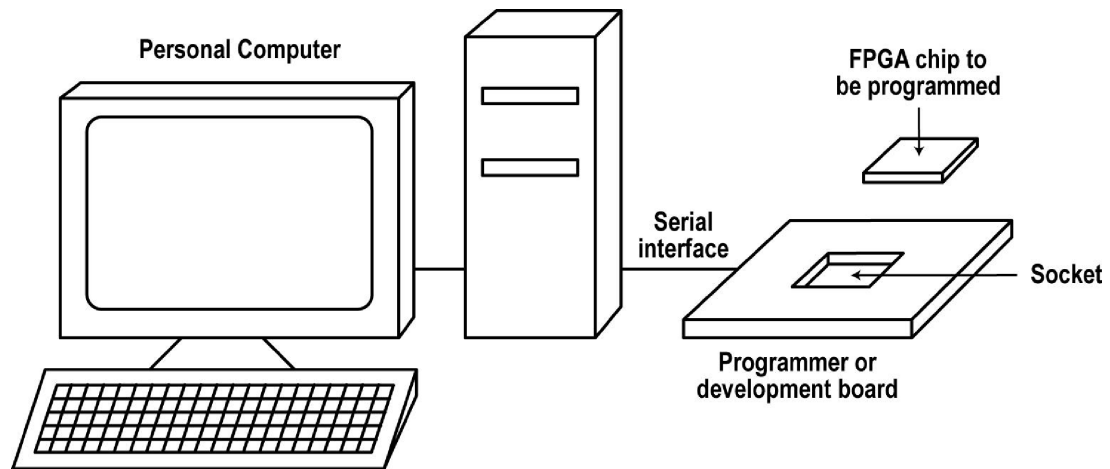


# Programming PLDs

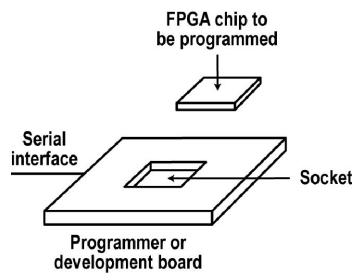
# A PLD Development System



This figure shows a development system used in programming FPGAs. It consists of a personal computer (PC), software, and a programmer or development board.

Software that runs on the PC includes a hardware description language (HDL) as well as simulation and synthesis software that converts the design input information into the bit pattern to be loaded into the PLD.

# Development Board

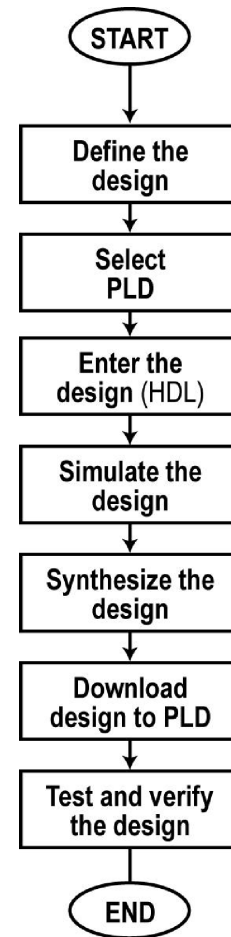


A programmer or a development board contains the hardware to support the download of the bit pattern to the device to be programmed. A programmer is essentially a box with a socket into which the PLD is plugged. It contains necessary power supplies as well as a serial interface to the PC. A development board is a PCB containing the PLD and other support hardware that makes it easy to build a prototype of the end circuits. It also serves as a programmer and interfaces to the PC.

Every PLD manufacturer supplies a development system for each type of PLD they make. Some development systems and software are also sometimes available from third party suppliers.

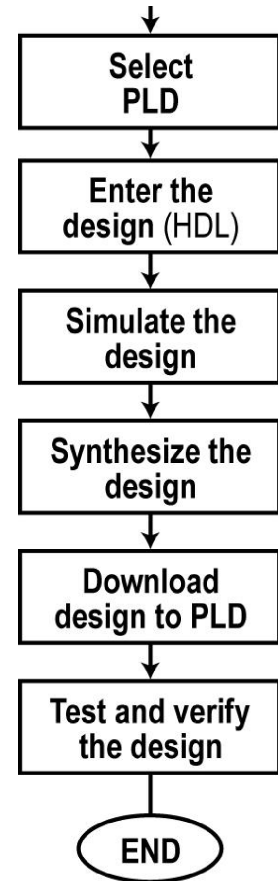
# The Design Process

The design process for a FPGA or other PGA is outlined in the flow chart shown here. The design begins by defining the project in terms of the logic and functions the hardware is expected to perform. It may take one of several forms including truth tables, Boolean logic equations, flow charts, state diagrams, and even verbal descriptions. For simple designs, a single engineer can use paper and pencil during this phase to create the design. For very large projects, the design is next partitioned into major logical sections and then each is designed by a different engineer.



# The Design Process

Next, a commercial PLD is selected. It could be it a simple PAL or a FPGA. Accompanying that device will be some form of development system consisting of a hardware platform (PCB) containing the target chip plus a collection of software that typically runs on a PC. The designer will then use the software on the PC and enter the design details in one form or another. The software will then translate the inputs into bit patterns of 0s and 1s that define the device interconnections. This bit pattern is then sent to the chip on the development system board for testing.



# Entering the Design

Early PLD development software packages used what is called schematic entry. It is a graphical way to draw the logic diagram on the screen of the PC using the mouse and keyboard. Standard logic symbols in the software are accessed and placed on the screen. Lines are then drawn to interconnect them. The process is similar to that still used in some electronic simulation programs like Electronic Workbench (EWB)/Multisim.

The software then translates the schematic/logic diagram into what is called a netlist. A netlist is a text file for a computer that defines interconnectivity in a circuit. It is a list of statements that essentially define every logic element and circuit to be used and state which output is connected to which input for every connection in the circuit. This netlist becomes the input to the remainder of the development software.

# Hardware Description Languages

While graphical schematic entry is an option for some of the older development systems, today most PLD design and programming is done with what is called a hardware description language (HDL). This is like other programming languages in that it uses a unique syntax to enter the design information. Lines of code name the various inputs and outputs then define the logic function and operation.

In the early days of PLD development, each vendor had its own HDL. Over the years, HDLs have become standardized so that one or two languages or both occur in each development software suite.

Some of the early HDLs created for simple PLDs include PAL Assembler (PALASM), Advanced Boolean Expression Language (ABEL), and Common Universal tool for Programmable Logic (CUPL)

Most of these are still available to use on the simple PLD products in use.

# Advanced HDLs

The two most widely used HDLs are called Verilog and VHDL.

The earliest HDL that became a popular standard is Verilog. Since it was created in the 1980s, it has been enhanced and updated over the years and is now a formal Institute of Electrical and Electronic Engineers (IEEE) standard.

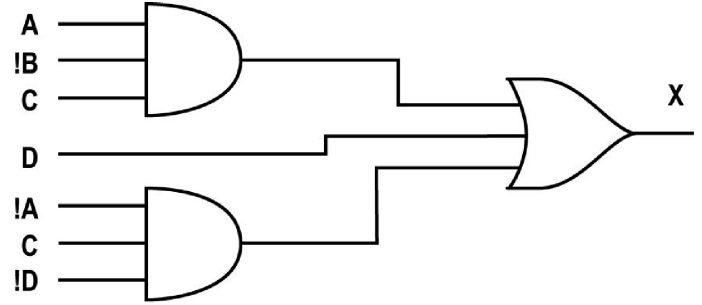
VHDL was developed as part of the Department of Defense's Very High Speed Integrated Circuit (VHSIC) in the mid 80's. VHDL means VHSIC HDL. It too is an IEEE standard.

Most company development system software uses one or the other or may support both languages.



# HDL Coding Example

An example of HDL coding is shown below. This shows how the simple logic circuit is defined by the software syntax.



$$X = (A \& !B \& C) + D + (!A \& C \& !D)$$

Begin

port (A, B, C, D: in bits; X: out bit)

End

Begin

X <= A and not B and C or D

or not A and C and not D

End

# Coding Explanation

The first part of the coding defines and names the inputs and output what are designated by letters of the alphabet. Notice the Begin and End designations tell the software where the beginning and ending of each statement occurs.

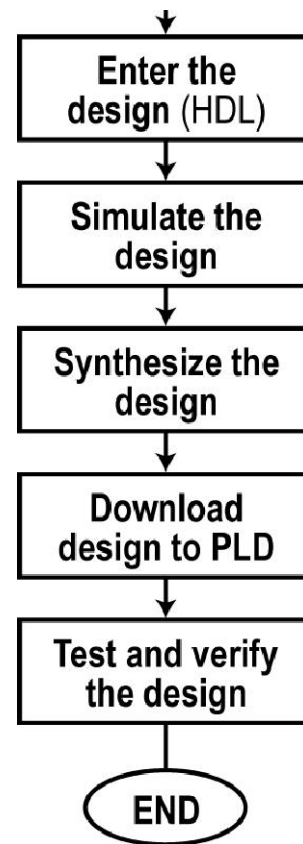
In the second part of the code, the actual logic operations are just written out in the form shown. Again, Begin and End show the software that this is one logic circuit.

Most HDL coding must be entered in the sum of product (SOP) form. If the original equation is not in this form, you must put it in this form. This is usually done by using deMorgan's theorem to convert from product of sums (POS) to SOP.

Furthermore, you may also want to apply logic minimization techniques with Boolean equation manipulation or the use of Karnaugh maps to provide the simplest form of the equation. Some software packages may perform the POS to SOP conversion and/or minimization for you.

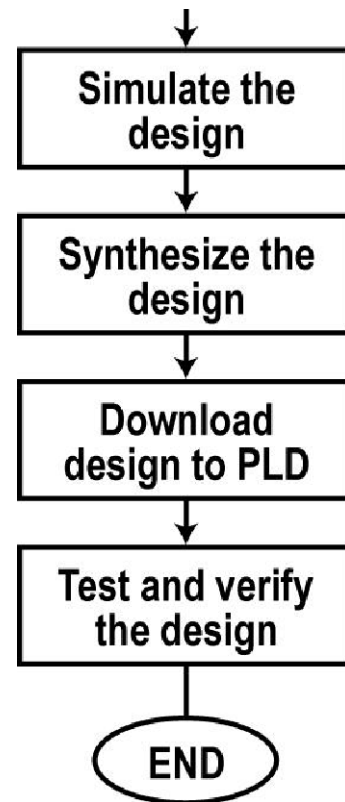
# What the Development Software Does

Once the design has been entered into the system using the HDL of choice, the software compiles the input into a unique code or bit pattern that will then be used as the input to other parts of the software.



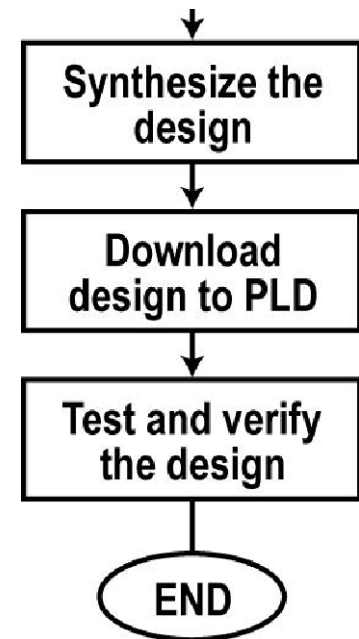
# Development Software: Simulator

The compiler output called object code is then sent to a simulator. The simulator is software that configures itself as the hardware design but in software that runs on the PC. The simulator is then run to be sure the design actually works. It performs what is called functional verification that determines that the logic is correct and timing verification that ensures that there are no timing errors, race conditions or other time violations introduced by the propagation delays of each device used.

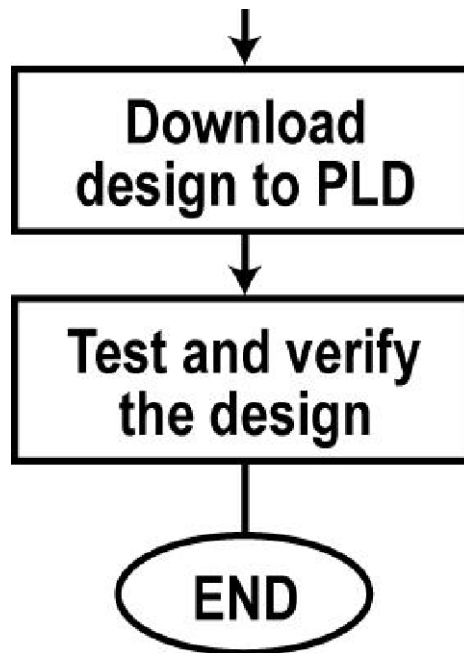


# Development Software: Synthesizer

The verified design is then sent to a portion of the development software called the logic synthesizer. This part of the software is unique to each PLD type. The software performs functions called placing, routing, and packing. Here the software assigns which blocks within the device are to be used and how they are interconnected. The software optimizes the connections to ensure minimum propagation delays. It then converts the compiled design into the bit patterns that actually define the specific connections between logic elements, logic blocks, and other circuits available in the target chip.



# Development Software: Testing



Finally, the resulting bit pattern is downloaded into the chip in the programmer or development board. From there the chip can be tested to be sure that it works as designed.

# Test your knowledge

## Programmable Logic Devices Knowledge Probe 5 Programming PLDs

Click on [Course Materials](#) at the top of the page.  
Then choose **Knowledge Probe 5**.