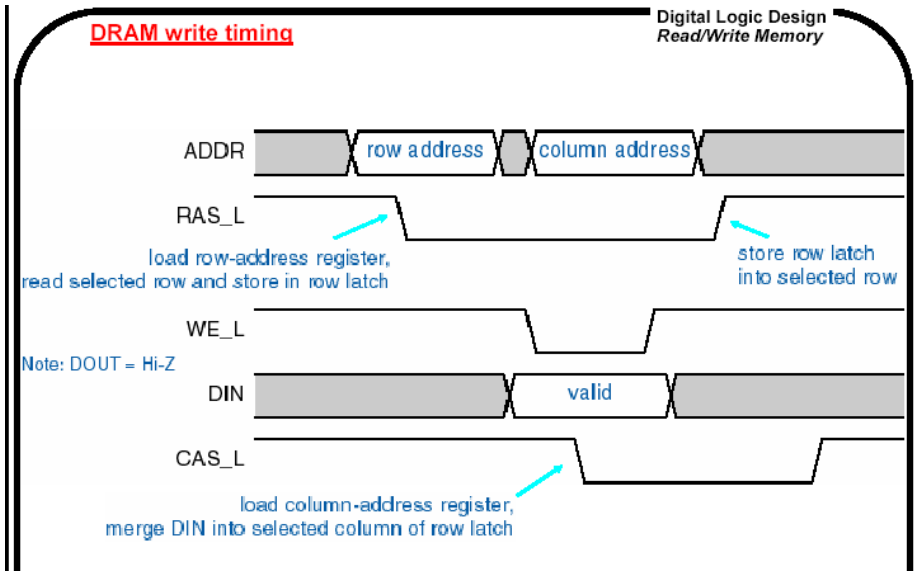
DRAM charge leakage



- Typical devices require each cell to be "refreshed" once every 4 to 64 mS.
- During "suspended" operation, notebook computers use power mainly for DRAM refresh.


DRAM refresh timing
Digital Logic Design
Read/Write Memory
ADDR
row address
Note: CAS_L = HIGH
RAS_L
load row-address register,
read selected row and store in row latch
store row latch
into selected row
2003
35


Digital Logic Design
Read/Write Memory
DRAM read timing
ADDR
row address
column address
Note: WE_L = HIGH
RAS_L
load row-address register,
read selected row and store in row latch
store row latch
into selected row
CAS_L
load column-address register,
output-enable DOUT, drive with selected bit
output-disable DOUT
DOUT
valid



Error Detection and Correction

- Used to improve the reliability of the memory unit.
- Parity could be used for a single error detection, multiple parity bits could be used for error correction.
- **Hamming Code**: *k* parity bits are added to *n* data bits to form *n+k* bit word.
- The bit positions are numbered from *1* to *n+k*. The positions numbered as power of 2 are reserved for the parity bits (remaining bits are data bits).

Error Detection and Correction

- **Example**:

Position	1	2	3	4	5	6	7	8	9	10	11	12
	p_1	p_2	b_0	p_4	b_1	b_2	b_3	p_8	b_4	b_5	b_6	b_7
			1		1	0	0		0	1	0	0
	0	0		1				1				

P_1=XOR of bits (3,5,7,9,11)

P_2=XOR of bits (3,6,7,10,11)

P_4=XOR of bits (5,6,7,12)

P_8=XOR of bits (9,10,11,12)

Error Detection and Correction

- We read the word, and calculate
- C_1=XOR of bits (1,3,5,7,9,11)
- C_2=XOR of bits (2,3,6,7,10,11)
- C_4=XOR of bits (4,5,6,7,12)
- C_8=XOR of bits (8,9,10,11,12)
- The number $C = C_8C_4C_2C_1$ is the location of the error, if 0 no error

Error Detection and Correction

-

	C_8	C_4	C_2	C_1
No error	0	0	0	0
Error in bit 1	0	0	0	1
Error in bit 5	0	1	0	1

C is called the syndrome. It has values in the range of *0* to 2^k-1.) means no error, any other value indicates the error position. Which means that $2^{k-1} \geq n+k$

Error Detection and Correction

- How to arrange the xored bits, for example bit three must be present in P_1 and P_2, bit 5 must be present in P_4 and P_1.
- **Single-Error Correction, Double-Error Detection** If we add P_{13} as an extra bit, to be the XOR of the 12 other bits, After reading the word from the memory, the parity bit over the 13 bits are calculated.

Error Detection and Correction

- For even parity

C=0 and P=0	No error
C≠0 and P=1	A single error can be corrected
C≠0 and P=0	A double error can not be corrected
C=0 and P=1	Error in the P_{13} bit.

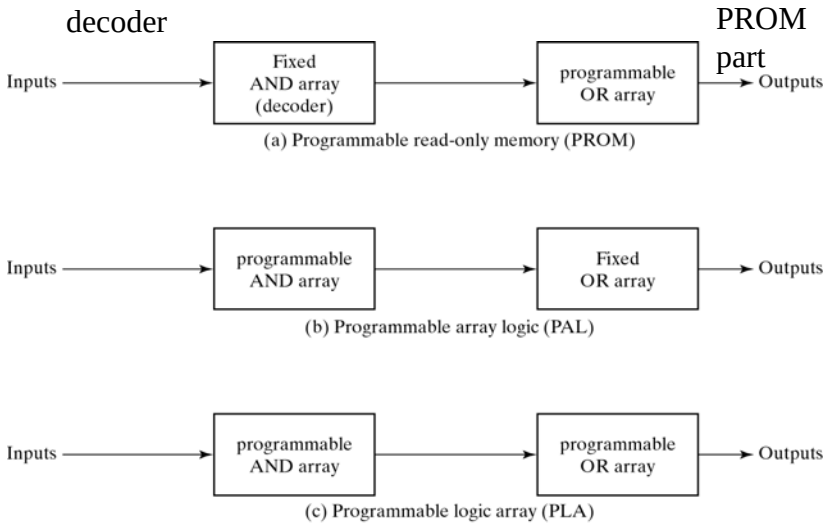


Fig. 7-13 Basic Configuration of Three PLDs

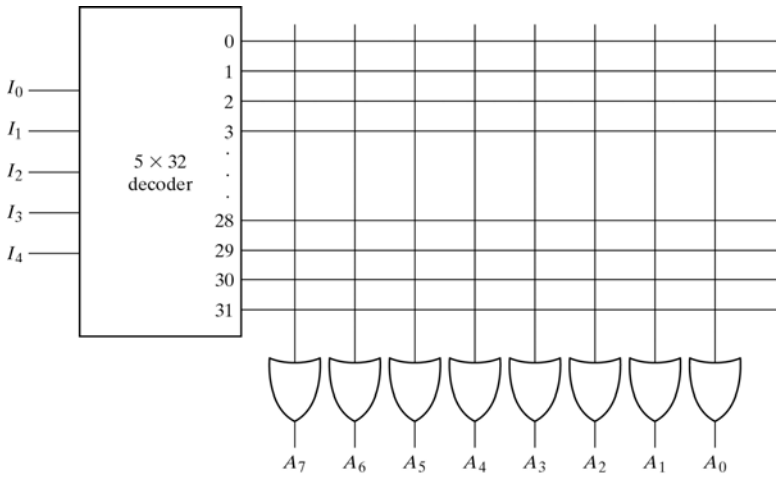


Fig. 7-10 Internal Logic of a 32 × 8 ROM

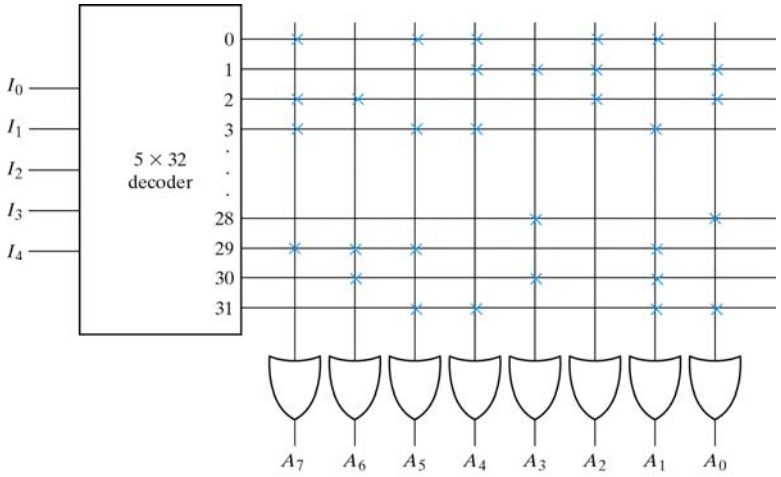
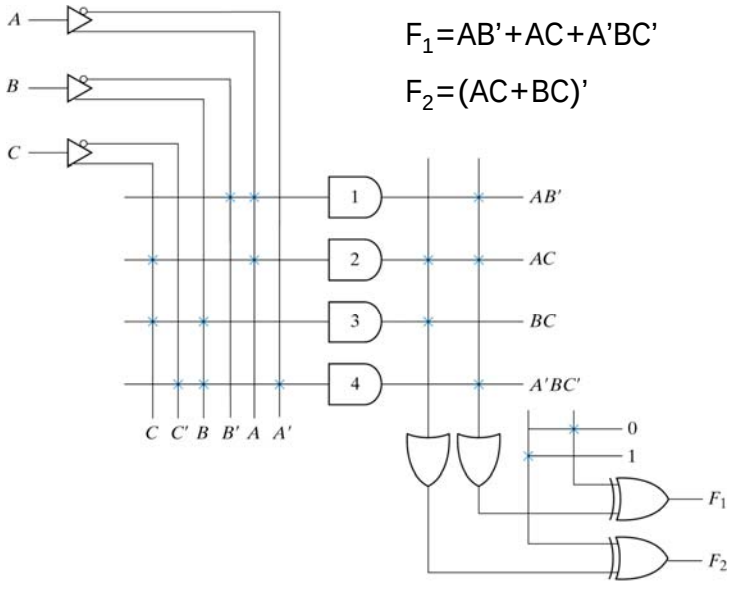


Fig. 7-11 Programming the ROM According to Table 7-3

Programmable Logic Arrays

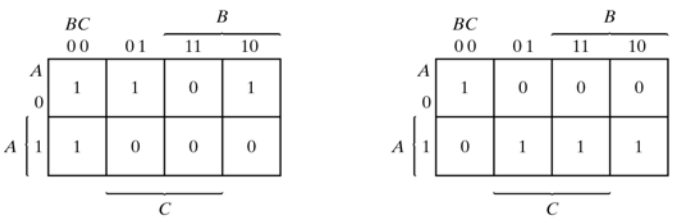
- Similar to PROM except it does not have the full decoding capabilities and does not produce all the minterms.
- The decoder is replaced by an array of AND's that could be programmed to generate any product term of the input var.
- The product terms are connected to OR's to produce the SoP



$F_1 = AB' + AC + A'BC'$

$F_2 = (AC + BC)'$

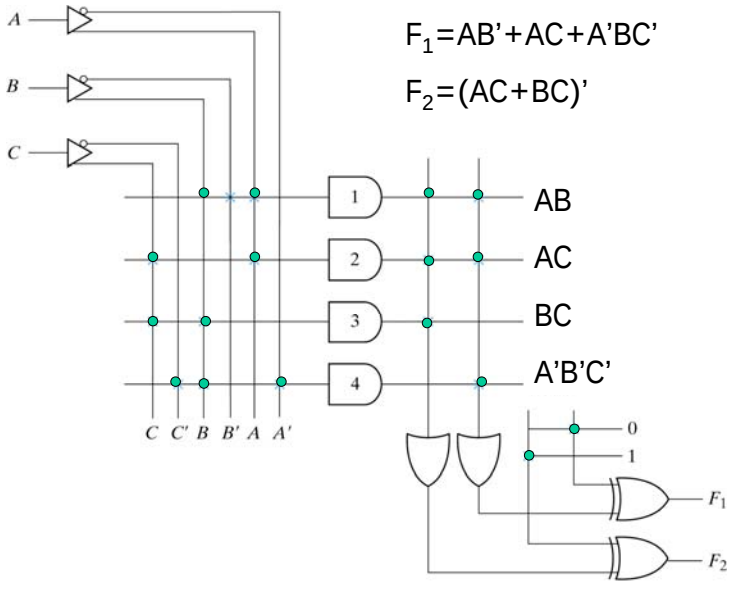
Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs



PLA programming table

	Product term	Inputs A	B	C	Outputs (C) F_1	(T) F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
$A'B'C'$	4	0	0	0	–	1

Fig. 7-15 Solution to Example 7-2



$F_1 = AB' + AC + A'BC'$

$F_2 = (AC + BC)'$

Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

Programmable Array Logic

- Only the AND gates are programmable, easier to program but not as flexible.
- Figures shows 4 inputs (each input has a buffer inverter gate) and 4 outputs.
- Each output is generated by a fixed OR gate.

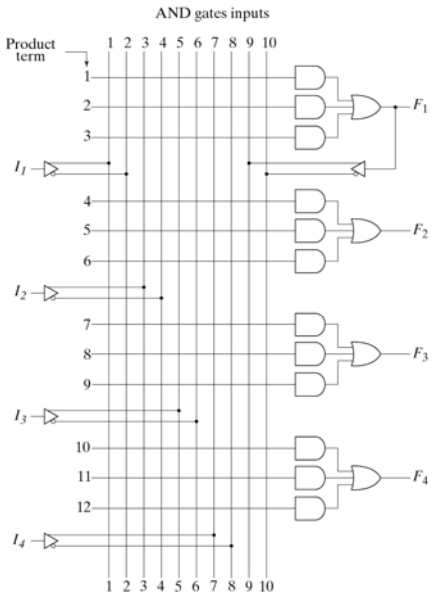


Fig. 7-16 PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

PAL Example

- After minimization

  W=ABC'+A'B'CD'

  X=A+BCD

  Y=A'B+CD+B'D'

  Z=ABC'+A'B'CD'+AC'D'+A'B'C'D

  = W+ AC'D'+A'B'C'D

PAL Example

Product Term	AND inputs					Output
	A	B	C	D	W	
1	1	1	0	-	-	w=ABC'
2	0	0	1	0	-	+A'B'CD'
3	-	-	-	-	-	
4	1	-	-	-	-	x=A
5	-	1	1	1	-	+BCD
6	-	-	-	-	-	
7	0	1	-	-	-	y=A'B
8	-	-	1	1	-	+CD
9	-	0	-	0	-	+B'D'
10	-	-	-	-	1	z=w
11	1	-	0	0	-	+AC'D'
12	0	0	0	1	-	+A'B'C'D

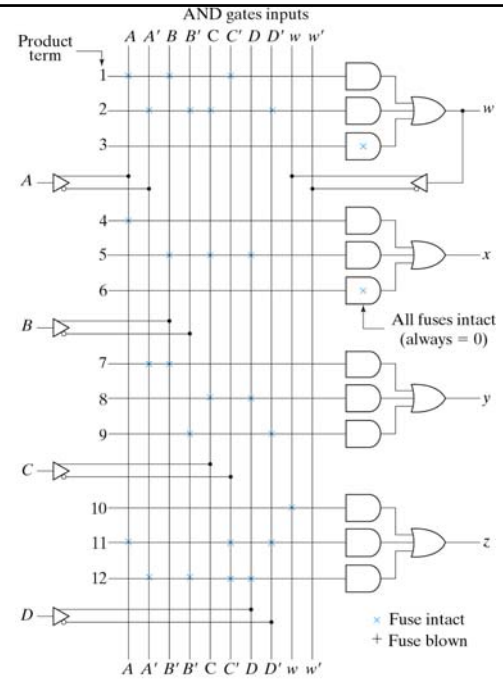


Fig. 7-17 Fuse Map for PAL as Specified in Table 7-6

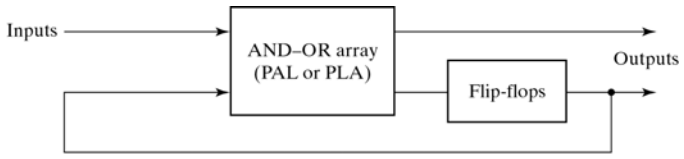


Fig. 7-18 Sequential Programmable Logic Device

A flip-flop is added and the output is fed-back to the array.

Each section of the PAL is called a microcell.

May have other features, use or bypass F-F, selection of clock edge polarity, preset or clear for F-F, or tru/complement output

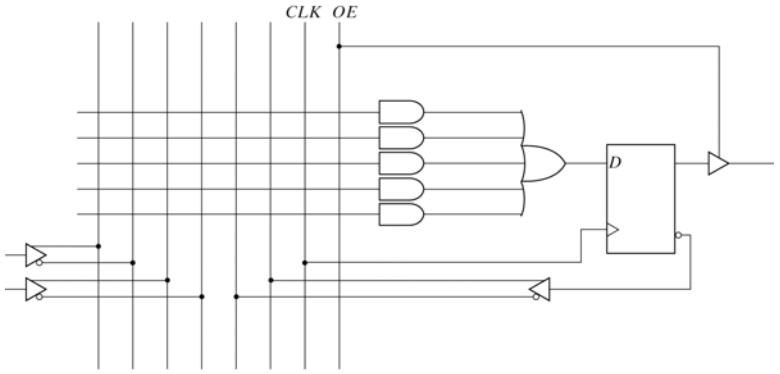


Fig. 7-19 Basic Macrocell Logic

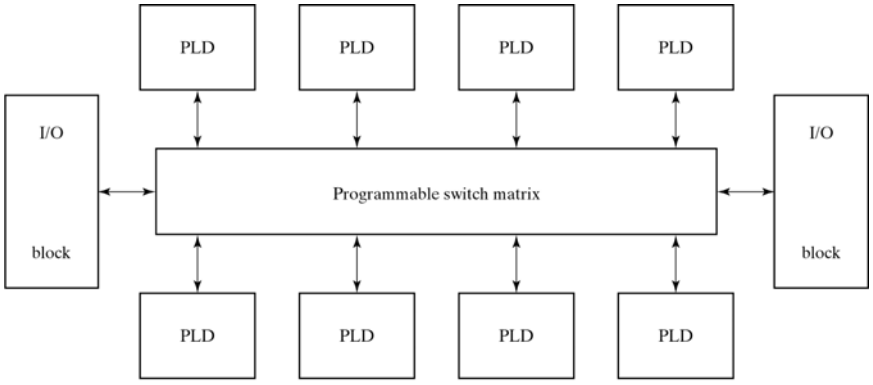


Fig. 7-20 General CPLD Configuration

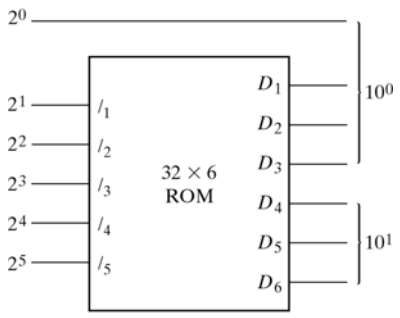


Fig. P7-17