**OLUWATOSIN AYENI**

**15/ENG02/010**

**DIGITAL SYSTEMS FAMILY TREE**

A digital system family tree showing most of the hard ware choices that are currently available can be useful in sorting out the many categories of digital devices. The graphical representation in the figure does not show all the details, some of the more complex device types have many additional subcategories, and older obsolete device types have been omitted for clarity The major digital system categories include standard logic, application specific integrated circuits (ASICs) and microprocessor digital signal processing (DSP) device.

first category of standard logic devices refers to the basic functional digital components (gates, flip-flops, decoders, multiplexers, registers, counters, etc.) that are available as SSI and MSI chips. These devices have been used for many years (some more than 30 years) to design complex digital systems.

Very few new designs apply TTL logic, but many, many digital systems still contain TTL devices. CMOS is the most popular standard logic device family today, primarily due to its low power consumption. ECL technology, of course, is ap plied for higher-speed designs. Standard logic devices are still available to the digital designer, but if the application is very complex, a lot of SSI/MSI chips will be needed. That solution is not very attractive for our design needs today.

The microprocessor/digital signal processing (DSP) category is a much different approach to digital system design. These devices actually contain the various types of functional blocks that have been discussed throughout this text. With microcomputer/DSP systems, devices can be controlled electronic.

Four subcategories of ASIC devices are available to create digital systems:

programmable logic devices, gate arrays, standard-cell, and full-custom.

Programmable logic devices (PLD), sometimes referred to as field programmable logic devices (CPLDs), can be custom-configured to create any desired digital circuit, from simple logic gates to complex digital systems.

Many examples of PLD designs have been given in earlier chapters. This ASIC choice for the designer is very different from the other three subcategories

With a relatively small capital investment, any company can purchase the necessary development software and hardware to program PLDs for their digital designs. On the other hand, to obtain a gate array, standard-cell or full-custom ASIC requires that most companies contract with an IC foundry to fabricate the desired IC chip. This option can be extremely expensive and usually re quires that your company purchase a large volume of parts to be cost effective.

Standard-cell ASICs use predefined logic function building blocks called cells to create the desired digital system. The IC layout of each cell has been designed previously, and a library of available cells is stored in a computer database. The needed cells are laid out for the desired application, and the interconnections between the cells are determined. Design costs for standard-cells ASICs are even higher than for MPGAs because all IC fabrication masks that define the components and interconnections must be custom designed. Greater lead time is also needed for the creation of the additional masks. Standard cells do have a significant advantage over gate arrays.

**FUNDAMENTALS OF PLD CIRCUITRY**

A simple PLD device is shown in Figure 13-3. Each of the four OR gates can produce an output that is a function of the two input variables, A and B. Each output function is programmed with the fuses located between the AND gates and each of the OR gates.

Each of the inputs A and B feed both a noninverting buffer and an in verting buffer to produce the true and inverted forms of each variable. These are the input lines to the AND gate array. Each AND gate is connected to two different input lines to generate a unique product of the input variables. The AND outputs are called the product lines.

**PLD ARCHITECTURES**

The concept of PLDs has led to many different architectural designs of the inner circuitry of these devices. In this section, we will explore some of the basic differences in architecture.

**PROMs**

The architecture of the programmable circuits in the previous section in wolves programming the connections to the OR gate. The AND gates are used to decode all the possible combinations of the input variables, as shown in Figure 13-5(a) For any given input combination, the corresponding row activated (goes HIGH) If the OR input is connected to that row, a HIGH appears at the OR output. If the input is not connected, a LOW appears at the OR output. Does this sound familiar? Refer back to Figure 12-9. If you think of the input variables as address inputs and the intact/blown fuses as stored 1s and O s.

The PROM can generate any possible logic function of the input variables because it generates every possible AND product term. In general, any application that requires every input combination to be available is a good candidate for a PROM. However, PROMs become impractical when a large number of input variables must be accommodated because the number of fuses doubles for each added input variable.

**Programmable Array Logic (PAL)**

The PROM architecture is well suited for those applications where every possible input combination is required to generate the output functions. Examples are code converters and data storage (look-up) tables that we examined in Chapter 12. When implementing SOP expressions, however, they do not make very efficient use of circuitry. Each combination of address inputs must be fully decoded, and each expanded product term has an associated fuse that is used to OR them together.

The PAL has an AND and OR structure similar to a PROM but in the PAL, inputs to the AND gates are programmable, whereas the inputs to the OR gates are hard-wired. This means that every AND gate can be pro grammar to generate any desired product of the four input variables and their complements. Each OR gate is hard-wired to only four AND outputs.

The PAL family also contains devices with variations of the basic SOP circuitry we have described. For example, most PAL devices have tristate buffer driving the output Others channel the SOP logic circuit to a D FF input and use one of the pins as a clock input to clock all of the output flip-flops synchronously. These devices are referred to as registered PLDs because the outputs pass through a An example is the PAL16R8, which has up to eight registered outputs (which can also serve as inputs) plus eight dedicated inputs

**Field Programmable Logic Array (FPLA)**

The field programmable logic array (FPLA) was developed in the mid-1970s as the first nonmemory programmable logic device. It used a programmable array as well as a programmable OR array. Although the FPLA is more than the PAL architecture, it engineers. FLAps are used mostly in state-machine design where a large number of product terms are needed in each SOP expression.

**THE GAL 16V8 (GENERIC ARRAY LOGIC)**

The GAL 16V8, introduced by Lattice Semiconductor, has an architecture that is very similar to the PAL devices described in the previous section.

Standard, low-density PALs are one-time programmable. The GAL chip, on the other hand, uses an EEPROM array (located at row and column intersections in Figure 13-7) to control the programmable connections to the AND matrix, allowing them to be erased and reprogrammed at least 100 times. In addition to the AND and OR gates used to produce the sum of product functions, the GAL 16V8 contains optional flip-flops for register and counter applications, tristate buffers for the outputs, and control multiplexers used to select the various modes of operation. Consequently, it can be used as a generic, pin-compatible replacement for most PAL devices. Specific locations in the memory array are designated to control the various programmable connections in the chip. Fortunately, it is not necessary to delve into the addresses of each bit location in the matrix. The programming software takes care of these details in a user-friendly manner.

The major components of the GALAXY devices are the input term matrix; the AND gates, which generate the products of input terms; and the output logic macro cells (OLMCs). Notice that the eight inputs (pins 2-9) are each connected directly to a column of the input term matrix. The complement of each of these inputs is also connected to a column of the matrix. These pins must always be specified as inputs when programming the 16V8. A logic level and its complement are also fed from each OLMC back to a column of the input matrix. This accounts for the 32 input variables (columns in the input matrix) that can be programmed as connections to the 64 multiple-input AND gates.

The eighth product term is connected to a two-input product term multiplexer (PTMUX), which drives the eighth input to the OR gate. The eighth product term also connects to one input of a four-input multiplexer (TSMUX). The output of TSMUX enables the tristate inverter that drives the output pin [I/O(n)]. The output multiplexer (OMUX) is a two-input MUX that selects between the combinational output (OR gate) and the registered output (the D flip-flop). A fourth MUX selects the logic signal that is fed back to the input matrix. This is called the feedback multiplexer (FMUX).

Each of these multiplexers is controlled by programmable bits (AC1 and ACO) in the EEPROM matrix. This is the way that the OLMC configuration can be altered by the programmer. Another programmable bit is the input to the XOR gate. This provides the programmable output polarity feature.

The FMUX selects the signal that is fed back into the input matrix. In this case, there are three possible selections. Selecting the MUX input that is connected to an adjacent stage or the MUX input connected to its own OLMC I/O pin allows an existing output state to be fed back to the input ma trix in some of the modes of operation. This feature gives the GAL 16V8 the ability to implement sequential circuits such as the cross-coupled NAND gate latch I/O pin to be used as a dedicated

input as opposed to an aunt One of these two feedback paths is chosen, depending on the MODE that the chip is pro grammar for. The third option, selecting the output from the D flip-flop, al lows the present state of the flip-flop (which can be used to determine the next state) to be fed back to the input matrix. This allows synchronous se sequential circuits, such as counters and shift registers, to be implemented.

With all of these options, it would seem that there must be a long list of possible configurations. In actual practice, all these configuration decisions are made by the software. Actually, the GAL 16V8 has only three different modes: (1) simple mode, which is used to implement simple SOP combinational logic without tristate outputs; (2) complex mode, which implements SOP combinational logic with tristate outputs that are enabled by an AND product expression, and (3) registered mode, which allows individual OLMCs to operate in a combinational configuration with tristate outputs (similar to the complex mode) or in a synchronous mode with clocked D FFs synchronized to a common clock signal.

The GAL 16V8 is an inexpensive and versatile PLD chip, but what if a de sign requires more hardware resources than is contained in the 16V8? It may be possible to split the design into smaller blocks that can be implemented in several 16V8 chips. Fortunately, there are other members of the GAL family to choose from. Another popular, general-purpose PLD is the GAL 22V10.

This device has 10 output pins and 12 input pins in an architecture that is similar but not identical to the GAL 16V8. Groups of product terms are logically summed with an OR gate, which feeds an OLMC. Unlike the 16V8, however, each OR gate in the 22V10 does not combine the same number of product terms.