

# Programmable Logic Devices

*The glue logic required to design a microcomputer system is typically implemented using programmable logic devices (PALs, CPLDs and FPGAs). This lecture describes the structure of these three types of devices, their advantages and disadvantages.*

*After this lecture you should be able to select and justify the choice of a programmable logic device based on: (1) the complexity of the design; (2) maximum allowed propagation delay; (3) device cost; and (4) available development tools. In future design problems you will be required to select an appropriate device to implement your solution.*

*You should also be familiar with basic PLD terminology including the terms PAL, CPLD, FPGA, OTP, sum of products, CLB, LUT, netlist, and place-and-route.*

## 1 Introduction

In recent years programmable logic devices (PLDs) have all but replaced special-purpose logic devices such as AND gates, flip-flops, counters, multiplexers, etc. PLDs are chips that can be programmed, and often re-programmed, to implement different logic functions.

The main reason for using programmable logic is to reduce total costs. This is due to a number of reasons: One important advantage is that design with PLDs is faster and this reduces the time required to bring a product to market. Programmable devices also reduce the risks associated with product development since they allow last-minute changes, often without having to redesign circuit boards. Since PLDs often replace several other special-purpose devices the design usually has fewer components and this reduces PCB, assembly, test and repair costs. Using PLDs also means fewer parts need to be stocked and this reduces inventory costs. Since more of the logic is integrated into each chip the number of interconnections is decreased and this increases the reliability of the product.

Exercise: Name 5 advantages to using PLDs.

Of course, there are some disadvantages to using programmable logic. Design with PLDs requires additional development software and hardware which is often very expensive. Design staff often need to be trained to use new design tools. In addition, parts must be programmed before they can be assembled into a final product.

In spite of these disadvantages programmable logic usually makes economic sense except for very simple (e.g. bus buffers, latches, some decoders),

very complex (e.g. CPU), or very high-speed circuits (e.g. DRAM controller). Even for one-of designs it's often easier to use PLDs if the development tools are available.

Compared to ASICs (standard-cell or gate arrays) PLDs offer lower NRE costs (zero vs tens of thousands of dollars), fast (1 hour vs several weeks) design turn-around, lower risk and simpler design tools. On the other hand, ASICs will operate at higher speeds and will be less expensive at very high volumes (typically many thousands of parts).

## 2 PALs

A PAL (programmable array logic) is a programmable logic device in which each output is computed as a two-level "sum of products" (an OR of ANDs).

Modern PALs use a programmable "macro cell" on each output. These macro cells contain a D flip-flop and programmable switches (multiplexers) that configure the macro cell as either a registered or combinational output and as either a positive-true or negative-true logic. Each of the first-level AND (product) terms can be programmed to include any combination of the inputs, the outputs, or their complements. Each of the second-level OR (sum) terms is connected to a fixed number of product terms (AND gate outputs) (typically 8 to 12).

The advantages of the PAL architecture include low and fixed (two gate) propagation delays (typically down to 5 ns), and simple, low-cost (free), design tools. However, the PAL architecture limits the design to simple state machines and simple combinational circuits.

Modern PALs typically have EEPROM configuration memories and are programmed with device programmers similar to EPROM programmers.

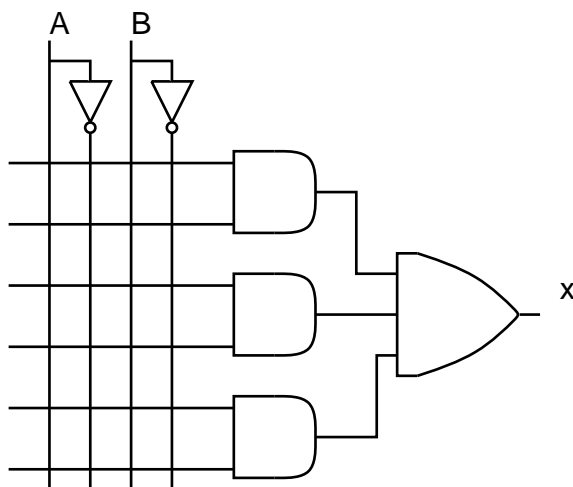
Because the applications tend to be limited to simple state machines and combinational circuits, PALs are usually designed with a combination of truth tables, tabular state machine descriptions and boolean logic equations. PLD languages such as PALASM, ABEL or CUPL convert these descriptions to fuse maps that are then used to program the devices. These tools are often available free from PAL manufacturers, usually come with target libraries that include that manufacturer's products.

As an example, the following CUPL program describes the function of one output of an 18CV8 PAL output using logic equations. Pins 1 and 2 are inputs and pin 12 is an output:

```
Device p18cv8 ;

Pin 1 = A ;
Pin 2 = B ;
Pin 12 = x ;

x = A & !B # A & B ;
```



Exercise: The grid in the diagram above shows the possible connections between two inputs and their complements and three 2-input AND gates. The 3-input OR gate is connected to the output of the AND gates to create a sum-of-products output.

Mark the intersections in the grid that represent connections that would have to be made to implement the logic equation described by the program above.

How can you create a don't-care input to the OR gate?

What is the maximum number of product terms that can be included in the equation for the output 'x' in the above architecture?

EEPROM-configured PALs are programmed with device programmer (similar to EPROM programmer).

An typical example of a PAL is the AMD PALCE20V10. This 24-pin chip contains 10 pins connected to outputs of macro cells, 11 dedicated inputs and one input which can be used as a the clock for a total of 22 possible inputs.

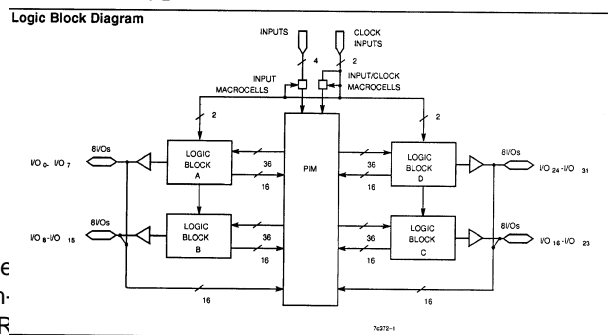
Exercise: How many inputs (columns) will be available to each AND gate in the AND matrix for a 20V10?

PALs range in price from about \$2 - \$15 depending on the number of macro cells and required propagation delay.

### 3 CPLDs

Since the number of possible interconnections in a PAL grows as the product of the number of inputs and outputs it's desirable to use other architectures as the number of possible variables in each term exceeds about 10.

There are a number of PLD architectures, typically called CPLDs, (complex programmable logic devices) that combine larger numbers of sum-of-product macrocells into structures that allow a smaller number of input/output pins to be used as inputs to the product terms. CPLDs also typically allow the macrocells to select between two or more flip-flop clock sources. The following diagram shows a typical CPLD architecture:



The advantage of CPLDs is that more complex designs can be implemented. Propagation time is only slightly worse than an equivalent-process PAL due to one extra set of programmable interconnections.

As with PALs, CPLDs have an internal configuration memory (typically EEPROM) and are also programmed with device programmers.

Smaller CPLDs can be designed with logic languages such as ABEL or CUPL but larger CPLDs are typically designed using schematic capture (using a CAD program to prepare schematics) or even through synthesis from HDLs such as VHDL.

A typical CPLD is the Cypress CYC372. This device, as shown above, contains 64 macrocells and 32 I/O pins in a 44-pin package with an overall propagation delay of 10 ns.

Typical CPLDs cost between \$5 - \$50 depending on the number of macrocells, the speed and the package.

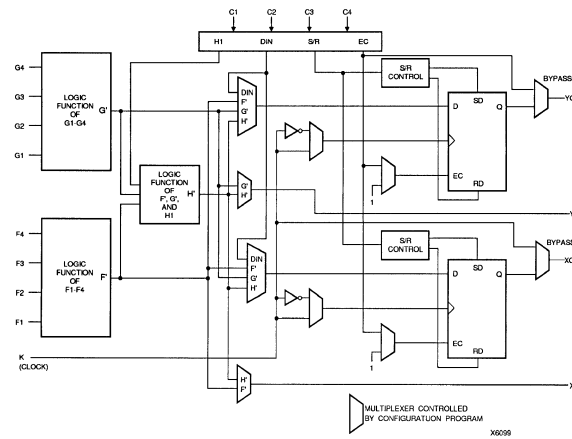


Figure 4. Simplified Block Diagram of XC4000 Family Configurable Logic Block.

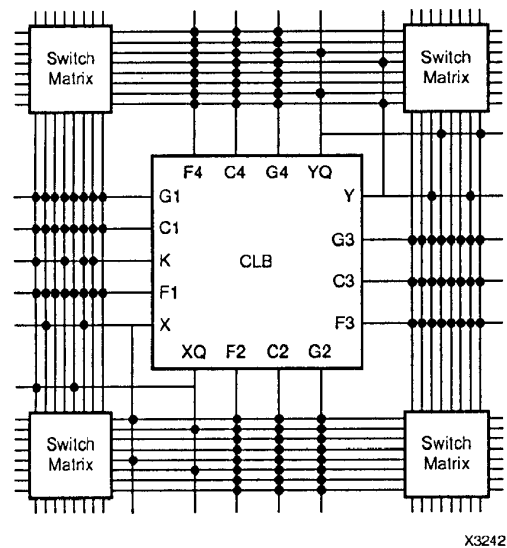
while the following shows how CLBs can be connected to their “neighbors”:

## 4 FPGAs

FPGAs (Field-Programmable Gate Arrays) are PLDs with large numbers of small macro-cells each of which can be interconnected to only a few neighboring cells. A typical FPGA might have 100 cells, each with only 8 inputs and 2 outputs. The output of each cell can be programmed to be an arbitrary function of its inputs. FPGAs typically have a large number (> 64) of I/O pins.

FPGA architectures vary in the complexity of their individual cells (simple cells are “fine-grained” and complex cells are “coarse-grained”) and the flexibility of the interconnections between cells (“routing resources”). Simple cells are arranged as simple ROM-like look-up tables (LUTs) while more complex cells such as the Xilinx CLBs (Configurable Logic Blocks) can include more specialized logic such as carry look-head generators for high-speed adders and may include multiple levels of logic within the cell.

A typical FPGA is the Xilinx XC4003. The XC4000 family is a coarse-grained architecture in which each CLB has 2 sets of 4 inputs, a second level of logic to combine the two intermediate outputs and 2 registered outputs. The XC4000 CLBs also includes features for efficient arithmetic processing. The following diagram shows the structure of a Xilinx CLB:



FPGAs are ideal for designs that require large amounts of logic since it’s possible to integrate multiple storage registers, arithmetic and logic circuits, controllers, etc. on the same device.

One disadvantage of FPGAs is the relatively large propagation delays. To complicate things, these delays are hard to predict before the circuit design is finished. This is due to the need to route signals through multiple levels of logic and interconnection blocks. Typical RAM-based FPGA circuits will have propagation delays of 20 to 50 ns.

Another disadvantage of FPGAs is the expensive (\$3000 and up) and slow design software. FPGAs are typically designed using either schematic capture or HDLs. This is typically done with third-party tools (e.g. Viewlogic schematic capture or Synopsys FPGA Compiler). These tools then output netlists

(lists of gates or cells and how they are to be logically interconnected). The components in the netlist must then be assigned to specific cells on the device and the routing between cells laid out. This “place and route” process is a complex optimization problem which typically includes constraints on propagation delay and the number of cells used.

Although [E]EPROM and OTP (one-time programmable) FPGAs are available, the most popular designs use RAM to store their configuration. The device’s configuration RAM thus has to be loaded each time it is powered up. Stand-alone designs can load their configuration from serial EPROMS while FPGAs in larger circuits can be loaded from another device such as a microprocessor.

It should be noted that OTP fuse-based FPGAs are considerably faster than RAM-based FPGAs because signals have to propagate through fewer gates.

FPGAs cost between \$10 and \$300 depending on the number of cells, pins and speed.

## Summary

The following table summarize some of the differences between the different types of PLDs. Note that the values quoted below are approximate and change continuously as newer technology becomes available.

Device	PAL	CPLD	FPGA
cost	\$2 - \$15	\$5 - \$50	\$10 – \$300
macro-cells	8–10	32 – 128	100 – 1000
pins	20 – 24	44 – 160	84 – 256
prop. delay (ns)	5	> 8	5-10 per CLB
clocks	1	2 – 4	> 2
typical device	AMD 22V10	Cypress 7C372	Xilinx 4003
configuration	EEPROM	EEPROM	RAM or OTP
design	boolean equations	HDL or schematic	HDL or schematic