Lecture 20, April 4, 2007

Memory and Programmable Logic Devices

In a digital system such as a digital Computer data is stored in three different places:

- Long term memory such as: hard disk, magnetic tape.
- Short term memory such as Random Access Memory (RAM).
- Registers.

In a Computer, or any digital System there are a few registers used to hold data while it is being processed. For a example, in an addition operation, the numbers to be added are read from two registers, added and the result is written into either a third register or one of the two original registers (if the initial data is not needed any more).

The information to be processed has to be transferred from the long term memory, e.g., hard disk, to the registers before being processed. Doing this for every piece of information is slow. So, it is customary to write the information into a faster short term memory such as a RAM. The information in the RAM, usually a block of data will be transferred to registers a word at a time.

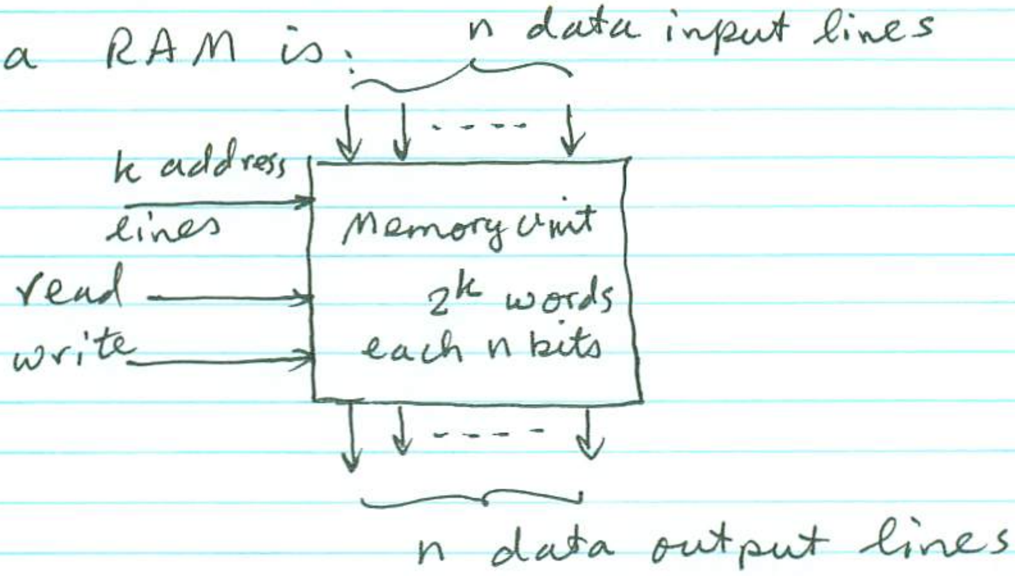
Another type of memory is Read Only Memory (ROM). While, we can write into a RAM and also read from a RAM, in the case of ROM we can only read out the data. The information is written into a ROM a priori. A ROM is, in fact, a special gate of Programmable Logic Device (PLD) where data is programmed into the Device, i.e., written permanently (etched into) it.

Random Access Memory (RAM)

When you deal with a magnetic tape or a hard disk, data is written sequentially and accessed sequentially. So, reading a word from this type of memory does not take the same amount of time. The data written to the beginning of the tape take less time to read than those written into the end of the tape.

In a Random Access Memory: Data is written and read in parallel, i.e., you put the address on the address line and you read from or write into the location addressed. So reading or writing takes the same amount of time for any randomly selected piece of data (a word). That is why we call this a Random Access Memory (RAM).

The block diagram of a memory unit such as a RAM is:



an n-bit memory unit with 2^k entries needs k address bits for addressing these 2^k words. A read signal puts the content of the location specified by the k address bits on the output lines. A on the write signal allows the n bits placed on the input lines be written in the location addressed by the k address bits.

For example a 1024 X 16 Memory, has 1024 words and each word is 16 bits. So, we need 10 address bits and 16 input data and 16 output data lines.

The following figure shows some typical entries in such a memory unit

Address	Content
0000000000	1011011010011100
000000000	1100100111100010
0000000010	0011110101111010
⋮	⋮
1111111101	1001110100001001
1111111110	0101111100011101
1111111111	1010110011101011

Read /Write operation

One way ~~too~~ to work with a RAM is to have a Read control input and a Write Control input. In such a case when we want to write:

1) We apply the address of the location to which we want to write to address lines.

2) Apply the data to be written to the input lines

3) Activate the write input.

To read from memory:

1) Apply the address of the location you wish to read from to address lines

2) Apply a one to the read input and data is ready at the output lines.

Another way, which is more Common is to have a chip select input (also called enable input) and a read/write input.

When the enable input is zero, the chip is disabled. When the enable input is 1 the chip is selected then a 0 (zero) on the Read/Write input allows writing to the addressed location while a one on the Read/Write allows reading from that location.

Memory Enable	Read/Write	Memory Opertation
0	X	None
1	0	Write
1	1	Read

There are two types of RAM:

- Static RAM.
- Dynamic RAM.

A Static RAM consists of latches that store binary data. The Content of memory is valid as long as the chip is powered, i.e., it is erased only when chip is not supplied power.

A Dynamic RAM (DRAM) stores binary data in the form of electric charges on capacitors implemented inside the chip using CMOS transistors. Since the charge of capacitor discharges, DRAM needs to be recharged (refreshed) Continuously every few milliseconds.

Since DRAM uses a single transistor per bit instead of several transistors in SRAM, it is more dense (has more storage Capacity) and also consumes less power. On the other hand

SRAM is easier to use and is faster, i.e., it has shorter read and write cycles.

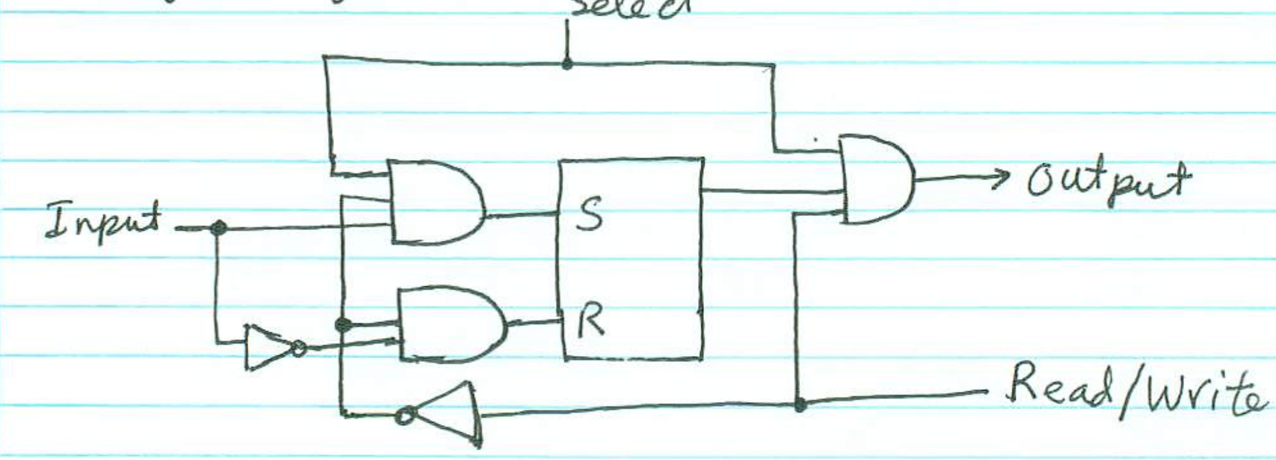
RAM is said to be a <u>volatile</u> type memory since it loses its information when power is turned off. ROM on the other hand keeps its information permanently.

Internal Construction of the RAM

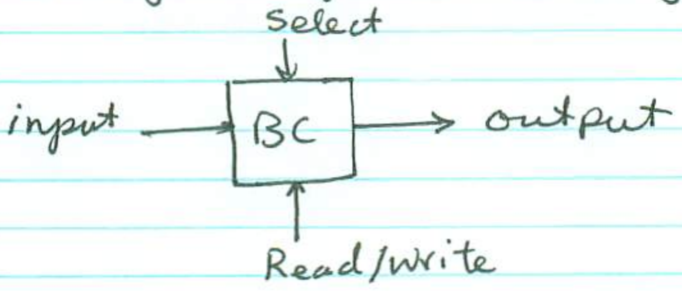
A RAM Consists a 2^k rows of n memory cells. Each memory cell has:

- An input
- An output
- A Select (enable) input
- A Read/Write input

The logic diagram of a memory cell is:



The block diagram for a memory cell is:

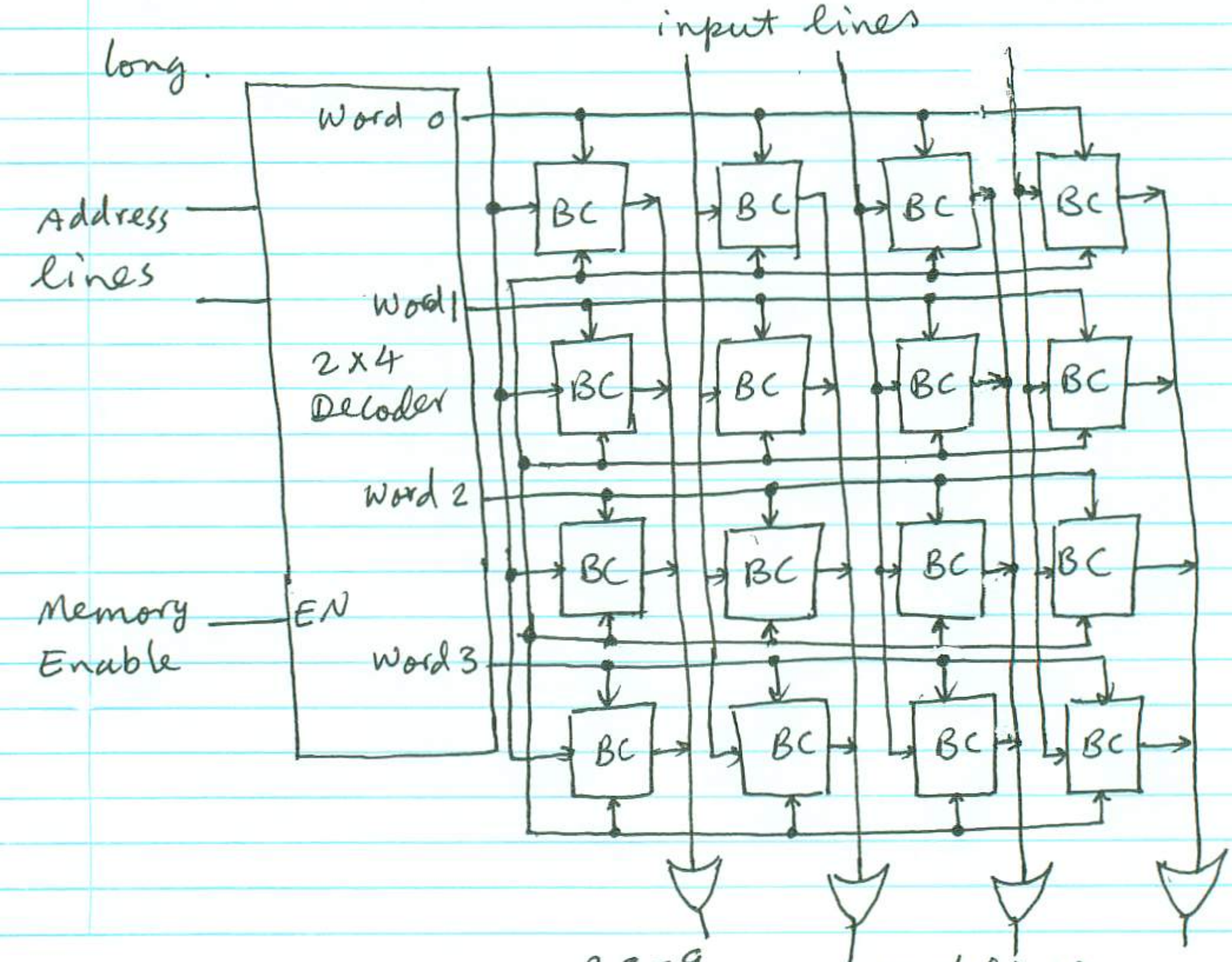


Address Decoding

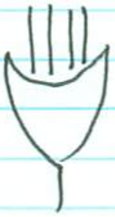
To address one of the 2^k memory locations(rows) we need a $k \times 2^k$ decoder.

For example, a 4x4 RAM needs 2x4 Decoder.

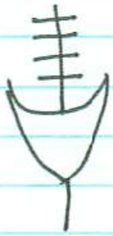
Here $k=2$ and $2^k=4$. Also, each word is 4 bits long.



As you see in the above example, in order to save space a logical gate's input is specified by perpendicular lines instead of parallel lines, i.e., instead of

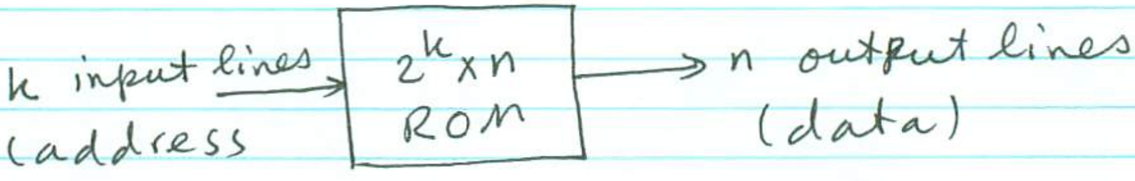


we use

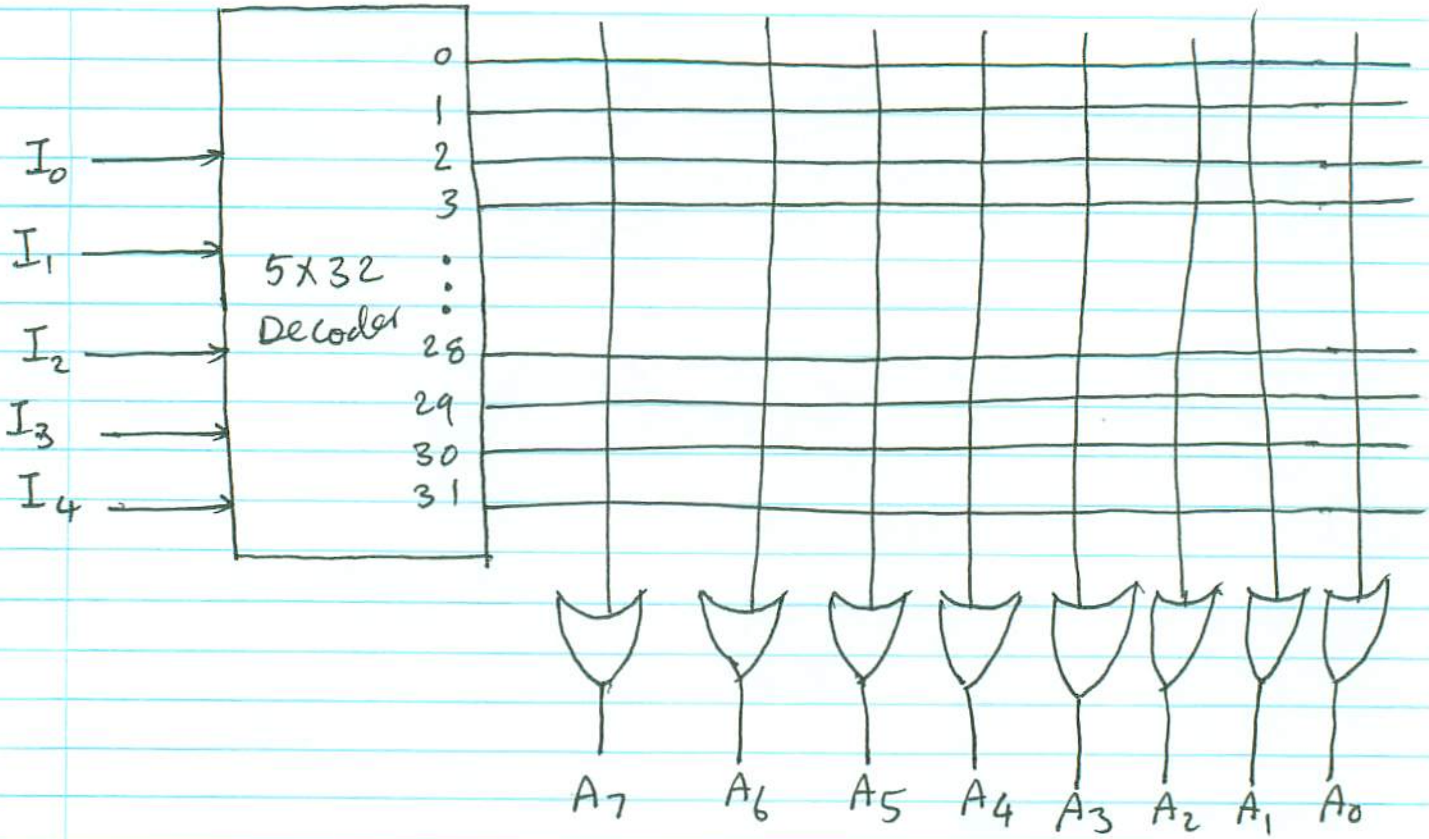


Read Only Memory (ROM)

A block diagram of a $2^k \times n$ ROM is shown below;



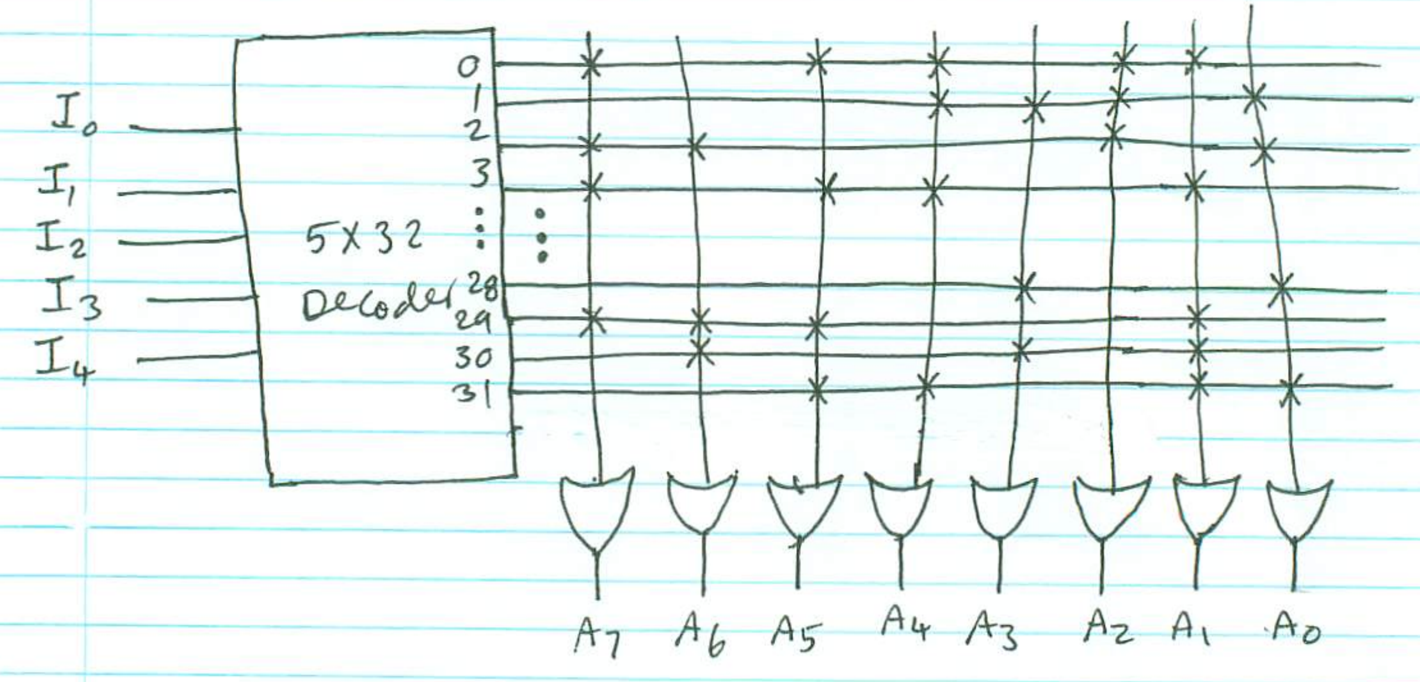
Following diagram shows the internal logic of a 32×8 ROM including the address decoder.



There are $2^k \times n$ intersections, i.e., possible connections between the 2^k outputs of the decoder and the inputs of n OR gates. For example, in the above circuit $2^5 \times 8 = 256$ intersections exist. These intersections are programmable. They can be fused to make connection between two crossing lines or left open. The way we program these intersections, dictatates what is written in the ROM.

For example, if we want to fill up the ROM with the values according to the following truth table, we need to burn (fuse) the ROM as shown below.

Address	Content A7 A6 A5 A4 A3 A2 A1 A0
0 0 0 0 0	1 0 1 1 0 1 1 0
0 0 0 0 1	0 0 0 1 1 1 0 1
0 0 0 1 0	1 1 0 0 0 1 0 1
0 0 0 1 1	1 0 1 1 0 0 1 0
⋮	⋮
1 1 1 0 0	0 0 0 0 1 0 0 1
1 1 1 0 1	1 1 1 0 0 0 1 0
1 1 1 1 0	0 1 0 0 1 0 1 0
1 1 1 1 1	0 0 1 1 0 0 1 1



Combinational circuit implementation using ROM

The fact that the contents of the ROM can be filled with the entries of a truth table, makes the implementation of Combinational logic using ROM possible. For example, in the above 32X8 ROM the output A_7 is

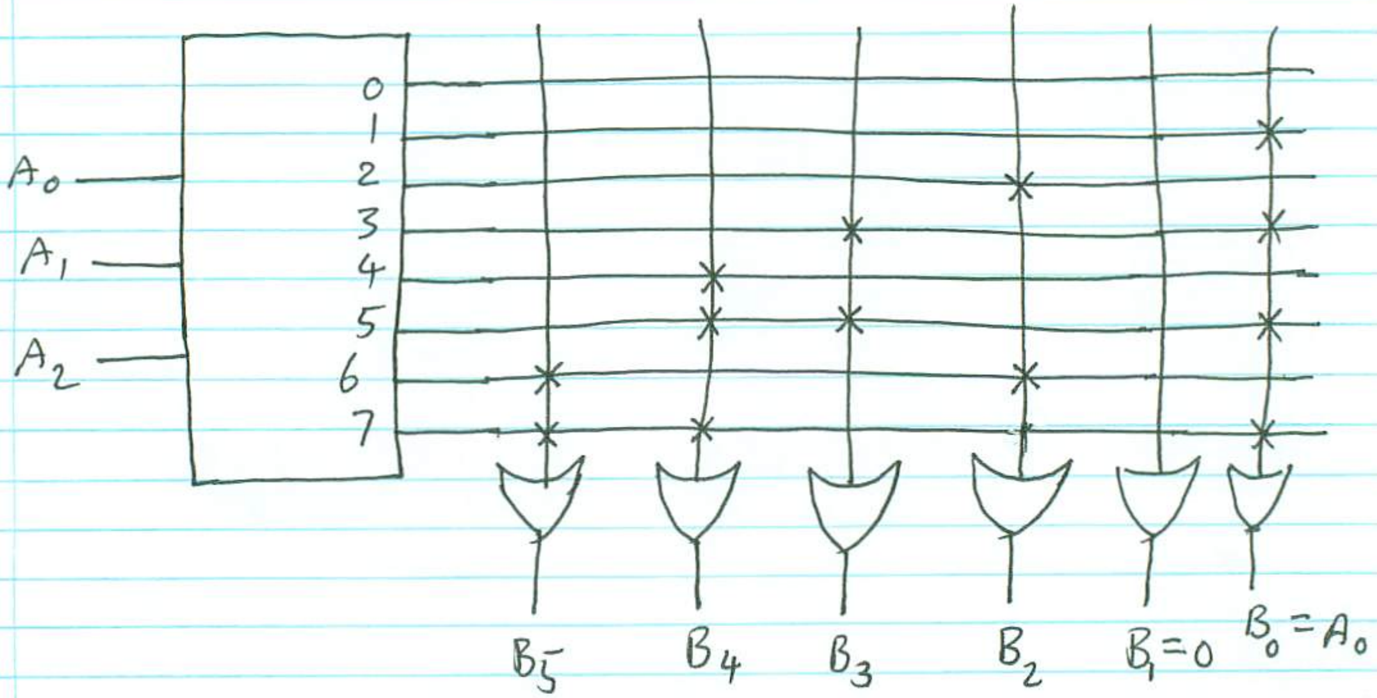
$$A_7(I_4, I_3, I_2, I_1, I_0) = \Sigma(0, 2, 3, \dots, 29)$$

Example: Design a circuit that has a 3-bit input and its output is the square of its input.

The truth table is:

Inputs			outputs						Decimal
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

Since there are 3 input bits we need a 3 x 8 decoder. The circuit diagram is:



Note that, since $B_0 = A_0$ and $B_1 = 0$, we only need to implement 4 outputs.

Types of ROM

- Mask Programmable ROM, or, Simply ROM is programmed at the factory.
- PROM: Programmable ROM can be programmed by the user using a PROM programmer, using high voltage pulses. The process is irreversible, i.e., once the data is written it

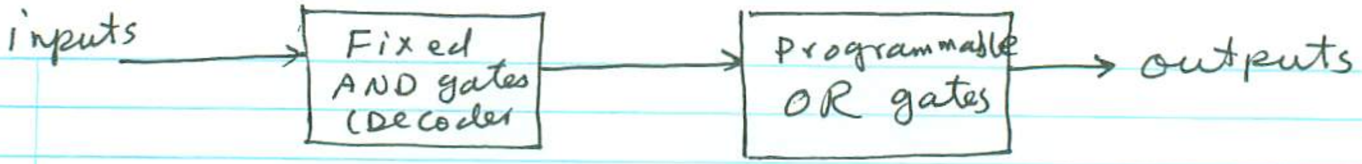
Cannot be erased.

- EPROM: Erasable PROM. An EPROM can be erased using an ultraviolet source and programed again.
- EEPROM (E^2PROM): Electically erasable PROM. This is similar to EPROM, but it can be erased using electric current.
- Flash memory: It is similar to EEPROM, but it has internal circuitry enabling to write and erase in specific locations without needing a special programmer.

Combinational PLDs:

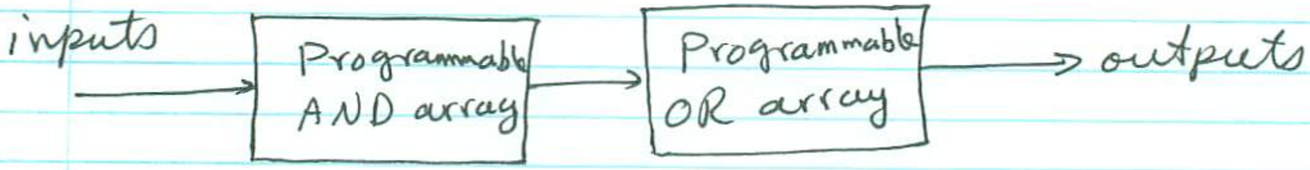
A PROM is an example of a Programmable Logic Device. In general, there are three types of Combinational PLDs. They are

- PROM: In a PROM we have a set of fixed AND gates (forming the address decoder) and a set of programmable OR gates as we saw in previous examples.



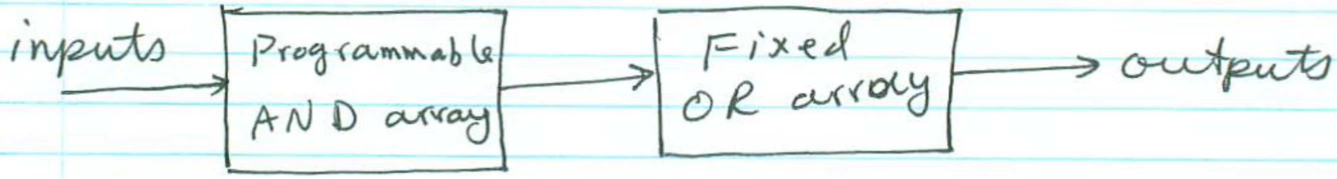
a) A PROM as a PLD

- Programmable Logic Array (PLA):
In a PLA both AND gates and OR gates are programmable



b) PLA

- Programmable Array Logic (PAL)
In a PAL the AND gates are programmable but the OR gates are fixed.

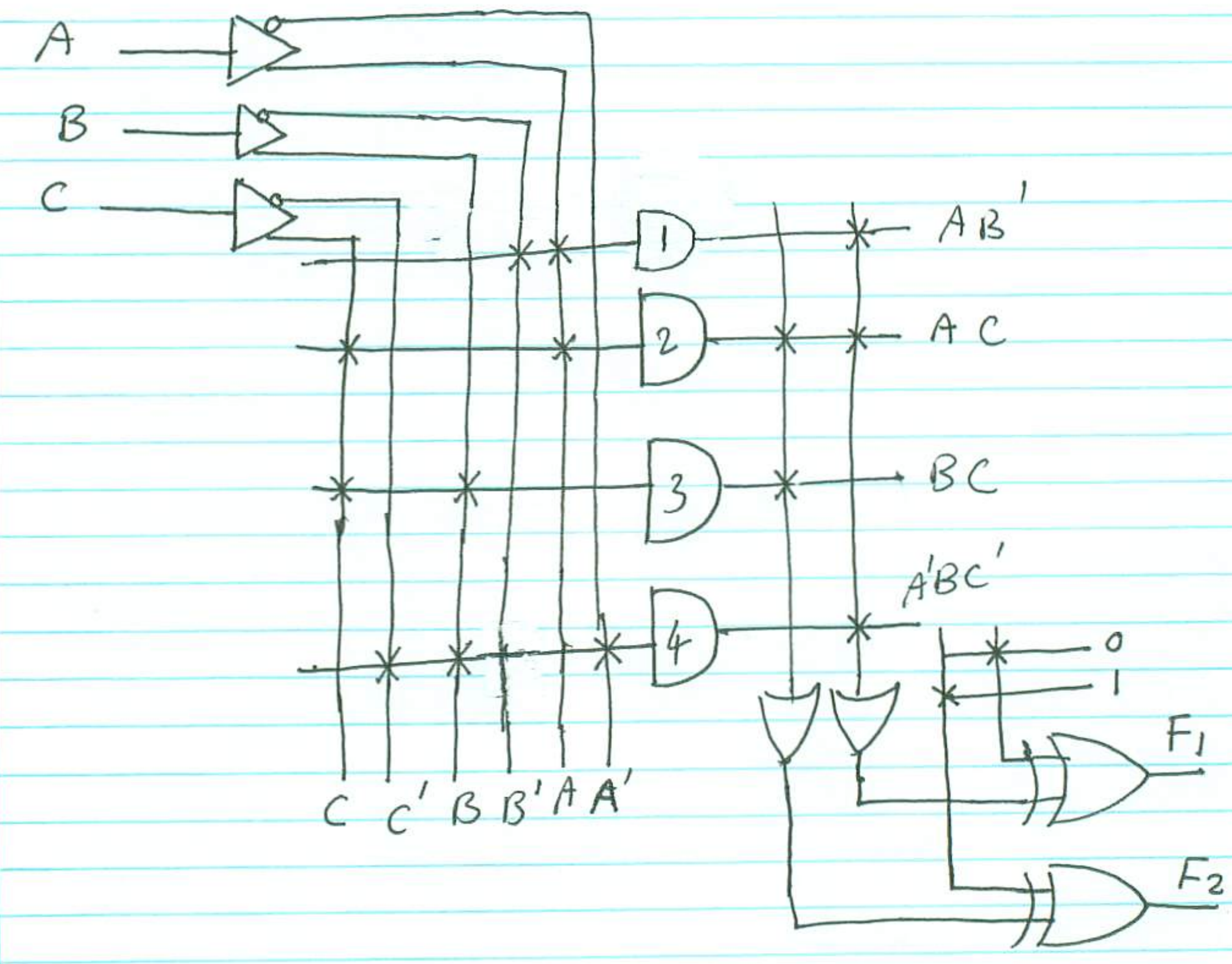


c) PAL

Programmable Logic Array (PLA)

A PLA is similar to PROM. The difference is that in a PLA all the minterms are not generate. Only the terms required for implementing a function are formed.

Following diagram shows a PLA with 3 inputs and two outputs.



Each input goes into a buffer-inverter ( ). The outputs of the buffer-inverters (the inputs or their complement) are connected to AND gates. The output of AND gates are fed to each OR gate. Finally the output of OR gates go to XOR gates with eith 0 or 1 connected to their other input so that either output is inverted or not. The particular connection in the above figure implements:

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

and

$$F_2 = (AC + BC)'$$

The PLA programming table for this example:

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

Example : Design the following functions using PLA :

$$F_1(A,B,C) = \Sigma(0,1,2,4)$$

$$F_2(A,B,C) = \Sigma(0,5,6,7)$$

A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$$F_1 = (AB + AC + BC)'$$

A \ BC	00	01	11	10
0	1	0	0	0
1	0	1	1	1

$$F_2 = AB + AC + A'B'C'$$

The programming table (fuse map) is :

Product term		Inputs			outputs	
		A	B	C	(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

Programmable Array Logic (PAL)

PAL is a programmable device with fixed OR gates and programmable AND gates.

An example of a PAL with 4 outputs is shown on the next page. This PAL has 4 sections each with 3 programmable gates connected to a fixed OR gate. So, each function implemented can have at most 3 terms. If a function has more than 3 terms, we may implement it using two sections.

Assume that we would like to implement

$$w(A,B,C,D) = \Sigma(2,12,13)$$

$$x(A,B,C,D) = \Sigma(7,8,9,10,12,13,14,15)$$

$$y(A,B,C,D) = \Sigma(0,2,3,4,5,6,7,8,10,11,15)$$

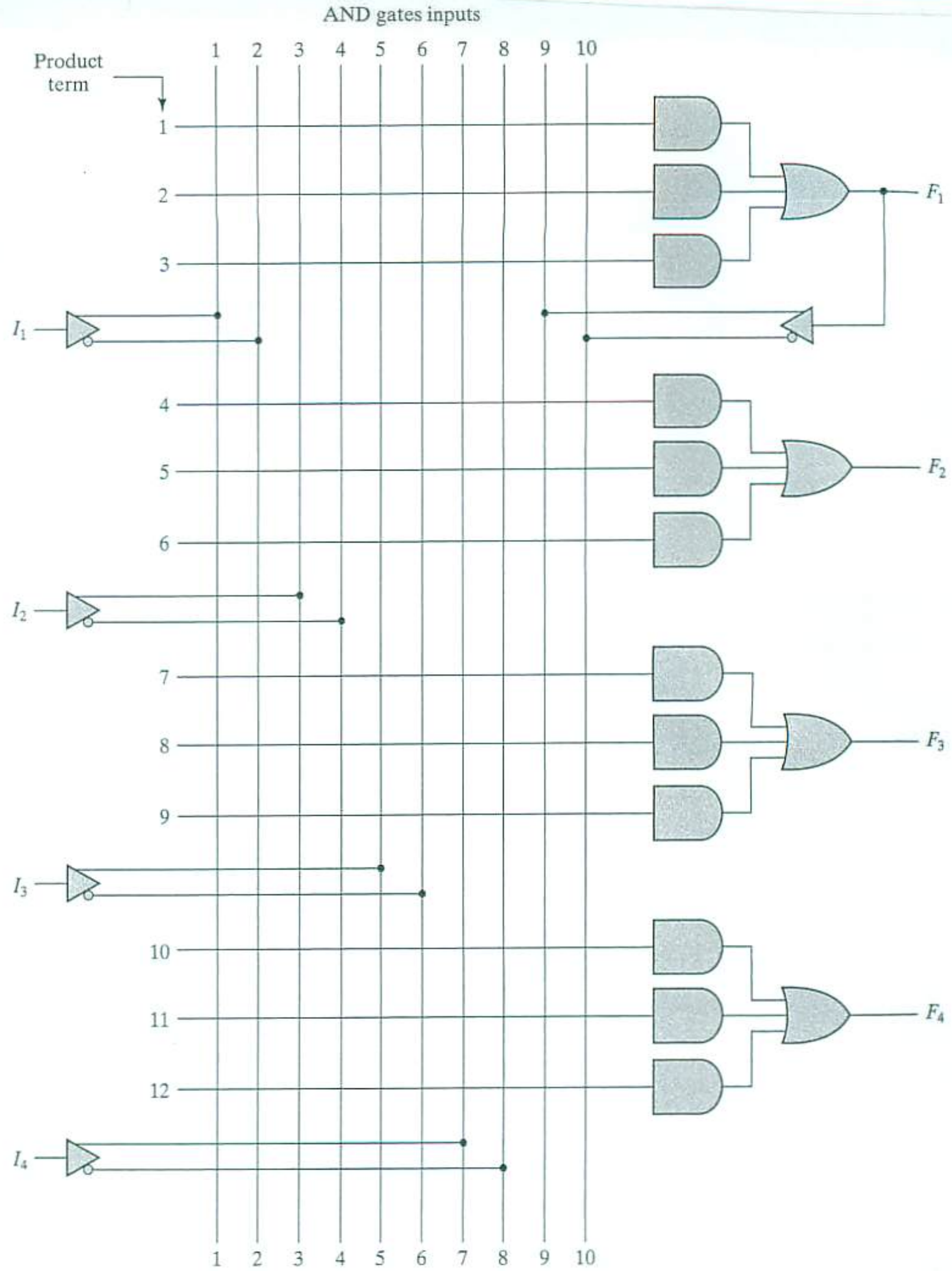
$$w(A,B,C,D) = \Sigma(1,2,8,12,13)$$

Using K-map we find that

$$w = ABC' + A'B'CD'$$

$$x = A + BCD$$

$$y = A'B + CD + B'D'$$



PAL with four inputs, four outputs, and a three-wide AND–OR structure

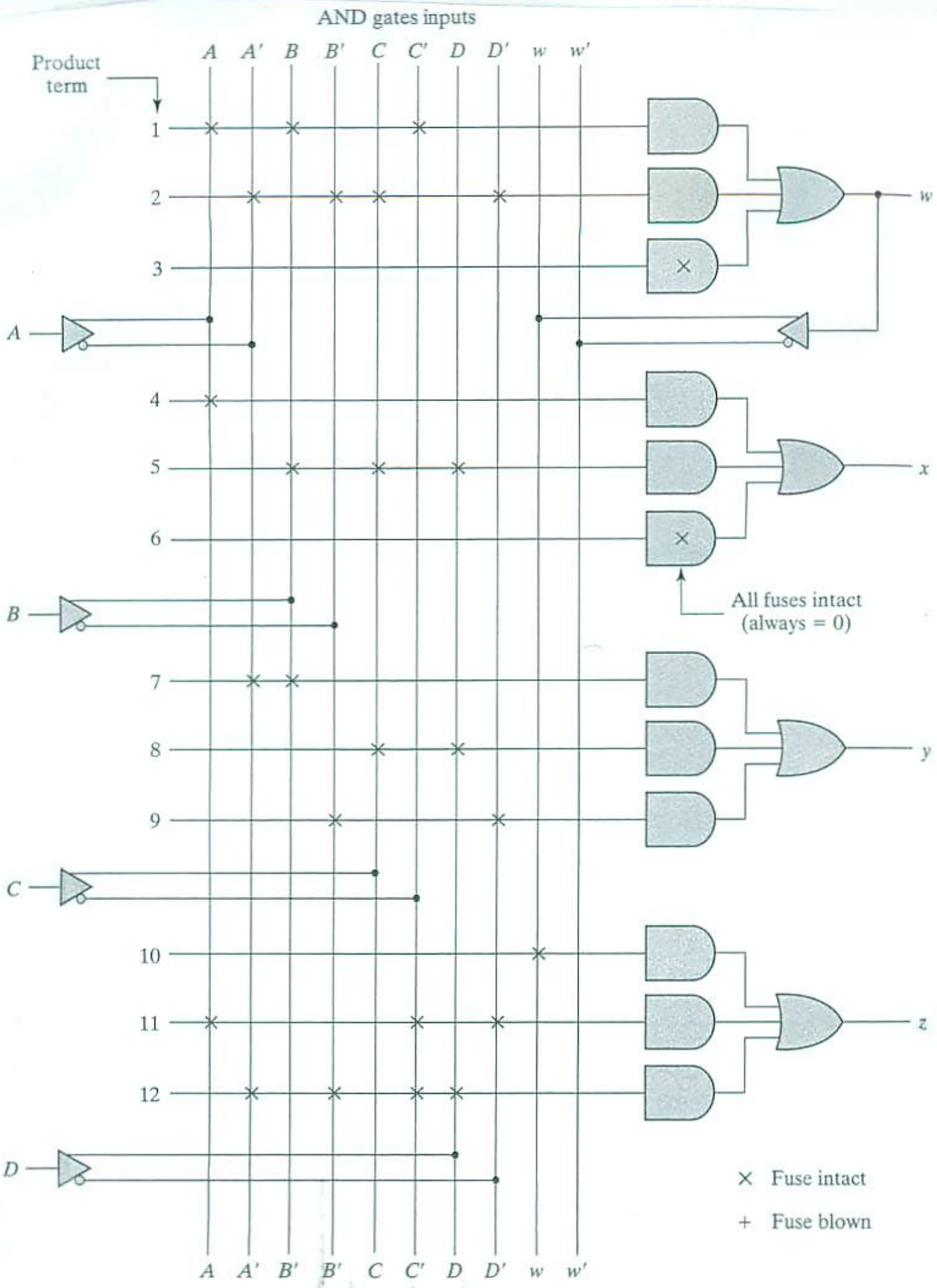
and $z = ABC' + A'B'CD' + AC'D' + A'B'C'D$

z has more than three terms however since $ABC' + A'B'CD' = w$, we have $z = w + AC'D' + A'B'C'D$

Using w as ~~one~~ of the AND inputs we have:

PAL Programming Table

Product Term	***A***	***B***	***C***	***D***	***w***	**Outputs**
1	1	1	0	—	—	$w = ABC' + A'B'CD'$
2	0	0	1	0	—	
3	—	—	—	—	—	
4	1	—	—	—	—	$x = A + BCD$
5	—	1	1	1	—	
6	—	—	—	—	—	
7	0	1	—	—	—	$y = A'B + CD + B'D'$
8	—	—	1	1	—	
9	—	0	—	0	—	
10	—	—	—	—	1	$z = w + AC'D' + A'B'C'D$
11	1	—	0	0	—	
12	0	0	0	1	—	


AND gates inputs
A A' B B' C C' D D' w w'
Product term
1
2
3
A
4
5
6
B
7
8
9
C
10
11
12
D
w
x
y
z
All fuses intact (always = 0)
× Fuse intact
+ Fuse blown
A A' B' B' C C' D D' w w'