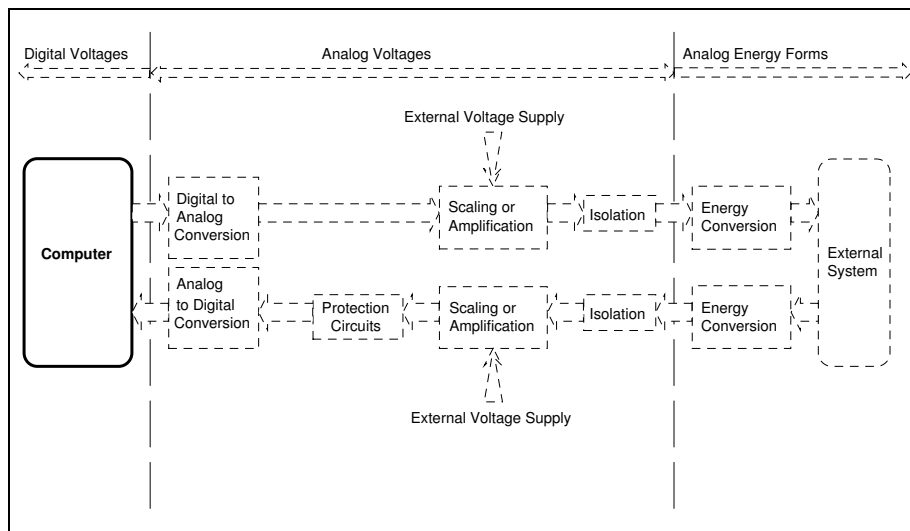


# Chapter 5

## Memory Systems and Programmable Logic

### A Summary...

A short chapter overviewing the types of memory devices available and their applications. An introduction to special purpose memory devices for implementing digital circuits (PROM, EPROM, EEPROM) and Boolean logic (Programmable Array Logic (PAL), Programmable Logic Array (PLA), etc.).



## 5.1 Introduction

In Chapter 4, we examined a number of techniques that could be used to create a reckoning machine, or computer, through simple digital circuits. In that chapter, we were able to instill a crude form of human reasoning into the machine through Boolean logic circuits and to carry out simple manipulation of numbers via numerical circuits designed from digital circuits. Clearly, these circuits are not sufficient to create the sort of computer systems with which we have become familiar. Another important trait of computer systems is their ability to store data on both a short and long term basis. In Chapter 4, we examined the flip-flop and the register (collection of flip-flops) as a mechanism for short-term data storage. In this chapter, we need to delve further into more practical memory storage devices that can be built with a higher density and lower cost.

Traditionally, magnetic (tape, floppy and hard disk) storage media have been used for the long term storage of data. The major reason for this is that these media are considered to be "non-volatile". This means that the contents remain in tact even when power is removed from the circuits responsible for storing and retrieving data from the media. More recently, optical devices such as compact disks have been introduced as an alternative format, again offering a relatively non-volatile storage of data at a very high density. Another reason for using all these types of storage systems is that they have provided relatively low cost data storage formats at times when other alternatives proved considerably more expensive. The problem with all these storage techniques is that they are based upon the movement of mechanical components that scan the surface of the medium and hence are very slow relative to the processing abilities provided by microprocessors and other digital circuits.

For several decades now, short term data storage has been facilitated by semiconductor memory. Digital memory circuits are orders of magnitude faster than any of the mechanically driven, long-term data storage formats described above. However, in the past, they have taken up more physical space than the equivalent mechanical formats (particularly because of IC packaging and pin-out) and have generally suffered from volatility problems - in other words, most semiconductor memory storage devices lose their contents when power is removed.

The last two decades have seen dramatic increases in the density of memory storage devices, thanks largely to the introduction of CMOS based circuits and improvements in semiconductor fabrication technology. The increases in density have been coupled to dramatic decreases in storage costs, to the extent where the costs of semiconductor memory are approaching those of mechanical storage systems. At this point however, the question of volatility has still not been resolved satisfactorily. Although it is possible to purchase memory devices which do not lose their contents when power is removed, the sorts of circuits that have this characteristic, and can be written to and read from, are still relatively costly and inefficient for general computing.

Data storage technologies change so rapidly that it is really inappropriate to dwell upon specific technologies within a text such as this. This chapter really provides an overview of the sorts of semiconductor data storage techniques that are available and their relevance to computing.

The secondary purpose of this chapter is to look at a range of devices that can be used to store Boolean logic functions, rather than simple binary data. The reason for examining these particular circuits (referred to generically as Programmable Logic Devices) in a chapter associated with memory circuits is because of the similarity between the corresponding devices.

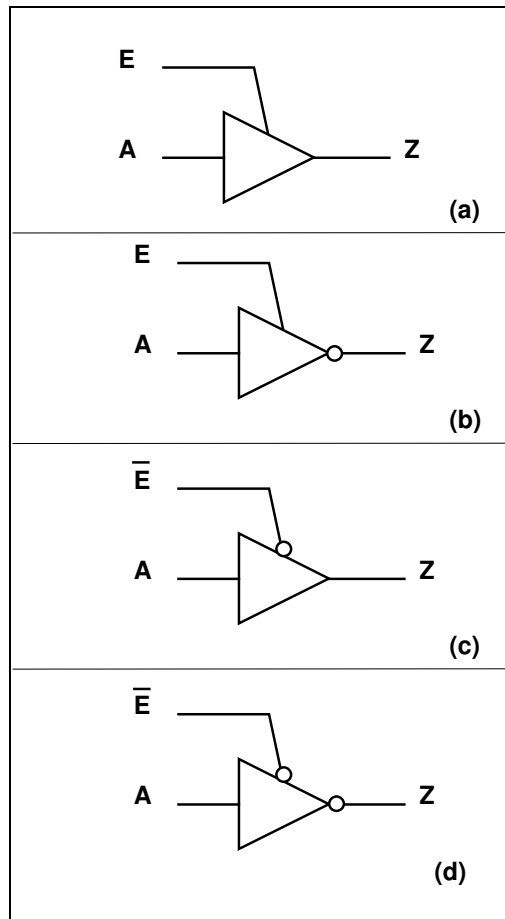
Prior to beginning a discussion on memory circuits, we need to examine a few new Boolean devices that will have significance to us as we progress. The first group of new Boolean devices that we need to examine are referred to as "Tristate Devices". So far, all the other devices that we have examined have only been able to provide two outputs - a voltage equivalent to binary one and a voltage equivalent to binary zero. However, in some digital circuits, such as in memory chips, we need to be able to electronically connect and disconnect (ie: isolate) devices from the remainder of the circuit. This is achieved by a third state, referred to as a high-impedance state or Hi-Z state. The voltage output from a tristate device in a Hi-Z state is mid way between the one and zero logic levels (ie: in the forbidden band where it cannot be misinterpreted as a logical output). A range of tristate devices are shown in Figure 5.1.

The simplest way to understand the operation of tristate circuits is to assume that the circuits perform their normal functions whenever enabled. For example, the circuit in Figure 5.1 (b) is a simple inverter whenever enabled - that is, whenever  $E = 1$ . The truth table for the circuit of Figure 5.1 (b) is shown in Table 5.1.

<i>A</i>	<i>E</i>	<i>Z</i>
0	0	Hi-Z
0	1	1
1	0	Hi-Z
1	1	0

*Table 5.1 - Truth-Table for Tristate Inverter with Active High Enable*

The Hi-Z state means that the impedance between the "A" and "Z" terminals is ideally infinite and so the inputs are isolated from the outputs. The circuit of Figure 5.1 (a) simply has A and Z equivalent whenever the circuit is enabled ( $E = 1$ ) but isolated from one another when disabled ( $E = 0$ ).



**Figure 5.1 - Tristate Logic Devices**  
**(a) Non-Inverting Circuit with Active High Enable**  
**(b) Inverting Circuit with Active High Enable**  
**(c) Non-Inverting Circuit with Active Low Enable**  
**(d) Inverting Circuit with Active Low Enable**

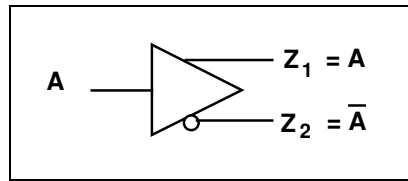
In order to understand the operation of the circuits in Figures 5.1 (c) and 5.1 (d), one needs to understand the "Active Low" terminology. Many digital circuits are enabled whenever a particular terminal is set to a voltage equivalent to binary zero. This inverse logic is represented by placing a bar over the top of the relevant input terminal. For example, in Figures 5.1 (c) and 5.1 (d), the enable input is shown as:

$\bar{E}$

thereby signifying that it should be set low to enable the circuit. An alternative is to place a circle at the input terminal to signify inverted logic. In Figure 5.1 (c) and 5.1 (d), both symbols are used but circuits are often drawn using only one symbol or the other.

Once one understands the significance of the inverted enabling logic, then the operation of the circuits in Figure 5.1 (c) and 5.1 (d) is almost self-evident. The circuit of Figure 5.1 (d) acts as an inverter whenever enabled ( $E = 0$ ) and otherwise isolates terminal "A" from "Z". The circuit of Figure 5.1 (c) has "Z" equivalent to "A" whenever the circuit is enabled ( $E = 0$ ) and "Z" isolated from "A" whenever the circuit is disabled ( $E = 1$ ).

Another simple circuit that needs to be introduced is a modified representation of the inverter gate as shown in Figure 5.2. It is introduced herein so that it can be differentiated from the Tristate logic device.



**Figure 5.2 - Inverter Gate with Both Inverting and Non-Inverting Outputs**

The inverter gate of Figure 5.2, with both inverting and non-inverting outputs looks confusingly similar to the circuit of Figure 5.1 (c) so one has to be careful when examining circuits. Normally, tristate enables are shown with a "bent" line, whereas the two outputs of the inverter are shown with straight lines. However, it is best to look for further written evidence when dealing with circuits that may contain both devices.

## 5.2 Overview of Memory Operation

Many people tend to classify computer memory into two distinct types - Random Access Memory (RAM) and Read Only Memory (ROM). This is rather misleading because all common types of semiconductor memory are, in principle, "random access". In other words, we can normally access any location at any time without first having accessed other locations. Strictly speaking, when people make such a division between memory types, they are really differentiating between Read/Write Memory and Read Only Memory.

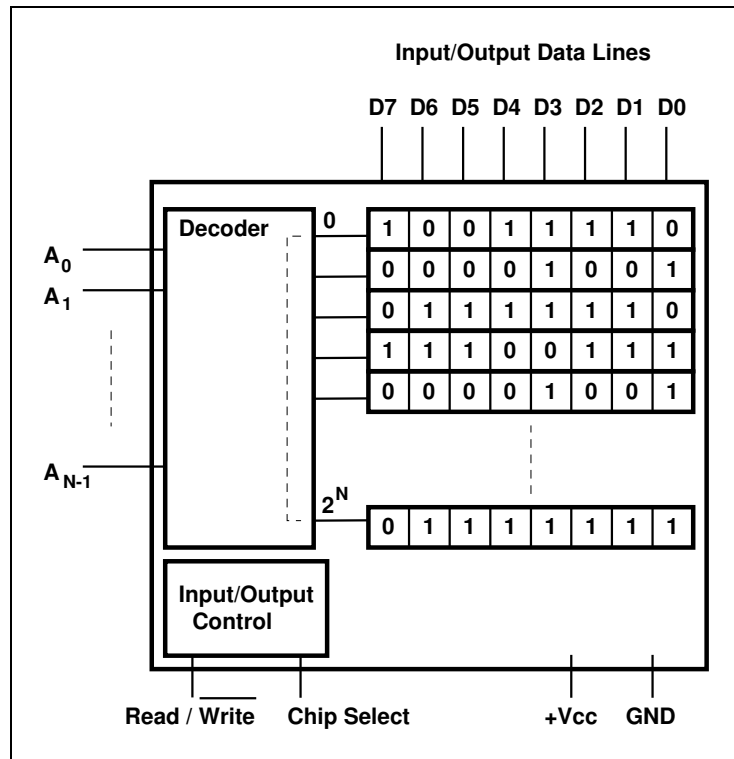
Given that we have to cope with both the common nomenclature for memory and its true functionality, in this section, we will be looking at the following different types of memory devices:

- (i) Static Read/Write Memory (commonly referred to as Static RAM or SRAM)
- (ii) Dynamic Read/Write Memory (commonly referred to as Dynamic DRAM) and Integrated RAM or IRAM
- (iii) Non-Volatile Read-Only Memory (ROM) including:
  - Masked ROM
  - PROM
  - EPROM
  - EEPROM
- (iv) Non-Volatile Read/Write Memory (commonly referred to as NVRAM).

Despite their differences, most of the above memory types have a number of common traits and we shall examine these before discussing the specific attributes of any one type.

Digital memory can be considered in terms of an array of storage locations in which each location or cell can store one bit of data. Schematically, we show the memory in terms of rows of bits. This is shown in Figure 5.3 for a memory chip that has  $2^N$  rows of storage, each of which is 8 bits wide. The width of each row is referred to as the "word length" of the memory chip and varies from device to device. The word length defines the smallest unit that can be written to, or read from, a memory chip. If we assume a hypothetical chip where  $N = 3$  (say) then we have 8 rows of storage (most realistic chips have much more storage than this). Each row has a unique address within the chip. For example, the first row in an 8 row chip would have the address 000 and the last row would have the address 111.

In order to enter data into a particular row (ie: write to that row) or extract data from a particular row (ie: read from that row), we need to address that row by activating the appropriate chip address. This is achieved via the address lines  $A_0 - A_{N-1}$ . A Boolean decoder within the memory chip enables the row of data, corresponding to the given address, to be written to, or read from, the outside world via the data lines D7-D0.



*Figure 5.3 - Schematic of Data Storage in Memory Chips*

All semiconductor memory chips, including non-volatile devices, need to have power applied to them via supply rails in order to function. However, even when power is connected to a memory chip, nothing can be written to or read from the device until the device is enabled or unlocked. Enabling is achieved via a special pin known as a "Chip Select" or "Chip Enable" pin that has to be set to the appropriate logic level in order to allow access to internal chip locations. Chip Enable and Chip Select pins are actually connected to the "enable" lines of tristate logic devices within the memory chip to control access to internal locations.

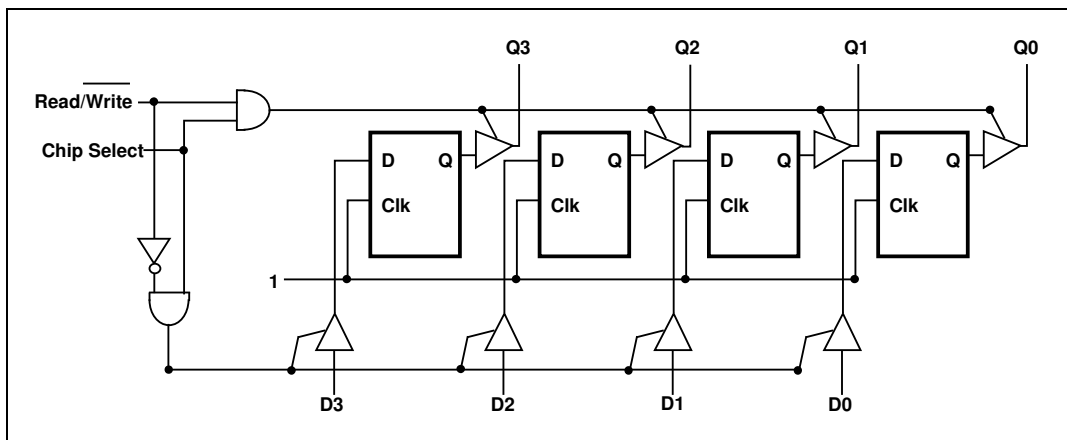
In memory devices designed for both read and write functions, an additional pin is normally provided to control the flow of data to and from internal addresses. The Read/Write pin is also connected to tristate logic devices within the chip that control the flow of data into or out of the chip. In Figure 5.3, for example, the chip can be written to by applying binary zero to the Read/Write pin and the chip can be read from by applying binary one to the same pin.

To summarise then, a number of steps have to be taken in order to access data in a memory chip. These include:

- (i) Supply power to the chip
- (ii) Apply the appropriate bit combination to the address lines of the chip, corresponding to the internal word address that is to be accessed
- (iii) Enable the chip by applying appropriate logic to the Chip Enable or Chip Select pin
- (iv) Place the chip into read or write mode
- (v) Apply a binary combination to the data lines to write to the chip or extract the binary combination from the data lines to read from the chip.

### 5.3 Volatile Read/Write Memory

There are two basic types of volatile read/write memory. These are referred to as static RAM (SRAM) and dynamic RAM (DRAM). Static RAM is perhaps the easier of the two types to understand in terms of common Boolean circuits, since each word (or row of data) is effectively a register, composed of flip-flops. This is shown schematically in Figure 5.4. In this diagram, the sort of logic circuit that could be used to construct a static RAM chip with one (4-bit) row of storage is shown together with the enabling logic that facilitates the "Chip Select" and "Read/Write" functions to occur.



*Figure 5.4 - Designing a One Word Static RAM Chip*

In Figure 5.4, the data inputs to the device correspond to the inputs to the D-flip-flops and the outputs correspond to the Q outputs of each flip-flop. Since the inputs and outputs are never enabled simultaneously by the Read/Write pin, then it is possible to join the two together without a conflict and thereby form a bidirectional data port. Note also that because the memory storage is purely transistor based, it is not possible to access data unless power is supplied to the digital devices within the chips. Moreover, the contents of the flip-flops are lost whenever data is removed.

The flip-flops in static RAM chips are not actually fabricated from NAND gates or NOR gates but directly from transistors. This reduces the number of transistors per flip-flop (ie: storage bit) to approximately four. This takes up a considerable amount of space, even when CMOS devices are used, and hence SRAM chips are generally only used where low memory applications are required.

The major advantage of SRAM chips is their speed of operation. Moreover, because static RAM chips are low density, it is easy to address individual words. For example, the Intel corporation's 2114A device was designed for only 10 address lines, thereby storing 1024 words, each of which is 4 bits in length. Static RAM chips can be connected in parallel so that the effective word length of a chip can be increased. For example, a device which only stores 4 bits can be converted to an 8, 12, 16, 32 or even 64 bit device. As long as the chip select lines and read/write lines are connected in parallel, the data can also be stored and retrieved in parallel.

The problem of low density storage in static RAM chips is a serious one and makes them difficult to use in large quantities in computer systems which require ever increasing amounts of memory space for modern programming applications. The more common form of memory for high volume data storage applications is the dynamic RAM chip or DRAM.

In a dynamic RAM chip, each storage cell is made up of a transistor and a capacitor, rather than a flip-flop. The capacitor emulates the role of the flip-flop by retaining its charge for a period of time after having been "set" or "reset". The overall amount of semiconductor space required to store a bit of data is reduced significantly. However, the charge leakage within the capacitor means that the stored charge or voltage is effectively lost within a few milliseconds. This means that every single cell in a DRAM chip needs to be "refreshed" every few milliseconds in order to retain the data. In terms of modern microprocessor systems, a few milliseconds is a considerable period of time and so the refresh problem is not as drastic as may first appear. However, it does mean that DRAM chips can only be used in conjunction with special support circuits whose role is to constantly refresh each memory location. DRAM, like SRAM, can only be operated when power is applied to the chips and all data is lost when power is removed.

The major advantage of the DRAM chips (high density) is also the source of another major drawback. That is, the problem of addressing each memory location. DRAM chips can store hundreds of kilobytes of data and so, theoretically, they would require a large number of addressing lines on each chip. This presents major problems because the number of pins on a chip (ie: the pin-out) is a major expense in terms of fabrication. The problem is resolved by using a smaller number of data lines, in two phases, to address the same word. In other words, each address is divided into two segments and the address lines are switched (multiplexed) to accept the low order and high order address segments. The low order address segment is called the "row" and the high order address segment is called the "column". This enables a chip with 256K (ie:  $256 \times 2^{10}$ ) addresses to be designed with only 9 address pins (instead of the usual 16), thereby allowing the chip to be implemented in a 16-pin integrated circuit (IC) package.

Refreshing of DRAM chips is carried out by pulsing a special pin on the chip once for each row of data. This sort of functionality is achieved by special DRAM controller chips. However, the refresh cycle takes time and while in process prevents the DRAM chip from carrying out its normal functions. The net result is that the DRAM chip is both slower and more difficult to use than the SRAM chip.

Despite the problems associated with DRAMs, they are still the most prevalent form of semiconductor memory at this point in time because of their high storage density and hence, most modern systems are designed with sophisticated controllers that refresh memory chips at optimum periods during computer cycles.

One of the more innovative solutions to the problem of refreshing memory chips is to combine the refresh circuitry with the chips themselves. This alleviates the designer from the task of designing the special control circuitry required to handle the refresh cycles. The combined devices are given the title "Integrated RAM" or IRAM. Although these would seem to provide the ideal combination of characteristics, the built-in control circuitry takes up a considerable amount of chip space and tends to eliminate many of the gains made by using high density dynamic RAM in the first instance.

## 5.4 Non-Volatile Read Only Memory

The term "Read Only Memory" or ROM is another misnomer, since any storage device that can never be "written to" is clearly of little value. Read Only Memory chips are devices in which it is difficult to store data on a regular basis. In some cases the data is physically burnt into the chips by blowing fuses and in other cases it is built into the chip itself during the semiconductor fabrication process. In all cases, the objective is to build a device whose contents are permanently retained, even when power is removed. However, all chips must have power applied to them before data can be accessed.

There are many reasons why some memory chips need to be designed as "Read" devices rather than as Read/Write devices. Quite often, it is necessary to build special purpose computer controllers that always work on a fixed program. In small-scale systems, the cost of the controller can be reduced by eliminating disk-drives and other unnecessary devices. This means that programs need to be stored entirely in non-volatile memory devices that retain their contents even when power is removed. ROM chips are ideal for these types of applications. Another common use for ROM chips is to prevent users from tampering with particular areas in a computer system. Most modern computers have a number of basic functions burnt into ROM so that users cannot interfere with their functionality and create complex system faults. Finally, ROM chips can also be thought of as small disk-drives or cartridges that can be used to store programs and that can be removed and replaced with other chips to change the operation of a particular system.

In order to have any purpose, all ROM chips need to be written to at least once. Some ROM chips can be written to a number of times. A device which can be written to only once is referred to, in computer jargon, as a "WORM" (Write Once Read Many) device. The "Masked ROM" is effectively a WORM chip. It is notable for its low cost in high volume applications and it is effectively "written" to by the semiconductor manufacturer. The "masked" notation comes from the fact that data is embedded into the chip by growing layers of silicon, silicon-dioxide and metal in areas of semiconductor selectively masked to create the required pattern of zeros and ones (ie: the data). The Masked ROM is really aimed at the mass production market (eg: video games, etc.) because of the set-up costs for manufacturing what are effectively special-purpose semiconductor devices.

The Programmable Read Only Memory (PROM) chip is another WORM device. However, it can be programmed by low-cost hardware connected to a small computer work-station. A PROM is composed of a number of microscopic fuses (commonly known as "fusible links") and there is one fuse per bit of stored data. A logical one or zero is represented by the ability of a region to conduct or not conduct data.

Fuses within a PROM chip are blown by addressing a row in a chip and applying "programming voltages" to the data lines of the chip (corresponding to the data to be stored). The programming voltages are considerably larger than the normal binary voltage levels and are sufficient to blow fuses in the required pattern. After a PROM chip has been burnt, its contents can be accessed just like any other memory chip. The disadvantage of PROMs is that once they have been burnt, the contents of the chip can never be changed, so they are not well suited to prototyping.

A commonly used prototyping chip is the Erasable PROM or EPROM. In an EPROM, a one or zero is represented by the presence or absence of charge in an electrically isolated region called a cell. Like the PROM chip, the EPROM is programmed by the application of "programming voltages" to selected addresses within the chip. The programming voltages trap electric charge within cells, thereby forming a non-volatile storage method.

The advantage of EPROM devices is that they can be erased by applying ultra-violet light to the semiconductor material. In fact, EPROM chips are notable for the glass window that exposes the semiconductor material through the packaging of the chip. Natural sunlight and most indoor lights emit a component of ultra-violet and hence EPROMs need to be shielded in order to prevent accidental erasure of data. This shielding can be realised through the application of opaque tapes across the erasure window. However, even without the shielding, it normally takes several years exposure to fluorescent lighting to erase an EPROM, so special-purpose lights need to be purchased for quick erasure when reprogramming.

## 5.5 Non Volatile Read/Write Memory

In Section 5.4, we examined a few of the basic non-volatile Read Only Memory systems. However, the so-called Electrically Erasable PROM or EEPROM or E<sup>2</sup>PROM more closely resembles read/write memory than it does read only memory. It is effectively a non-volatile read/write memory and is sometimes referred to as "read mostly memory".

EEPROM devices can be electrically written to and read from but with some restrictions. Although reading times are similar to most other memory chips, write times are in the order of milliseconds, thereby (currently) making them impractically slow for general purpose computing. Moreover, EEPROMs can only be written to for a limited number of times before their reliability becomes unacceptable as a result of charge retention problems within each storage cell. The most common application for EEPROM is to store system configuration data in personal computer systems and small-scale controllers.

There are difficulties involved in improving the response time of EEPROM chips and so some hybrid systems have been developed in recent years. One such hybrid is the so-called "Non-Volatile RAM" or NVRAM, where a static RAM chip is combined with an EEPROM chip. The RAM chip operates as normal, but every few milliseconds mirrors its data onto a parallel EEPROM chip in case power is removed from the system.

Ideally, of course, the objective is to have non-volatile RAM chips that can perform both read and write functions at normal operating speeds and can be produced at a relatively low cost.

## 5.6 Programmable Logic Devices

Memory devices are good at storing binary data, but there are also good reasons why we need a range of other devices that can be used to store Boolean logic - in other words to replace hard-wired Boolean gates. If we can store relatively complex Boolean logic into individual chips, then we can eliminate unreliable wiring, protect our logic designs and minimise power consumption. We can make circuits with fewer parts (again increasing reliability) and streamline the prototyping of circuits. The devices which help us to accomplish these functions have the generic title of "Programmable Logic Devices" or PLDs.

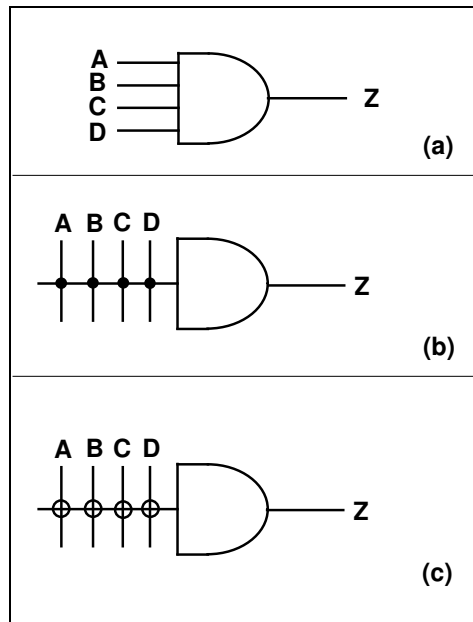
PLDs are included in this chapter because they are really special-purpose memory devices that are used to store Boolean logic and also because their structure is not dissimilar to that of the traditional PROM chip - particularly because it relies upon the "blowing of fuses" in order to ascribe a particular logic.

There are many different types of programmable logic devices and these include the following:

- Programmable Array Logic (PAL)
- Programmable Logic Array (PLA)
- Gate Array.

The gate array is a chip in which logic is partially embedded by the semiconductor manufacturer, then tailored by an end-user and subsequently completed by the semiconductor manufacturer. Such devices are clearly designed for high-volume applications and so in this chapter we will look at the PAL and PLA devices, which can be tailored in low volumes by developers. As we shall see later, the basic concepts of PAL and PLA are similar, but for practical reasons, the PAL device is the more prolific of the two.

In order to understand the functionality of the PAL and PLA devices and the difference between them, one needs to understand the special Boolean symbols used to represent elements in these programmable devices. The special symbols are used in place of traditional logic symbols because they enable structures to be drawn in a straightforward manner. The conventional and PLD representations for an AND gate are shown in Figure 5.5. All three diagrams in Figure 5.5 represent the same AND function in different ways. The representation of Figure 5.5 (c) shows programmable points in the circuit, which are actually created by small fuses that can be blown by applying a programming voltage. As a general rule, if the input variables are drawn with either a "•" (representing a fixed connection) or a " $\oplus$  or X" (representing an intact fuse) then there is a connection between the input variable and the gate - otherwise there is no connection (representing a blown-fuse).



**Figure 5.5 - Representations for AND gates**

**(a) Traditional Representation**

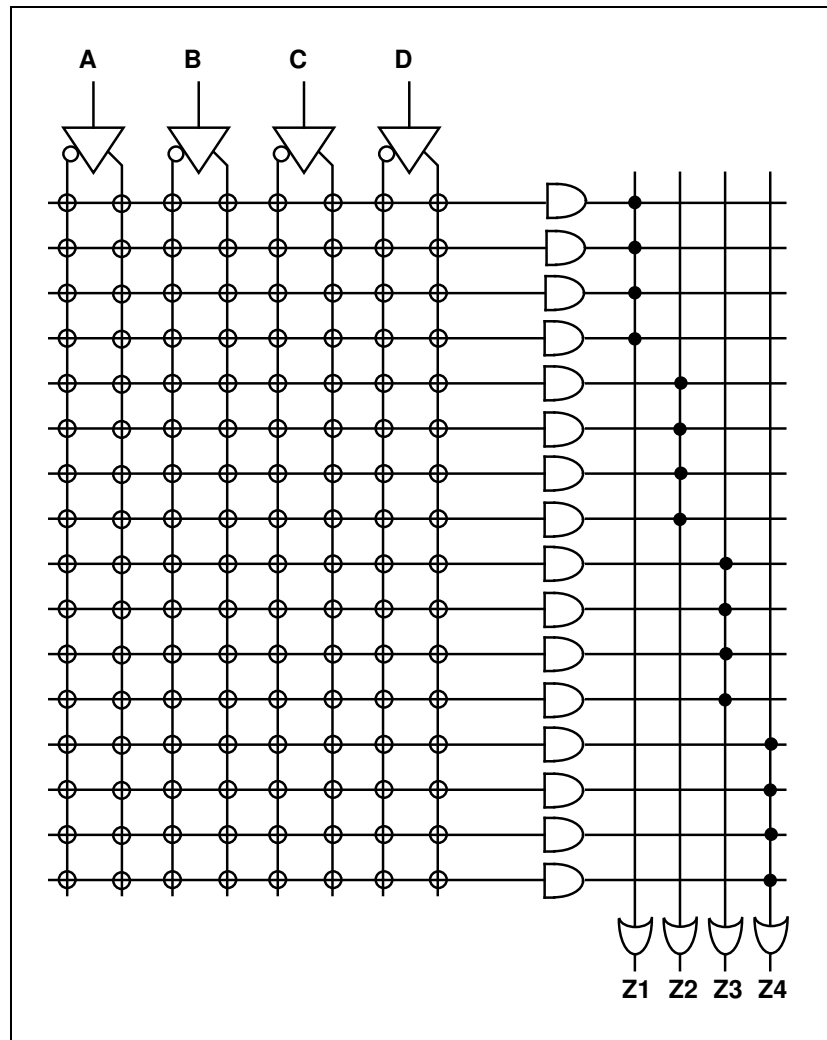
**(b) PLD Equivalent Representation with Fixed Connections (•)**

**(c) PLD Equivalent Representation with Programmable Connections (⊕)**

If one can come to terms with the representation of Figure 5.5, then it is not difficult to understand the functionality of either the PAL device or the PLA device which are shown in Figures 5.6 and 5.7 respectively.

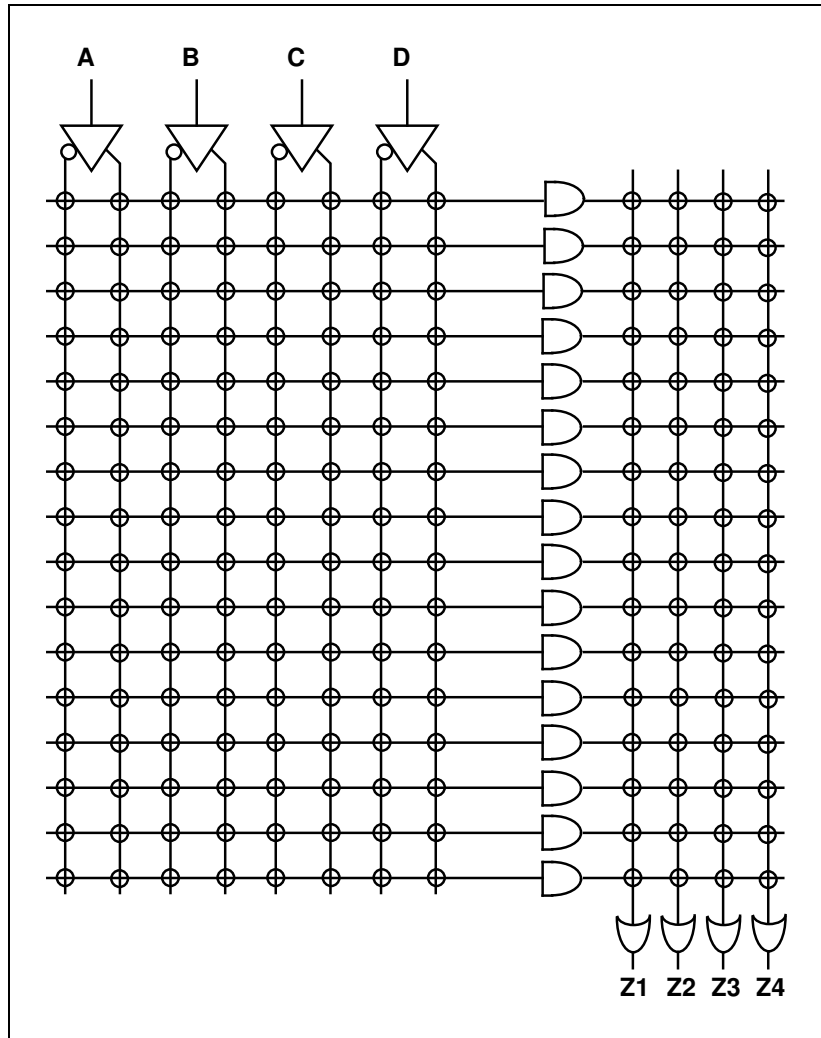
The PAL device is composed of a programmable array of AND gates and a fixed array of OR gates. The size of the array and the number of inputs and outputs depends upon the specific device. In Figure 5.6, the hypothetical device is composed of four inputs and four outputs. By selectively blowing fuses in the AND array, we can create a required logic in the form of a sum of products expression.

The PLA device is composed of a programmable array of AND gates and a programmable array of OR gates, thereby offering maximum programming flexibility. In the PLA device, fuses can be blown on the AND array and on the OR array in order to achieve a required logic function.



*Figure 5.6 - Programmable Array Logic (PAL) Structure*

The PLA device offers more flexibility than is generally required for most applications and so is used less frequently than the PAL. However, the common feature of both devices is that they enable a range of logic functions to be implemented on a single chip. In both cases, the logic has to be converted into an AND/OR form before implementation. Another point of interest is that the PAL and PLA devices are similar in concept to the PROM device, which is actually composed of a fixed AND array and a programmable OR array. This structural similarity between programmable logic storage devices and programmable data storage devices is another reason for combining them in this chapter.



*Figure 5.7 - Programmable Logic Array (PLA) Structure*

PAL and PLA devices are programmed in much the same way as a PROM devices, with a low cost PAL programmer that is connected to a personal computer workstation. Developers normally purchase special software that enables them to generate the required logic in PAL or PLA and then simulate its operation. The required logic is then burnt into the PAL/PLA by applying suitably high voltage levels to appropriate pins on the device. The programming process involves fuse-blowing and is therefore irreversible, so the circuit simulation software used in design of PAL/PLA is important if wastage is to be minimised.

Consider the application of PAL, in the following exercise in order to cement your understanding of the PAL design process.

**Design Problem 6:**

*In Design Problem 1 (in Chapter 4) you were asked to determine the logic required to control an incubation chamber. Using the Sum of Products expressions initially derived from the truth table in that problem, implement the logic using the hypothetical PAL device of Figure 5.6.*

**Solution to Design Problem 6:**

*From Design Problem 1, we know that the original Sum of Products expressions (unsimplified) are as follows:*

$$H = \overline{T2} \cdot \overline{T1} \cdot \overline{T0} + \overline{T2} \cdot \overline{T1} \cdot T0 + \overline{T2} \cdot T1 \cdot \overline{T0}$$

$$F = T2 \cdot \overline{T1} \cdot \overline{T0} + T2 \cdot \overline{T1} \cdot T0 + T2 \cdot T1 \cdot \overline{T0} + T2 \cdot T1 \cdot T0$$

*Although these were greatly simplified by Karnaugh Mapping (in Design Problem 4), we will take these raw expressions and apply them to the PAL to show how such logic can be implemented. The result is shown in Figure 5.8.*

*Note how even the raw Sum of Products expression originally determined in Design Problem 1 can still be transferred to a single PAL chip. Although one would normally use techniques such as Karnaugh Mapping to simplify expressions before committing them to a PAL implementation, this problem demonstrates that as long as the logic fits into a given PAL chip, the inefficiency of the expressions does not affect the cost of implementation. However, simplification of Boolean expressions is important because it minimises the complexity of the implementation and hence the probability of generating an erroneous design. Moreover, it is important, as a general engineering rule, to always begin with the best possible design approach, rather than allowing technology to cover up inefficiencies that should be removed in the first instance by the designer.*

*Realistic PAL chips can support 16 or more inputs and so it is possible to implement relatively sophisticated Boolean logic on a single device. Moreover, because PAL circuit implementations are produced on only two levels of logic, propagation delays in signals are minimised.*

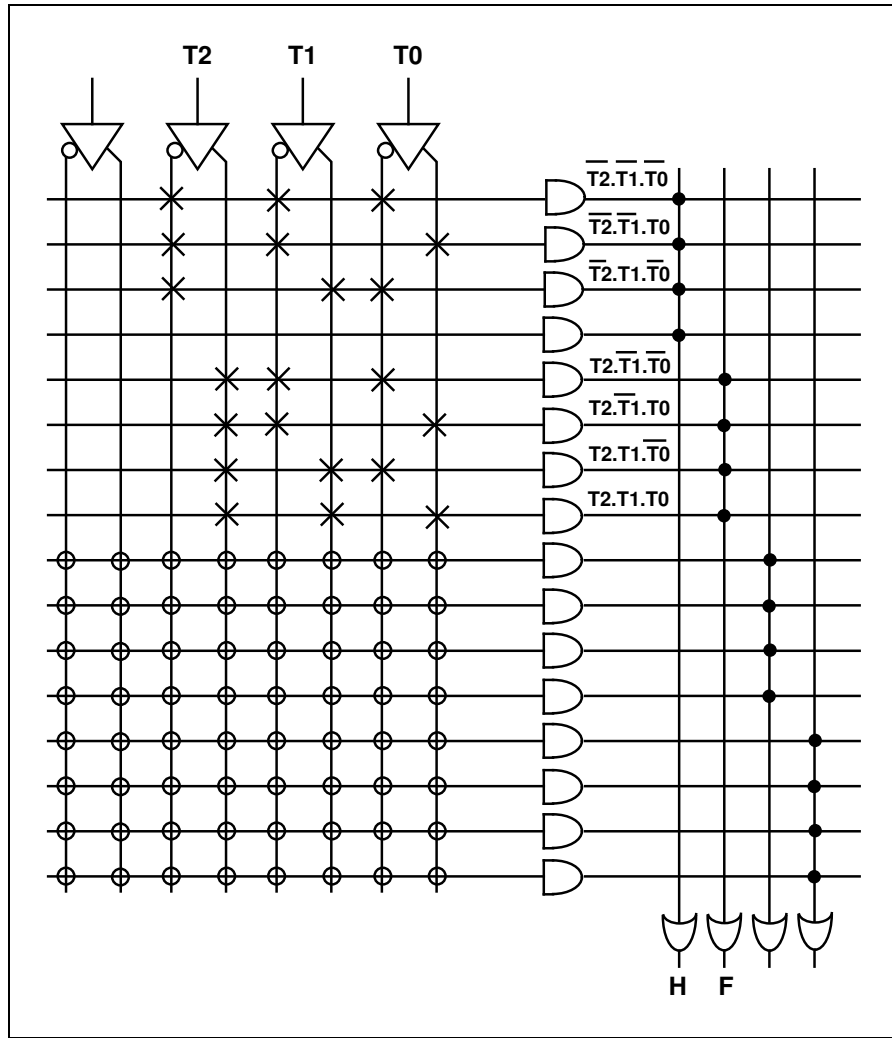


Figure 5.8 - PAL Solution to Incubator Design Problem 1 from Chapter 4