

## Programming PLDs

Programmable Logic Devices or PLDs come in many varieties. The first were ROMs and the various programmable versions PROMs, EPROMs and EEPROMs. Next came the Programmable Logic Array (PLA) ®, followed closely by the Programmable Array Logic (PAL) ®. What makes programmable logic programmable? The feature of PLDs that makes them programmable is connections or “links” that can be selectively removed. The original PLAs and PALs used bipolar transistor technology and the links were literally fuses that were blown. The process, of course, was irreversible. Once you blew a fuse, you could not “unblow” it. Later, the links were implemented with EPROM and then EEPROM technology, so that chips could be reprogrammed.

Figure 1. is a simple combinational PAL, showing the links as X's. Notice that every link is within a matrix of rows and columns. Thus, each link has a row identity and a column identity, and can be addressed not unlike a memory location. Theoretically, one could manually circle links that one wanted to retain and (somehow) remove the remaining links (fuses).

The PAL manufacturer provided information for accessing the fuses. The chip was put into a programming state and various pins used to provide row and column fuse addresses. A programming voltage would be applied which (with a little bit of luck) would blow selected fuses, leaving others intact. A standard format for designating fuse states was instituted. This is called a JEDIC file. It has a suffix .JED. This is a text file, and theoretically could be made by hand with a text editor. To program a PAL, then, a JEDIC file would be created which (hopefully) represents the intended pattern of blown and unblown fuses. This file is downloaded into a “Programmer” which stores the fuse pattern in its own RAM and uses the PAL manufacturers algorithm for addressing and blowing fuses.

Generating the JEDIC file by hand is impractical. Accordingly, the original PAL vendor created a program called PALASM (for PAL ASSEMBLER) that creates a JED file from the designer's text file. Logic equations are written to represent all outputs in terms of inputs and outputs.

There were several disadvantages with PALs. First, since the programming process is irreversible, you got only one chance per chip. Second, a programmer was necessary. These might cost from three or four hundred to a thousand dollars or more. It also took a whole family of different chips to make typical functions. This was greatly improved with the GAL by Lattice. The GAL ® architecture is a PAL architecture implement with EEPROM. Instead of having a fixed number of outputs which might vary from two to ten depending on the chip, a 20-pin GAL has 10 inputs and 8 macrocells which can be configured

Here's a good site for the ABEL language

[www.ee.upenn.edu/rca/software/abel/abel.primer.html](http://www.ee.upenn.edu/rca/software/abel/abel.primer.html)

<http://mazsola.iit.uni-miskolc.hu/cae/docs/xabel.html>

This is a link for a document called The Abel Hardware Description Language (HDL)

It appears to come from a university in Hungary. It is 32 pages long.

<http://eet.etec.wvu.edu/etec373/SynarioExs/>

ABEL examples by Todd Morton