MODULE 7 - INTRO. TO PROGRAMMABLE DEVICES

OVERVIEW:
Presently designers seldom use classical logic gates to design digital circuits. Most of the modern designs use programmable logic devices (PLDs). There are actually several classes of programmable devices: **PALs**, or programmable array logic, **PLAs**, or programmable logic arrays, and **GALs**, or gate array logic. All of the classes of programmable logic are very similar.

Recall that all truth tables can be reduced to a Boolean algebra expression, and any Boolean algebra expression can be realized using combinations of AND and OR gates. In a programmable logic device (PLD), a matrix of gates with multiple inputs and multiple interconnections is possible by selectively burning open all of the places you do not want the devices to connect. A special programmer, called a PLD burner, interfaces with a personal computer and allows you to input the Boolean algebra expression, identifying the pin numbers where each input variable is to be applied. The computer program converts the code into a programming sequence that selectively burns open the logic paths that are not allowed, leaving only the circuit you specified with your equations.

**CONCEPT 7.1:** PLD’s Are Used To Make Extremely Large Circuits

PLD’s are typically very dense, capable of generating some extremely complex circuits. The GAL22V10 shown below has the capability of inputting up to 10 input variables with 10 outputs, each output capable of having 9 terms in the equation. This means that the little "AND" gates shown are 40 input AND gates because you can also input the outputs, or "STATE VARIABLES", back into the circuit.

The GAL22V10 also has "D" latches on the outputs, which we will use in the coming modules to design sequential logic, or logic that remembers where it is in a sequence of counts or control steps. On the diagram in Figure 1 below, the "AND" gate inputs appear to have only one wire going into each. This is because the wire is a 40 wire bus. Connections to the bus are represented by drawing X’s where the bus intersects with the input wire of the matrix to be input into the "AND" gate of interest. This type of diagram shows the designer approximately what his design will look like. The realization of the design is accomplished by studying the truth table, and extracting the Boolean algebra equations for each truth table output.

One of the powerful things about designing with PLD’s is that when you have generated the truth table, you do not need to simplify the resulting logic
design. There are almost always many more gates, inputs, and outputs than is ever needed for the typical combinational logic circuit design.

FIGURE 1: SCHEMATIC DIAGRAM OF THE GAL22V10 PLD
CONCEPT 7.2: PLD’s Use Arrays and Clusters of Logic Called “Cells”

If you break down the GAL22V10 into cells, you see that the components are a series of Parallel AND gates ORed together for each cell. The ORed output of each cell goes into the "D" input of a latch. The output of the latch and its inverse, and the output of the OR gate and its inverse are individually selected as the final cell output. One last feature is to add a tristate gate (which is basically a switch) to the output of the cell so it can be connected or disconnected from a line with other cells connected to the same wire.

DEFINITION 7.1: “D” Latch
A “D” Latch is a device that has a single bit “D” or data input that is either at a “1” or “0”, and a clock input that triggers the “D” latch to capture the state of the “D” input. The captured level is output at a pin labeled “Q”. The “D” latch can store the high or low state of a data line. The “D” latch is a simple single bit memory device.

DEFINITION 7.2: PLD
Programmable Logic Device. A PLD is an array of AND and OR gates that can be interconnected by a computer controlled programmer to form an SOP (Sum of Products) combinational logic circuit. PLD is the general name given to all such devices and includes PALs (Programmable Logic Arrays), GALs (Gate Array Logic), PLAs (Programmable Logic Arrays), and any other special devices such as CPLDs that can be computer configured into logic circuits.

The AND-OR cell configuration is not the only cell structure. There are OR-AND type devices, and some other specialized PLD’s that have very uncommon organizations. Every type is described by the manufacturer with instruction about how to use it. The SOP or AND-OR organization of the GAL22V10 is widely used where a classic combinational logic design is desired.
CONCEPT 7.3: A Common Language Used to Design GALs is “CUPL”

CUPL stands for Cornell University Program Language. It was written as a formal way of describing logical functions before any significant PLDs (Programmable Logic Devices). Because it was easy to describe a logic design using Boolean algebra in CUPL, it was adopted by several of the large chip manufacturers as the language of choice for programming their devices. Their chip programming equipment came with a CUPL based compiler. The compiler converted the design code into the correct sequence of electrical signals necessary to burn the connections into the PLD. The result was a PLD chip that was transformed into a circuit that behaved like the original Boolean algebra design.

CONCEPT 7.4: An Example of a CUPL Program

A CUPL program has the following components:
1. Program name field;
2. Name of device being created;
3. Date;
4. Revision;
5. Designer;
6. Company or Organization;
7. If part of another assembly, which?
8. Location;
9. Device Type;
10. Input Pin Definitions;
11. Output Pin Definitions;
12. Defining Equations (Boolean algebra expressions of outputs in terms of inputs in ABEL format.)
Using PLD’s to design combinational logic is no different than designing circuits built from conventional logic. The biggest difference is that simplification is generally not needed. Most PLD’s have more than enough gates in the array to build the unsimplified version of the circuit. In addition, extremely complex circuits can be built. The GAL or PAL 22V10 will support 10 inputs, 10 outputs, a clock line, and will feed back the output lines. Ten input variables would require a truth table with 1024 possible combinations. Most designs and applications will only use a small fraction of the device’s capability.