S I G N E T I C S

AMAZE

PC/MS DOS

USER'S MANUAL

Release 1.7

1989

IMPORTANT NOTICE

Signetics reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Signetics assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products or processes, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified. Applications that are described herein for any of these products are for illustrative purposes only. Signetics makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

LIFE SUPPORT POLICY

SIGNETICS PRODUCTS ARE NOT FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT EXPRESS WRITTEN APPROVAL OF AN OFFICER OF SIGNETICS CORPORATION. As used herein:

- Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

- A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

AMAZE COPYRIGHT NOTICE

Amaze Software is provided to Signetics' Customers and Agents for the purpose of developing custom fuse patterns for programmable logic. Under the copyright, Signetics Corporation authorizes copies to be made of all documentation and magnetic media in their entirety for our Customer's and Agent's internal usage. There is no restriction on the number of copies made for their internal usage.

Signetics Corporation does not allow the use of AMAZE for resale purposes, inclusion in customer OEM equipment or software, or external distribution without prior written authorization.

Signetics Corporation is exempt from consequential damages resulting from the use of the AMAZE package.

INTRODUCTION

Version 1.7 is the latest enhancement of AMAZE and so far the easiest version to use. Its' biggest departure from version 1.65 is that the pinlist information is no longer kept in a separate file (i.e., fn.PIN). It is now kept in the Boolean Equation Entry (.BEE) file under the header '@PINLIST'. In addition, under this header, more than just the pin labels, functions and pin numbers may be found. Also included is useful reference information showing the pin name defined in the data sheet and the label of associated terms controlling the pin's output function. Using the Pin Editor within the BLAST section of AMAZE automatically updates this section of the BEE file or this information can be manually edited via the Users' desired Text Editor.

Another major enhancement gives AMAZE the ability to automatically start Boolean equations with the proper syntax in the BEE file based upon the pin function defined in the Pin Editor. These 'equation starts' are placed in a new design's BEE file under the @LOGIC EQUATION header as comments if the Pin Editor is invoked as a first step. In addition, equation starts for output control information will be placed under the @OUTPUT ENABLE, @I/O DIRECTION, and @FLIP-FLOP MODE headers (PLC42VA12).

AMAZE now supports some PAL architecture devices, such as the 16L8, 16R8, 16R4, 16R6, 16V8 18P8, 20L8, 20R8, 20R4, 20R6 and the 20V8. Schematic capture interface, state transfer and Boolean equations are supported and BLAST provides JEDEC fusemap files for commercial programmers. There is an electronic bulletin board available to support AMAZE users. It may be reached using a 1200 or 2400 baud modem at (800) 451-6644. Set your modem to 1200/2400 baud, no parity, 8-bits, and 1 stop bit (1200/2400-N-8-1). Included in this manual are references to the PLC42VA12. However, AMAZE Version 1.7 does NOT support the PLC42VA12. An update diskette will be available by 4/89 for AMAZE users to compile PLC42VA12 designs.

TABLE OF CONTENTS

Section 3 BLAST Users Guide [continued]

Section 4

TABLE OF CONTENTS [continued]

TABLE OF CONTENTS [continued]

LIST OF FIGURES

LIST OF FIGURES [continued]

LIST OF TABLES

AMAZE

Overview

1. AMAZE Overview

AMAZE (Automated Map And Zap Equations) is a software package designed to simplify the process of using Programmable Logic Devices (PLD) and Programmable Logic Elements (PLE's = PROM's) in logic circuits[1].

AMAZE consists of the following software modules:

1. BLAST (Boolean Logic and State Transfer)
2. DPI (Device Programmer Interface)
3. PTP (PAL to PLD Conversion)
4. SIM (PLD Functional Simulator)
5. PTE (Program Table Editor)

The structure of AMAZE is shown in Figure 1-1. All modules communicate through the PLD standard fusemap file (.STD file). The following paragraphs provides a brief explanation of the capabilities of each module. For a more complete description of each software module, refer to the appropriate User's Guide.

BLAST is an interactive software module which supports the engineer in implementing logic design with Signetics PLDs. It checks the input design data and automatically generates a PLD Program Table (.STD File).

The PLD simulator (SIM) is capable of simulating the programmed PLD and PLE products in two modes: Interactive and Automatic. SIM can automatically create a set of test vectors for testing programmed devices according to the device fusemap file.

---

[1]This manual is not intended as a PLD tutorial. For a complete specification of all the PLD devices and more extensive application notes, refer to the 1989 Signetics PLD Data Manual and the "AMAZE Tutorial and Technology Overview" diskette. Also, an excellent book on the subject of PLD's in general is: Bostock, Geoff, "Programmable Logic Devices", McGraw-Hill, NY, 1988.

PTE is a special Editor for the PLD Program Table. PTE generates/modifies the PLD Standard Table from the information supplied by the user in a friendly and familiar environment. A subset of PTE, called FTE is the Fuse Table Editor used specifically for fusemap editing of the PLHS501 PML device. Note that PTE cannot be used to create or edit PLE (PROM) fusemaps as these are HEX programmer files only.

DPI is a software utility module that provides an interface between the computer and commercial device programmers. It is capable of uploading and downloading files in several formats (including JEDEC) from/to various PLD/PROM programmers.

PTP is a software module capable of converting 20 pin and 24 pin PAL fuse maps into PLD fuse maps. This allows almost automatic conversion of existing PAL designs into logically equivalent Signetics sourced "generic" PLD devices.

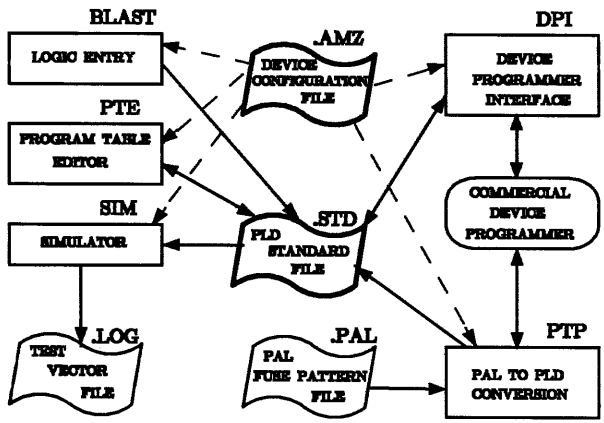


Figure 1–1 AMAZE Structure

1.1 Starting Amaze

Before reading this section, please refer to Section 2 (AMAZE Installation Guide) for information on installing AMAZE. AMAZE may be started from any disk or sub-directory provided a DOS PATH command to the AMAZE installed directory was issued prior to issuing the AMAZE command. For information on the PATH command, please refer to your DOS manual. If you do not wish to set a path command to AMAZE, simply enter the hard-disk directory where AMAZE resides before starting AMAZE. The command to run AMAZE is "AMAZE". When first invoked, AMAZE will display a logo screen along with an information box. The information box is shown in Figure 1-2.

AMAZE creates numerous files associated with a particular design. These files hold such data as the pinlist assignment and design equations, state equations, or fusemap information, to name a few. All files associated with a design will have the same filename, but will have different file extensions (i.e., BEE, SEE, STD, etc.). The information box shown in Figure 1-2 and on the AMAZE logo screen is prompting for the disk (i.e., A:, B:, C:, D:) and the directory path name in which the design files should be created. The last prompt is for the design filename (less extension), that all of the design files will share.

Prior to version 1.65 a diskette placed in drive A: dubbed 'AMAZE MASTER' was required to hold design files. It is still possible to run AMAZE 1.7 with a diskette in drive A:. To do this, the user should first have the DOS path command set with the drive letter of the AMAZE sub-directory and editor sub-directory (i.e., PATH C:\AMAZE; C:\PCWRITE). Next, default to drive A: and at the A> prompt type AMAZE. When the logo screen prompts for the disk drive type "A". At the prompt for the design file path name type "\".

```
Design File Drive  =
Design File Path   =
Design File Name   =
```

Figure 1-2 Logo Screen Information Box

1.2 Main Menu

The modules in the AMAZE package can be accessed through the AMAZE main menu, shown in Figure 1-3, by entering the number of the module after "SELECT:_". Alternately, the modules may be called up individually by entering their name shown on the right hand side of AMAZE menu (i.e., PTP or BLAST) at the DOS prompt, if skipping the AMAZE main menu is desired.

```
                    A M A Z E
              PLD DESIGN SPECIFICATION
                     VER. 1.7
Signetics Corporation                          Copyright 1988

  1-  Boolean Logic and State Transfer..... (BLAST)
  2-  Program Table Editor ..................... (PTE)
  3-  PLD Functional Simulators ................ (SIM)
  4-  PAL To PLD Conversion .................... (PTP)
  5-  Device Programmer Interface .............. (DPI)
  6-  Set Default
  7-  Conversion Utilities
  8-  Critical User Information
  9-  Copyright Information

SELECT :                                  Press ESC to Return
```

Figure 1-3 AMAZE Main Menu

Note that this menu is the only access to the "Set Default", "Conversion Utilities" and "Critical User Information" modules of AMAZE. Also, any design that was originally done on a very early version (Rev A, B, C or D) of AMAZE must be updated to version 1.6. Version 1.6 and later files will run directly on version 1.7. Refer to Section 1.2.2 – Conversion Utilities for more information.

All AMAZE modules will automatically create data files in the design file directory. This allows design information to pass among the different AMAZE modules.

Throughout the AMAZE package, the ESC key is used to exit from the menu. In every menu a set of pre–assigned defaults such as Selection, Format, Editor Name, Baud Rate, or Path Name are displayed when appropriate. For the default selection, press the RETURN key. The currently displayed item will be implemented. Other default information is displayed in appropriate menu boxes. These defaults can be temporarily modified while running the module by selecting the SET function (for that particular default) in the menu, or permanently modified by selecting Option 6 (Set Default) in the AMAZE main menu.

Each module in the AMAZE package has it's own section in this manual. Refer to the appropriate section for more information on a module (i.e., BLAST, PTE, SIM, PTP and DPI).

1.2.1 Set Default

When Option 6 of the AMAZE main menu is chosen, as shown in Figure 1–4, a list is displayed of all AMAZE package parameters which can have a default value. By selecting an item in that menu, the present default is displayed with the opportunity to change it by typing in a new default. When finished selecting defaults, press ESC and the new information will be saved in AMAZE.DEF in the current directory.

Any editor that can create text files without control characters (i.e., nondocument), can be used to input design files in BLAST. However, the editor calling name must be specified in the AMAZE default information Option 9 in the Default Menu. Note that the preset AMAZE default for the editor name is PC-WRITE (calling name:ED). The text editor must either reside in the AMAZE directory or a DOS path must be established to the directory containing the text editor.

```
1- BLAST Main Menu Selection
2- BLAST Compile Menu Selection
3- BLAST Print Menu Selection
4- DPI Main Menu Selection
5- DPI Format
6- PTP Main Menu Selection
7- PTP Format
8- Communication Parameters
9- Editor Name
0- Path Name
```

Figure 1-4 Set Default Menu

1.2.2 Conversion Utilities

The Conversion Utilities menu is shown in Figure 1-5. Two of the menu entries (1 & 2) are provided to update design files created using very early revisions of AMAZE (i.e., rev A, B, C, or D). No update procedure is necessary for files created using revision 1.60 or 1.65.

To update any early revision, copy only the design files (i.e., fn.BEE, fn.PIN, fn.STD) to a new floppy or sub-directory on the hard disk. To update revision A files, two steps are necessary. First, run option 1 to change the device name in the STD file from 82SXXX to PLSXXX. Secondly, run option 2 to update the format of the pinlist (i.e., fn.PIN) to a 1.60 format. To update revision B, C or D files, run only option 2 to update the pinlist.

Because of changes within BLAST, the structure of the pinlist file (fn.PIN) has been changed with AMAZE Version 1.6. Option 2 in the Conversion Utilities Menu will update this file. This conversion must be done only once. It is necessary to update the pinlist of all previous versions of AMAZE. This conversion will keep a backup of the old file as fn.~IN. This conversion can modify only one file at a time.

```
1- Change Device Names to PLSnnn
2- Update Pinlist to Ver. 1.6 Pinlist
3- PLS105 to PLUS405 Conversion
```

Figure 1-5 Conversion Utilities

If the design is from a pre-Rev. D AMAZE, then the .STD file needs to be updated also. Option 1, Change Device Names to PLSnnn, performs this function on all standard (.STD) files it finds in the design file directory. This program will update the first line in the .STD files to reflect Signetics' new numbering policy - from 82Snnn to PLSnnn.

All modules in AMAZE Version 1.6 or later refer to the filename (usually defined in BLAST) for the design and **DO NOT** append the device number (i.e., 159) to the end of the filename entered, as was done in earlier versions except REV. D. In those versions, if the user entered a filename of "DFLOW", a device number (say 153) would be appended to the end of DFLOW, causing information to be stored under DFLOW153.*. In Rev. D, and in Version 1.6 the same information will be stored under DFLOW.*.

Thus, to use old information with the new AMAZE software package, one must include the device number when Filename is asked for (i.e., DFLOW153) or change the old filenames.

The last menu entry (option 3) PLS105 to PLUS405 Conversion, is a convenient way to convert a PLS105 design's .STD file to a PLUS405's .STD file. Only the .STD file will be updated. A new file containing the PLUS405's .STD information will be created. A "Z" will be appended to the beginning of the design's filename. If the filename has 8 characters, the last character will be dropped.

AMAZE

Installation Guide

2. AMAZE PC/MS−DOS Installation Guide

2.1 Dual Floppy System Installation

Files associated with the AMAZE module BLAST no longer fit on one 360K byte diskette. For this reason AMAZE is no longer supported for use on a dual−floppy system.

2.2 System Requirements

The minimum system configuration is as follows:

- o One hard disk drive with 1.8M bytes available
- o one double sided floppy disk drive.
- o 640K bytes of main memory
- o MS DOS 2.0 or later
- o Asynchronous communications port (RS−232) COM1 or COM2 is required for interface to commercial PLD programmers.
- o A text Editor (PCWRITE is provided).

2.3 Hard−Disk System Installation

Before installing AMAZE, check that there is at least 1.8 Mbytes free space remaining on your hard disk. Insert AMAZE diskette #1 into drive A: (or drive B:) and enter

```
CD \
A: (or B:)
INSTALL
```

The INSTALL program will prompt for a sub−directory name into which it will place the AMAZE software. By typing only a carriage return at the prompt, a default

sub-directory of c:\AMAZE will be created, or the user may enter a new drive specification and/or name for the directory where AMAZE is to be installed. The new sub-directory will be created by the installation program if it does not already exist.

Next, the INSTALL program will prompt for the text editor to be used for editing various AMAZE text files. Most editors may be located in any sub-directory as long as a DOS PATH command is issued prior to invoking the editor. The PC-WRITE editor is not copied to the hard disk as part of the install procedure, and must be copied separately. PC-write is the default editor for AMAZE and is included on a separate diskette.

After the first diskette is copied, INSTALL will prompt for the remaining diskettes to be loaded one at a time. When it is finished, it will read the CONFIG.SYS file in the root directory and will prompt if the following commands were not found:

FILES = 20

BUFFERS = 16

The commands must be included to run AMAZE, and INSTALL will add them to the CONFIG.SYS file if an affirmative response to the prompt is given. A CONFIG.SYS file will be created by the AMAZE install routine if one does not already exist.

Design files may be located on any disk and/or in any sub-directory. In addition, AMAZE itself may be invoked from any sub-directory provided a DOS PATH command was issued prior to invoking AMAZE (i.e., PATH C:\AMAZE; C:\PCWRITE) or AMAZE may run from the directory where AMAZE has been installed if a path to the AMAZE directory has not been issued. It is probably most convenient to put a PATH command containing the name of the AMAZE sub-directory and the name of the text editor's sub-directory in an AUTOEXEC.BAT file located in the root directory of the hard disk. That way, after booting, the user simply can type AMAZE while in any sub-directory.

It is recommended that the user invoke AMAZE from either the sub-directory containing the design files or from the sub-directory containing AMAZE. The reason for this is AMAZE keeps track of the current design filename throughout all modules. It does this by storing the filename and path in a file called AMAZE.DEF. AMAZE looks for an AMAZE.DEF file in the directory from which it is invoked, and if it is not found one will be created. Therefore, if AMAZE is started from many different sub-directories at different times, many copies of AMAZE.DEF will exist.

The sub-directory containing or to contain the design files must exist on the hard disk PRIOR to invoking AMAZE, AMAZE will prompt for the design files path name, however, it will not create it if it does not exist. A path error message will appear if AMAZE cannot find the assigned design files directory.

Note: If drive-A diskette is the assigned drive for the design files, then a diskette must be in drive-A before entering AMAZE.

Please refer now to Section 1.1 - Starting Amaze

BLAST

Users Guide

3. BLAST Users Guide

3.1 Description

BLAST (Boolean Logic and State Transfer) is a software module that facilitates the implementation of logic equations into Signetics Programmable Logic Devices (PLD). Figure 3-1 depicts the BLAST capabilities in automating a standard design session. BLAST will check the design data entered by the user and automatically generate a Standard Program Table (.STD file and optionally, the JEDEC Fusemap File). Data from the Standard Program Table can then be used by other software modules such as DPI and SIM.

The advantages of using BLAST are summarized as follows:

- o Design work is done at a high level
- o Automatically translates a high-level design to Program Table Data and JEDEC Programmer File
- o Performs consistency and legality checks
- o Deletes redundant product terms
- o Minimizes time required for tedious manual tasks.

The above features increase productivity and reduce costs in typical design environments. BLAST will perform three operations during a design session:

1. Accepts design information.

   A. With it's own pinlist editor, BLAST immediately performs error checking on data entered to define a design's pin label and function. This information is kept in the Boolean Equation Entry file (fn.BEE).

   B. By using any predetermined text editor (defaults to PC-Write), BLAST accepts Boolean and/or state equations to describe the functions within a design. Equations are entered under BLAST generated headers per their specific function and stored in a file with an extension of BEE or SEE (fn.BEE, fn.SEE).

C. Using a schematic to Boolean conversion program, BLAST can accept a device's logic description in schematic form. It will generate equations from the schematic's netlist and will place them into a .BEE file. This process requires an external schematic capture package (ORCAD, FutureNet, SCHEMA, etc).

2. Compiles the provided information into a Standard Fusemap (or JEDEC FILE) after extensive error checking. In the event of errors being discovered, the detected ERROR CODES (up to 20) with their location (Line #, and Column #) are displayed on the screen and/or printer. After listing error codes, the error messages will be listed or printed for more error clarification. Complete error messages corresponding to the error codes can be found in Appendix B.

3. Provides complete design documentation in a user friendly printed format.

Detailed descriptions of the use and operation of each section of BLAST are included in Section 3.2.2. The user is recommended to obtain some general knowledge about BLAST by browsing through the next chapters, before proceeding into a design session. The user should also become familiar with specific PLD device(s) he/she intends to use by reading the Signetics PLD Data Manual.

Bracket notation in .BEE files [ ], allows the user to take advantage of devices using PML architecture (Programmable Macro Logic; i.e., the PLHS501). Bracket notation may only be used with devices that incorporate PML (folded NAND) architecture. Other device architectures are NOT supported with the bracket notation. Bracket notation for PML is discussed further in Section 3.1.6 and 3.1.4.1.1.

The PLD Application Notes listed in the back of the Signetics Programmable Logic Devices Data Manual have accompaning design files for BLAST in the AMAZE package. These files may be found in archived form on the "AMAZE Tutorial and Technology Overview" diskette included in the AMAZE package. Place the diskette into a defaulted diskette drive (i.e., drive-A) and enter the command "README" for instructions to copy the application notes to a directory of your hard-disk.

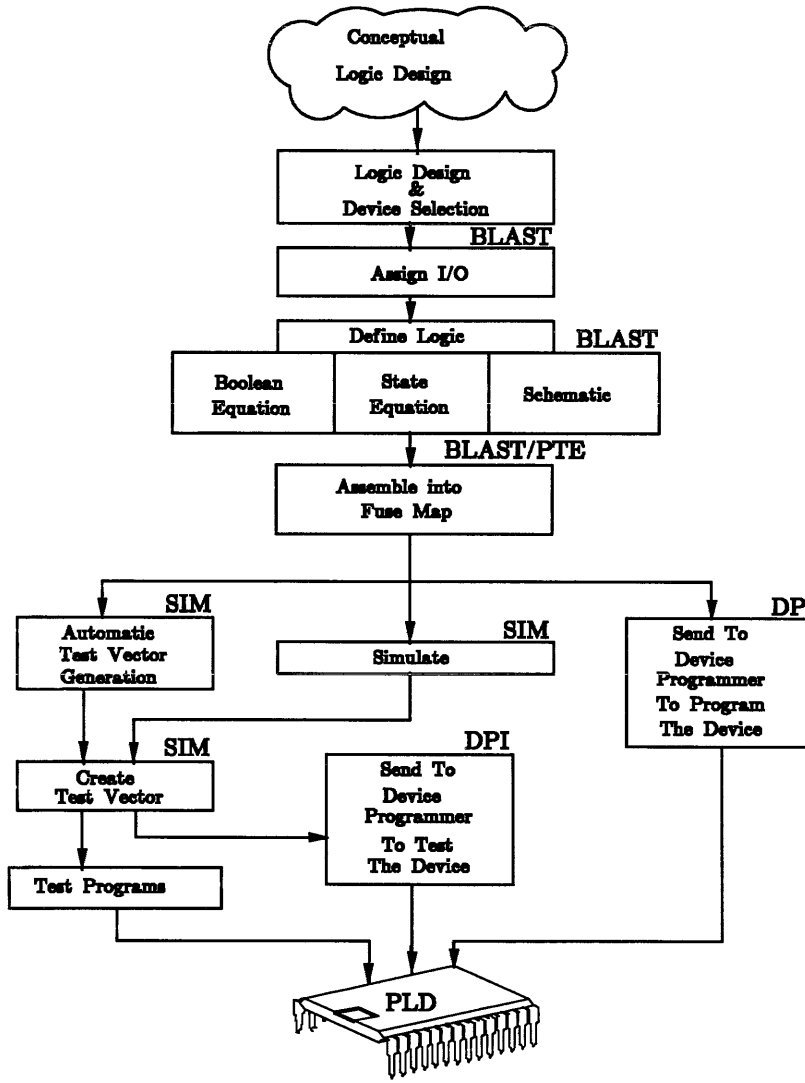


Figure 3-1 Design Flow

3.2 How to use BLAST

The BLAST software module consists of three major sections: EDIT, PRINT, and COMPILE.

By selecting option 1 from the AMAZE Main Menu (Figure 1–2) or typing BLAST from the DOS prompt, the AMAZE program control invokes BLAST. A design session starts by BLAST providing the menu as shown in Figure 3–2.

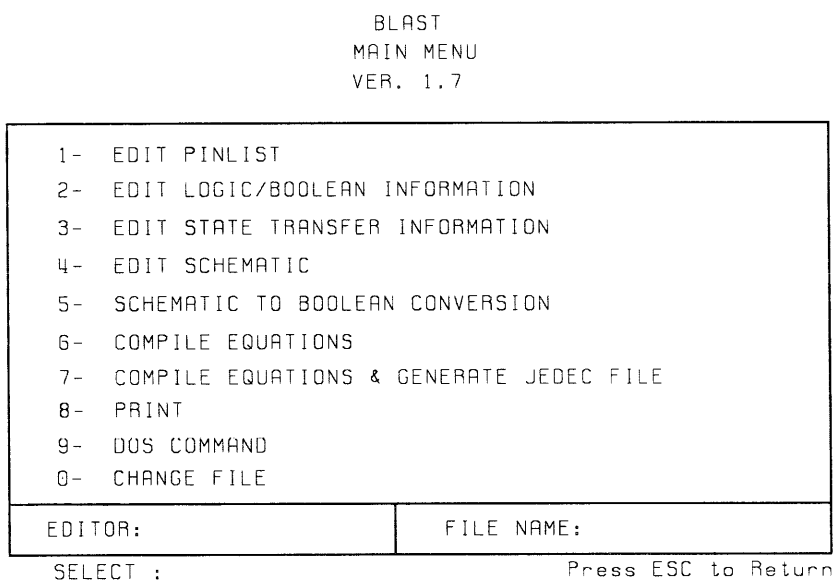


Figure 3–2 BLAST Main Menu

Selecting options 1 through 8 upon entering BLAST, or selecting option 0 at any time will generate a prompt requesting the user to enter a design filename (work file).

The filename can contain any alpha–numeric character including underscore (_) and

hyphen (-), up to 7 characters long. The entered filename will then appear in the FILE NAME box of the BLAST menu for future reference.

Designs may be changed by selecting option 0 in the menu, and providing a different name than the one displayed in the FILE NAME box.

BLAST displays a device menu and requests the DEVICE TYPE only when NEW design information (or a new design filename) is being created. A new set of information will be created if the specified filename does not already exist in the design file directory. BLAST will allow usage of the user's text editor to enter information for options 2 and 3. The editor calling name will be displayed in the EDITOR box, and can be changed by using option 6 (Set Default) of the AMAZE Main Menu. Make sure that the drive name of the editor is included (if other than the default drive), when specifying the editor calling name.

BLAST will search for the editor in all directories where a DOS path has been established before calling BLAST. Thus, if planning to keep the editor in another directory than the AMAZE directory, be sure to implement the PATH command to that directory BEFORE calling BLAST. Refer to the PATH command in the DOS manual for more information. A path command usually resides in an "AUTOEXEC.BAT" file in the root (\) directory of a hard-disk system.

BLAST will edit text files (fn.BEE, fn.SEE) with the specified text editor, and it is the users responsibility to create backup files if desired. However, a back-up of the .BEE file is created upon exit from the pinlist editor (fn.BE~). Also, after a successful compilation of the design, a back-up of the .STD file (fuse pattern) is generated. The BLAST convention for back-up is a '~' as the last character of the file extension. Example:

fn.STD becomes fn.ST~ as a backup file
fn.BEE becomes fn.BE~ as a backup file

Option 9, DOS COMMAND, in the main menu will allow execution of a DOS command such as DIR, COPY, RENAME, ..., or even non DOS commands such as executable programs. In general, this function allows usage of commands that can be done in the system environment. The user can quickly call PTE, SIM, or DPI from the BLAST menu using option 9 and entering the desired module name.

3.3 Pin Editor

BLAST provides special editing for specifying information regarding the selected PLD pin labels and pin functions. This editor is invoked by selecting Option 1 from the BLAST main menu. Figure 3-3 shows the screen displayed for a new design (with no design information) for a PLS159. There are two fields designated to every pin, the Function field (FNC) and the LABEL field. A design starts with all pin labels in a N/C (No Connection) state. These labels should be changed to any string of alphanumeric characters (max. of 12 letters). Cursor movement occurs using [RETURN] [TAB], [→], [←], [↑], [↓], [DEL], and [INS] keys. [TAB] or [RETURN] key moves from pin 1 to the highest pin number of the device and back to pin 1, pin by pin. For PML architectures: ^N goes to the next screen, ^P goes to the previous screen (^=CTRL key). The previous cursor location is remembered. If the function of a pin is user definable, the editor will enter the appropriate function field (i.e. I, O, /B, etc.) in front of the pin. Once moved into a function field, all the possible options for that pin will be displayed on the right hand side of the screen.

In case any illegal entries occur within any field, the interactive editor alerts with an audible alarm and an appropriate error message wil be displayed in the lower portion of the screen. Cursor transfer to another field or exiting the session is not allowed until the cause for the error is removed. Exit from the interactive editing session by pressing the ESC key. BLAST then conducts a general check on all pinlist entries.

```
#####################  P I N L I S T ################     : LABEL :
                                                         : LENGTH 12 CHAR MAX
  LABEL     ** FNC **PIN --------- PIN** FNC **     LABEL :A-Z, 0-9,/,-
N/C         ** CK  **  1-|       |-20 ** +5V **VCC        :N/C:Not Used
N/C         ** I   **  2-|       |-19 ** B   **N/C        :
N/C         ** I   **  3-|   P   |-18 ** B   **N/C        : F U N C T I O N
N/C         ** I   **  4-|   L   |-17 ** B   **N/C        :
N/C         ** I   **  5-|   S   |-16 ** B   **N/C        :
N/C         ** /B  **  6-|   1   |-15 ** B   **N/C        :
N/C         ** /B  **  7-|   5   |-14 ** B   **N/C        :
N/C         ** /B  **  8-|   9   |-13 ** B   **N/C        :
N/C         ** /B  **  9-|       |-12 ** B   **N/C        :
GND         ** 0V  ** 10-|       |-11 ** /OE **N/C        :
                         ---------                       :
                                                         :
                                                         :
      * * * * R E S E R V E D   L A B E L S * * * *      : C O M M A N D
      * EA, EB, LA, LB, PA, PB, D0, D1, D2, D3,   *      :  ESC-SAVE/EDIT
      * C, T, J, K, D, FC, RA, RB                 *      :
      * M0, M1, M2, M3, M4, M5, M6, M7,           *      :
      * VCC, GND                                  *      :
      * * * * * * * * * * * * * * * * * * * * * *      :
```

Figure 3-3 Pin List Screen with No Design Information

If any errors are found, an audible alarm occurs with the error location(s) highlighted. Take action to correct the error, or abort the pinlist screen by pressing the ESC again and respond 'N' to the prompt for saving changes. In this case, the previous pinlist information will remain in the .BEE File.

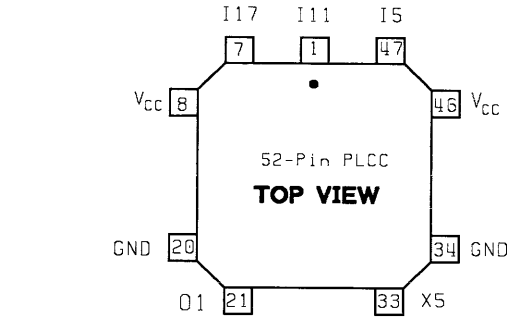
Before saving the results, a backup copy of the old information is automatically created in the design file directory (i.e.,fn.BE~). All pin information is stored under the @PINLIST header of the BEE file.

If at this time, the BEE file has not yet been edited, the Pin Editor will fill out as much information in the BEE file as possible. It will place as comments (i.e., within quotes) equations, based upon labels and pin functions, under appropriate headings as an aid to the user for equation syntax familiarity.

In PROM designs, it is not permissible to use a slash(/) in the PROM pin label within the pinlist. Also, unused pin labels for PROM devices (label not used in any equation) are NOT used in calculating the .STD file. This results in a more efficient use of the PROM. It also allows for the superimpose option to combine designs into a single PROM device.

3.3.1 Dual Screen Pinlist for the PML Device (PLHS501)

The Programmable Macro Logic (PML) devices are so large that consecutive screens are used to display all the signals and pins. Figure 3-4 indicates the exact variable names used with the PLHS501 device which has 52 pins. On the following pages notice how BLAST splits and defines this device package in the pinlist screen editor.



PIN NUMBERS

1	I11	12	I21	22	O2	32	X4	42	I1
2	I12	13	I22	23	O3	33	X5	43	I2
3	I13	14	I23	24	/O4	34	GND	44	I3
4	I14	15	B4	25	/O5	35	X6	45	I4
5	I15	16	B5	26	/O6	36	X7	46	VCC
6	I16	17	B6	27	/O7	37	/B0	47	I5
7	I17	18	B7	28	X0	38	/B1	48	I6
8	VCC	19	O0	29	X1	39	/B2	49	I7
9	I18	20	GND	30	X2	40	/B3	50	I8
10	I19	21	O1	31	X3	41	I0	51	I9
11	I20							52	I10

Figure 3–4 PLHS501 Pin Labels and PLCC Package Layout

```
**********PINLIST**********
      Left                          Right          |      L A B E L
 LABEL ** FNC ** PIN  ________  PIN ** FNC **  LABEL|LENGTH- 12 CHAR,MAX
VCC    ** +5V **   8-|        |-46  ** +5V **VCC    |A-Z, 0-9, /; -, _
N/C    ** I   **   9-|        |-45  ** I   **N/C    |  N/C - NOT USED
N/C    ** I   **  10-|        |-44  ** I   **N/C    |
N/C    ** I   **  11-|    P   |-43  ** I   **N/C    | F U N C T I O N
N/C    ** I   **  12-|    L   |-42  ** I   **N/C    |
N/C    ** I   **  13-|    H   |-41  ** I   **N/C    |
N/C    ** I   **  14-|    S   |-40  ** /O  **N/C    |
N/C    ** B   **  15-|    5   |-39  ** /O  **N/C    |
N/C    ** B   **  16-|    0   |-38  ** /O  **N/C    |
N/C    ** B   **  17-|    1   |-37  ** /O  **N/C    |
N/C    ** B   **  18-|        |-36  ** 0   **N/C    |
N/C    ** O   **  19-|        |-35  ** 0   **N/C    |
GND    ** OV  **  20-|________|-34  ** OV  **GND    |

  * * * * I N T E R N A L   L A B E L S * * * *    |  C O M M A N D S
  *  DB4, DB5, DB6, DB7, XE0, XE1, XE2, XE3,  *    |ESC- SAVE/EXIT
  *  S0,  S1,  S2,  S3,  OE0, OE1, OE2, OE3,  *    |^N - NEXT SCREEN*
  *  GND  VCC                                 *    |^P - PREV SCREEN
  * * * * * * * * * * * * * * * * * * * * * *    |
```

Figure 3-5 PLHS501 - First Screen Display

*Note that "Control-N" will advance to the second half of the PLHS501 pinlist screen.

The "Internal Labels" box contains all of the bidirectional pin and output enable control labels; these are used in the appropriate fields in Boolean Equation Entry of BLAST and are not permitted to be used as pinlabels. See "@I/O Direction" and "@I/O Steering" sections for explanation and use of these labels.

```
**********************PINLIST**********************
          BOTTOM                                    TOP          |      L A B E L
 LABEL ** FNC ** PIN  _________  PIN ** FNC ** LABEL|LENGTH- 12 CHAR,MAX
N/C    ** 0   **  21-|         |- 7  ** I **N/C      |A-Z, 0-9, /; -, _
N/C    ** 0   **  22-|         |- 6  ** I **N/C      |  N/C - NOT USED
N/C    ** 0   **  23-|         |- 5  ** I **N/C      |
N/C    ** /0  **  24-|    P    |- 4  ** I **N/C      | F U N C T I O N
N/C    ** /0  **  25-|    L    |- 3  ** I **N/C      |
N/C    ** /0  **  26-|    H    |- 2  ** I **N/C      |
N/C    ** /0  **  27-|    S    |- 1  ** I **N/C      |
N/C    ** 0   **  28-|    5    |-52  ** I **N/C      |
N/C    ** 0   **  29-|    0    |-51  ** I **N/C      |
N/C    ** 0   **  30-|    1    |-50  ** I **N/C      |
N/C    ** 0   **  31-|         |-49  ** I **N/C      |
N/C    ** O   **  32-|         |-48  ** I **N/C      |
GND    ** O   **  33-|_________|-47  ** I **N/C      |

  * * * * I N T E R N A L   L A B E L S * * * *    |  C O M M A N D S
  *  DB4, DB5, DB6, DB7, XE0, XE1, XE2, XE3,  *    |ESC- SAVE/EXIT
  *  S0,  S1,  S2,  S3,  OE0, OE1, OE2, OE3,  *    |^N - NEXT SCREEN
  *  GND, VCC                                 *    |^P - PREV SCREEN*
  * * * * * * * * * * * * * * * * * * * * * * *    |
```

Figure 3–6 PLHS501 – Second Screen Display

*Note that "Control–P" will return the user to the first half of the PLHS501 pinlist screen.

3.4 Logic/Boolean Information

By selecting Option 2 from the BLAST main menu, BLAST will invoke the assigned text editor and will display a template, such as the one shown in Figure 3-7, for the specified device. Remember that AMAZE uses the command given in the installation routine or in the Set Default Option of the Main Menu to call the text editor. If the editor specified is not invoked, be sure that there was either a DOS PATH command issued pointing to the editor's sub-directory or that the editor resides in the AMAZE sub-directory. Design information can be inserted into the template, below the proper header. Figure 3-4 shows PLS153 entered below "@DEVICE TYPE." The device type is automatically entered by BLAST based on the user selection from the Device menu.

While not a requirement, it is strongly recommended that new designs are started by first invoking the Pin Editor (BLAST option #1). The Pin Editor lists all possible choices for multiple function pins, gives immediate indications of improper inputs and only if invoked as a first step, will enter 'equation starts' under the proper headers. The pinlist information can, however, be edited with the users' Text Editor in the .BEE File (Figure 3-7); extra care must be used since the online error checking/validation used by the pinlist editor is not active during Boolean file editing.

Headers can be created providing that they are not misspelled. The title must follow an '@' sign. **Information entered on the header line will be ignored by BLAST.** A header can be deleted, if the corresponding information does not apply to the design, or if that PLD feature was not used.

```
@DEVICE TYPE
PLS153
@DRAWING
@REVISION
@DATE
@SYMBOL
@COMPANY
@NAME
@DESCRIPTION
@PINLIST

"<--------FUNCTION--------->   <--REFERENCE-->"
"PINLABEL     PIN #  PIN_FCT   PIN_ID      DE_CTRL"
N/C            1      I         I0          -  ;
N/C            2      I         I1          -  ;
N/C            3      I         I2          -  ;
N/C            4      I         I3          -  ;
N/C            5      I         I4          -  ;
N/C            6      I         I5          -  ;
N/C            7      I         I6          -  ;
N/C            8      I         I7          -  ;
N/C            9      B         B0          D0 ;
GND           10      0V        GND         -  ;
N/C           11      B         B1          D1 ;
N/C           12      B         B2          D2 ;
N/C           13      B         B3          D3 ;
N/C           14      B         B4          D4 ;
N/C           15      B         B5          D5 ;
N/C           16      B         B6          D6 ;
N/C           17      B         B7          D7 ;
N/C           18      B         B8          D8 ;
N/C           19      B         B9          D9 ;
VCC           20      +5V       VCC         -  ;

@COMMON PRODUCT TERM "CPT_label = (expression)"
@I/O DIRECTION " D0 ..Dn "
@LOGIC EQUATION
```

Figure 3-7 Template for Boolean Equation Entry (BEE) file for a PLS153

The BLAST Compiler may display "WARNING" messages for certain title lines (i.e., @I/O DIRECTION) if these exist within the ".BEE" file without any design information following them. This is nonfatal for PLD fusemap compilation and can be ignored since the BLAST compiler expects to find data under each header.

As stated before, BLAST processes the information sequentially. Thus the following rule must be obeyed:

DECLARATION BEFORE USAGE

This means that the elements on the right hand side of the equation must be declared in previous statements. This includes declaration of flip-flop type before defining equations for a register. Title duplication is forbidden, except for the @COMMON PRODUCT TERMS as shown in the following example:

```
@COMMON PRODUCT TERM
   cmp1 = IO * /I1 ;
   cmp2 = I1 * I5 ;
@COMPLEMENT ARRAY
   /c = /(CMP1 + CMP2 + I11* s5) ;
@COMMON PRODUCT TERM
   cmp3 = /c * S10 * I4 ;
   cmp4  = [I4 * I5 + I6] *I7 ;
```

Entries within the Logic Information file (fn.BEE) are **OPEN FORMATTED**. Blank character(s) can be inserted at any point of an equation to improve readability. However, no blank characters may be inserted within a label (i.e. IN1 is not the same as I N 1) without changing its meaning. Comments, enclosed in double quotes, can be placed anywhere within this file.

DRAWING, REVISION, DATE, SYMBOL, COMPANY, NAME, and DESCRIPTION sections are for documentation purposes only. The first line under each of the @NAME, @DATE, and @REVISION titles will be printed on the Program Table print out from either BLAST or PTE print commands.

All equations configuring the PLD Control Logic Circuitry (i.e., I/O Direction, Flip-Flop Control, etc.) must be consistent with all other information (i.e., Pinlist, Flip-Flop Mode) throughout the design. All labels on the left hand side of Control Logic equations must correspond to the defined in the Signetics PLD Data Manual.
EXAMPLE: Refer to Figure 3-8:

```
@I/O DIRECTION
  D0 = I0 * I1 * /C ;
```

In the above example, pin B0 is a controlled I/O pin. When I0*I1*/C is true, the pin is an output; otherwise, the pin is either an input or a three-stated output. Note the use and location of D0 on the left side of equation and in Figure 3-5.

One must not specify any information in the Boolean equation file (.BEE) relating to a pin or label which is not used in the design (N/C label in the pinlist). These labels are regarded as UNKNOWN LABELS within BLAST. For example, if B6 is not used (pin 16 in Figure 3-8), D6, and B6 are considered as UNKNOWN LABELS, and usage of these labels on the left hand side of an equation would be detected as an error during the compile process of BLAST.

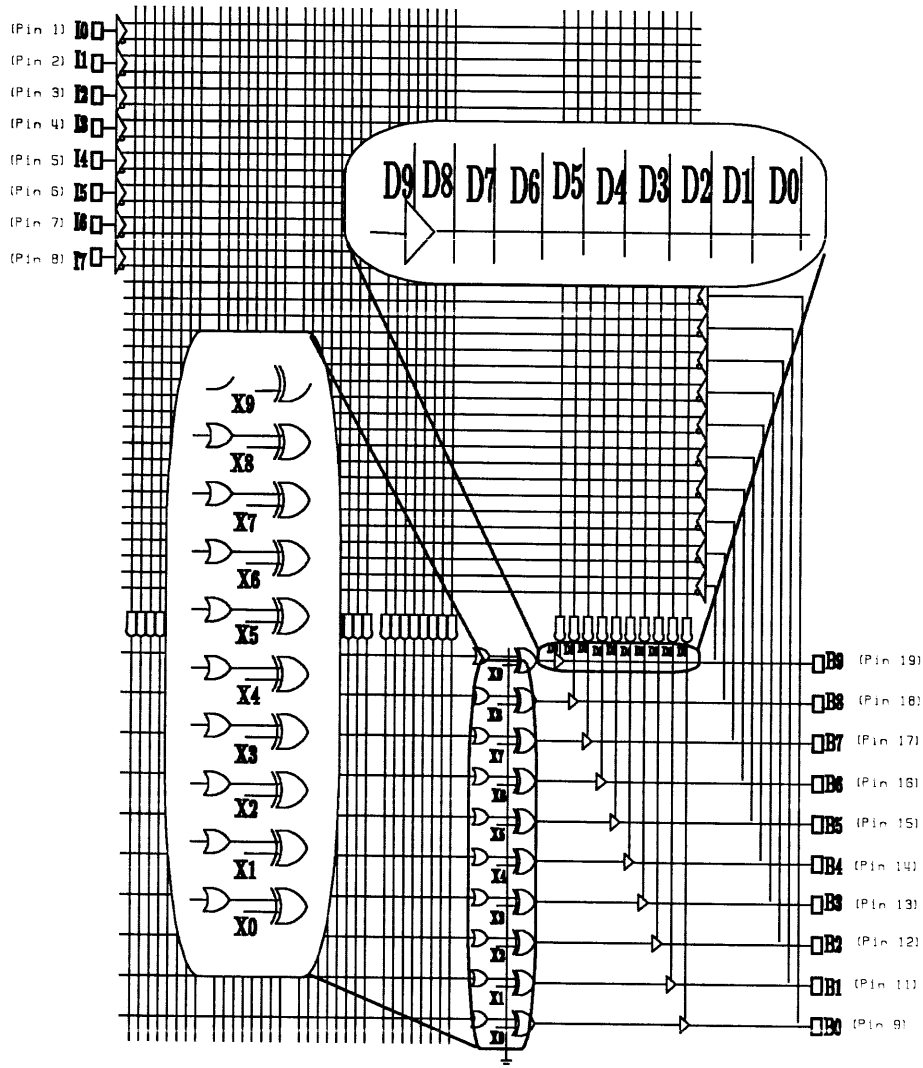


Figure 3-8 PLS153 Logic Diagram (20 Pin PLD)

3.5 .BEE File Headers

@Asynchronous Preset/Reset
@Common Product Term
@Complement Array
@Flip-Flop Control
@Flip-Flop Mode
@Initialize
@Init/OE
@Internal Clocks
@Internal Node
@Internal SR Flip-Flop Labels
@Internal Flip-Flop Labels
@I/O Direction
@I/O Steering
@Logic Equations
@Output Enable
@Pinlist
@Register Load

3.5.1 @Asynchronous Preset/Reset

Devices: PLS155, PLS157, PLS159, PLS179, PLC42VA12

Format:

LABEL = EXPRESSION ;

where LABEL can be RA, RB, PA, or PB. For the PLC42VA12 LABEL may also be PM9, PM0 and RM0–RM9. An EXPRESSION is either "1" (permanently high, active), "0" (idle), or a Boolean expression (controlled).

Description:
The user must write Boolean expressions to define how the flip-flops in the design may be asynchronously preset or reset. The default condition is the asynchronous preset/reset functions are disabled. Only labels indicated in the Signetics PLD data manual (RA, RB, PA, or PB) for the register banks may be used. Valid labels also are generated by BLAST and appear as a comment following this header. Remember, none of the equations for R and P of the same bank of registers should be the same.

(For PLS155)
Sum of Products is allowed for dynamic control of RA, PA, RB, and PB signals.

(For PLS157)
Sum of Products is only allowed for RA and PA. For RB and PB, only define a single Product term. Each equation must be entered separately.

(For PLS159, PLS179 and PLC42VA12)
Only a simple product term is allowed for PA, RA, PB, RB. See Complement Array section for "OR" function implementation.

Example:

```
@ASYNCHRONOUS PRESET/RESET
  PA = 0;          "Preset for Bank A is disabled; default conditions"
  RA = Q0 * /Q1 ;
  PB = Q1 * /Q0  ;
  RB = Q4 * /C   ;  "Bank B registers are reset when (Q4*/C)
                     is true"
```

3.5.2 @Common Product Term

Devices: All

Format:

SYMBOL = EXPRESSION ;

SYMBOL can be 1 to 12 characters. It may contain alphanumeric characters, hyphen, and underscore. A slash (/) may be used only as the first character. EXPRESSION can be a Sum of Product expression or a single product term.

Description:

The Common Product Term field can be used to define the common product terms or subset equations to be used in the Boolean equations. Thus if an equation is repeated, one may assign a symbol to the equation and use the symbol to represent the equation wherever Boolean expressions are allowed The equation must end the with a semicolon(;). This can be considered a form of macro substitution to simplify equation typing tasks where repeated functions are desired. Common Product Term equations may be composed of pin names, internal nodes, or other common product terms. If a common product term symbol is used in an expression, that symbol must have been defined prior to its use.

Example:

```
@COMMON PRODUCT TERM
COMP = /Q0 * in0 * /C  ;
```

The COMMON PRODUCT TERM title can be duplicated and thus specified twice in cases such as following:

```
@COMMON PRODUCT TERM
      Cmpl = i0 * il ;
      Cmp2 = il * i5 ;
@COMPLEMENT ARRAY
      /C = /( cmpl + Cmp2 + Ill * S5);
@COMMON PRODUCT TERM
       Cmp3 = /C * Sl0 + i4* ill ;
       Cmp4 = [I9*I8+I7]*[I6+I5*I4]+I3 ;
```

3.5.3 @Complement Array

Devices: PLS105, PLS155, PLS157, PLS159,
PLS167, PLS168, PLS179, PLUS405, PLC42VA12

Format:

/C = /(EXPRESSION) ;

where EXPRESSION can be any Sum of Product expression or product term. Note that "/C" is a reserved label and should not be used for any function other than the complement array.

Description:

The Complement Array field is used to define the COMPLEMENT ARRAY logic to be used later in the equations where a label is allowed in the expression (i.e., F-F Control, Boolean Equation, I/O Control, Register Load, etc.). The complement array is a NOR gate whose inputs are connectable to the AND gates outputs and whose output is connectable to the product terms inputs.

Example:

```
@COMPLEMENT ARRAY
/C  = /(cpl * pt2+dda_l) ;
/C  =  /( IO*/Il2*I2 + Il*/Q4 + /Q5*Ql ) ;

When two complement arrays (PLUS405) :
@COMPLEMENT ARRAY
/C0 =  /(cp3 + I9*I8*I7+I6) ;
/Cl = /cI9*/I8+cp4*/I7+l5) ;
```

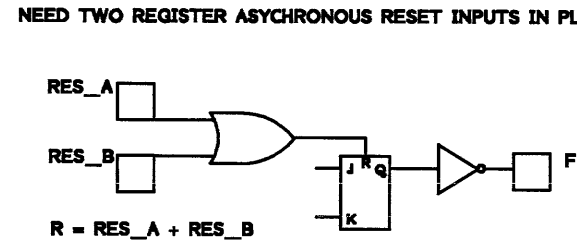


Figure 3–9 Desired Logic

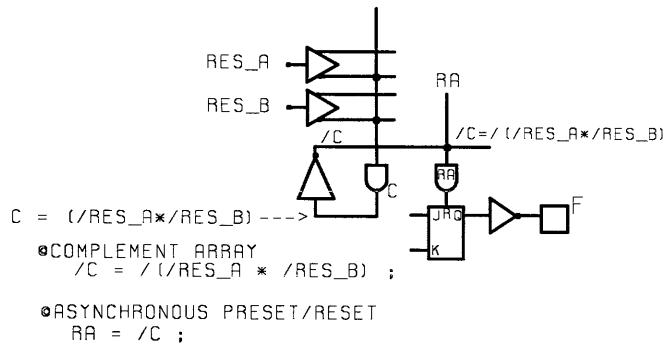


Figure 3–10 PLS179 Implementation

3.5.4 @Flip Flop Control

Devices: PLS155, PLS157, PLS159, PLS179, PLC42VA12

Format:

FC = EXPRESSION ;

where EXPRESSION can be "0" (logic low for D-type flip-flop; this is the default condition), "1" (logic high for J-K flip flop), or a single product term (for J-K or D type flip-flop controlled dynamically).

Description:

The Flip-Flop Control field is used to define the control logic equation of all the flip-flops that were established as dynamically and/or externally controlled in the Flip-Flop Mode field. The user may enter a Boolean expression with only one product term (complement array included) to define the logic for the FC line. If the label FC is assigned in the Flip-Flop Mode field, define an expression for FC in this field. Remember that any flip-flop mode specified as Mn = 1; is a dedicated J/K (or toggle type flip-flop) and is not controlled by the flip-flop control (FC) term, since specifying Mn = 1 blows the fuse connecting the Mn-gate input to the FC control term, and the line floats high, disabling the Mn-gate.

See also @Flip-Flop Mode.

Example:

```
@FLIP-FLOP CONTROL

      FC = 0;                    "D"
      FC = 1;                    "J-K"
      FC = A * /B;               "J-K if A*/B is true"
                                 "D if A*/B is false"
```

3.5.5 @Flip Flop Mode

Devices: PLS155, PLS157, PLS159, PLS179, PLC42VA12

Format:

Mn = EXPRESSION ; or

Mn, Mn = EXPRESSION ;

where n is the register number and EXPRESSION is either "1" (for permanent J-K flip-flop) or "FC" (for D or J-K / D flip-flop dynamically Controlled by the FC line). The equation must end with a semicolon (;). See individual PLD data sheets for the register Mn Assignments.

Description:

The Flip-Flop Mode field is used to define the type of flip-flops chosen to incorporate into the design. The label used to define the flip-flop mode is Mn, where n is the number of the designated flip-flop. Flip-Flop Mode can only be defined for registers which have been assigned a label in the pinlist. If this information is not provided, the flip-flop associated with these labels are defaulted to type J-K/D, controlled by FC. Remember that the default FC function is FC = 0. This implies D-type flip-flops are specified if the corresponding Mn labels are controlled by FC (and the FC line is left in the default condition).

See also @FLIP-FLOP CONTROL.

Example:

```
@FLIP-FLOP MODE
     M0, M2 = 1;              " F0 and F2 are J-K Flip Flops     "

     M1, M3, M4 = FC;         " F1, F3 and F4 may be J-K, D or   "
                              " dynamically J-K/D flip-flops"
                              " depending on the flip-flops"
                              " CONTROL (FC) definition"
```

3.5.6 @Initialize

Devices: 82HS187, 82HS189

Format:

LABEL = EXPRESSION, EXPRESSION, ... EXPRESSION ;

LABEL is the name the user gives to the specific initialized pin label in the pinlist.

EXPRESSION is any one or all the labels used in the pinlist as an OUTPUT.

Description:

The Initialize field is used to allow the user to access output control through the initialization function. You must end the equation with a semicolon (;).

Example:

```
@INITIALIZE
     INITOUT = Out0, Out1, Out2, /Out3, Out5, /Out7 ;
```

3.5.7 @Init/Oe

Device: PLUS405

Format:

LABEL = R0, ... Rn ;

Where Rn equals any of the reserved initialization bits
(R0 - R15).

Rn specifies async reset for F/F n.

/Rn specifies async preset for F/F n.

Description:

The INIT/OE field is used to define initialization of the device or the control of the outputs. The user has an option in the pinlist to select either the initialization function or the output enable function. The pin label given in the label field must match the label used to define initialization. If the user chooses the OE function, this will indicate that the output enable will be controlled externally with an active-low signal.

If the user chooses the INT function in the pinlist screen, for the PLUS405 on pin 19, versus the output enable option OE, then all 16 R-S Registers can have either an asynchronous PRESET input or a RESET input, selected individually. This allows pin 19 to asynchronously force all of the F/F to a user defined initialization state when the pin is asserted high.

Whatever function is specified for each flip-flop individually, all will be toggled by the one input line: INIT/OE. This is an asserted high pin when PRESET or RESET function is selected.

Example:

Example 1.

```
@INIT/OE
"       F0 F1 F2 F3 F4 F5 F6 F7  Output Register Labels"
CLEAR = R0,R1,R2,R3,R4,R5,R6,R7 "Async Reset Function"

 /R8,/R9,/R10,/R11,/R12,/R13,/R14,/R15; "Async Preset Function"
" P0  P1  P2   P3   P4   P5   P6   P7    State Register Labels"
```

Example 2.

```
@INIT/OE
 INIT =   R0, /R1,  R2,  /R15 ;
          ↑    ↑    ↑     ↑. preset P7  when Init high
          :    :    :..... reset  F2  when Init high
          :    :.......... preset F1  when Init high
          :.................. reset  F0  when Init high
```

3.5.8 @Internal Clocks

Device: PLC42VA12

Format:

LABEL = EXPRESSION ;

Where LABEL is the label assigned to a specific flip-flop under the @INTERNAL FLIP-FLOP LABELS header. EXPRESSION is a single product term describing the clock to be used for the specified flip-flop.

Description:

Each flip-flop in the PLC42VA12 may be clocked independently. This section, in conjunction with the @INTERNAL FLIP-FLOP LABELS section, defines the clocks to be used for specific flip-flops. Only those flip-flops which are to be independently clocked need to be defined. The default condition is a clock connection to the common clock input, which is pin 18 on the PLC42VA12.

Example:

```
@INTERNAL FLIP-FLOP LABELS
"P0   P1   P2   P3    P4   P5   P6   P7   P8    P9"
N/C  N/C  N/C  TEST  N/C  N/C  N/C  N/C  LAB2  N/C

@INTERNAL CLOCKS
TEST = A*B*D ;
LAB2 = B ;
```

In this example, P3 and P8 are flip-flops to be clocked by different signals. They were given labels of TEST and LAB2 respectively under the @INTERNAL FLIP-FLOP LABELS header. The clocks were defined under the @INTERNAL CLOCKS header. Therefore, P3 is to have a clock made up of a product term of inputs A, B, and D while P8 is clocked by input B. The remaining flip-flops are clocked by a common clock which is input to pin 18 of the PLC42VA12. Note that pinlist assignments of PLC42VA12 M-pins causes the associated pinlabel to be automatically entered under the @INTERNAL FLIP-FLOP LABELS header; see section 3.5.11.

3.5.9 @Internal Node

Device: PLHS501

Format:

LABEL1 LABEL2 LABEL3 ... LABEL72
LABEL1, LABEL2, LABEL3 ... LABEL72 ;

Where LABELN is the name given to an internal node. Each label may be up to 12 characters in length consisting of alphanumerics, underscores (_), or hyphens (-). The labels must be separated by at least one blank or a comma. A semicolon at the end of the line is optional.

Description:

The Internal Node field is used only for PLHS501 designs. It allows the user to distinguish a node within the circuit which feeds to another internal gate without going to external pins. This field is simply a place in which to list or declare all internal node labels.

The label may be comprised of up to twelve characters. Any single PLHS501 design may contain no more than 72 such labels in the current AMAZE release. Each label must be separated by at least one space and a semicolon is NOT required at the end of the line. After declaration in this field, logic equations using the label as a reference to the node may be written in the @LOGIC EQUATION field.

Example:

```
@INTERNAL NODE
intl     int2
@LOGIC EQUATION
intl = /(il * int2)     ;
int2 = /(i2 * intl)     ;
O19 = /i3 + intl        ;
O21 = /i4  * s * int2   ;
```

3.5.10 @Internal SR Flip–Flop Labels

Devices: PLS105, PLS167, PLS168

Format:

```
LABEL1   LABEL2   LABEL3 ... LABELN
LABEL1,  LABEL2,  LABEL3 ... LABELN ;
```

A LABEL may use up to 12 characters consisting of alphanumerics, underscores (_), hyphens (-), and a slash (/). However, a slash is reserved as the first character in the label indicating inversion. The State Register labels must be separated by at least one blank character, a comma, or a carriage return. A semicolon at the end of the line is optional.

Description:

The State Register Assignment is used to assign labels to the output of internal (buried) registers. This allows description of Boolean and/or state vectors for these flip-flops. These nodes can be treated as register output labels which do not drive a direct device pin. The first label corresponds to the LSB (least significant bit) register (P0). Any specific register, (i.e., P4), must have its name the correct position in this field.

Do not assign labels to any flip–flops which are connected to PLD output pins in the @INTERNAL SR FLIP–FLOP LABELS field. When using the shared registers (P0–P1 in PLS167; P0–P3 in PLS168) which are connected to PLD output pins and are available as state register functions; these labels must be in the pinlist screen but can be treated as state or output vector labels in BLAST.

Example:

Example 1.

```
@INTERNAL SR FLIP-FLOP LABELS
"P0   P1    P2    P3    P4     P5  In this case the only one used"
N/C   N/C   N/C   N/C   REG4       "is (P4), which has been"
                                   "assigned the label (REG4)."
```

```
Example 2.
  @INTERNAL SR FLIP-FLOP LABELS
    N/C          "In this case the label REG4 has been assigned"
    N/C          "to the internal F/F P3."
    N/C    REG4

Example 3.
   @INTERNAL  SR FLIP-FLOP LABELS
   "P0  P1  P2  P3  P4  P5"

   FF0  FF1  FF2 FF3 FF4 FF5   "All six internal F/Fs are
                                assigned labels."
```

3.5.11 @Internal Flip-Flop Labels

Devices: PLUS405, PLC42VA12

Format:

```
LABEL1   LABEL2   LABEL3 ... LABELN
LABEL1,  LABEL2,  LABEL3 ... LABELN ;
```

A LABEL may use up to 12 characters consisting of alphanumerics, underscores (_), hyphens (-), and a slash (/). However, a slash is reserved as the first character in the label indicating inversion. The labels must be separated by at least one blank character, a comma, or a carriage return. A semicolon at the end of the line is optional.

Description:

The Internal Flip-Flop Labels section is used to assign labels to the output of internal (buried) flip-flops. This allows description of Boolean and/or state vectors for these flip-flops. The first label corresponds to the LSB (least significant bit) register (P0). Any specific register, (i.e., P4), must have its name the correct position in this field.

For PLUS405 designs, this section is meant to assign labels to the state flip-flops only. These are the flip-flops (P0-P7) whose outputs are only fed back into the array. Their outputs do not directly connect to output buffers. When designing for the PLC42VA12, based upon the M-Pin function, AMAZE will place entries in this section after using the Pin Editor. For example, if a M-Pin is defined as a register input with Pin_label being the label given to the device pin, then /Pin_label' will automatically be placed in this section corresponding to the correct flip-flop. Also, defining a M-Pin as /BD, /BJ, /OD or /OJ will cause entries to be made under this header to reflect register output pin functions.

Notice the syntax of this section for the PLC42VA12 because it applies to writing equations for this device. When an M-Pin is set to RI (register input) or /BD or /BJ (3-state F/F output with foldback), then the input into the array direct from the pin is either pin-label or /Pin_label. The input into the array from the flip-flop output is pin-label' or /Pin_label' where an apostrophe follows the pin's label to indicate inclusion of the flip-flop in the input signal path from the M-pin.

Use of the pin's label as a flip-flop output label followed by an apostrophe is optional if the pin is defined as B or /B since for this case the flip-flop may or may not be used as a register input. It may be independent of the signal on the associated pin and may be given any label.

The apostrophe is used for input/bidirectional pins where there is a possible discrepancy as to which feedback path to use. In other words, when an M-Pin is being used as an input, two possible paths exist into the array. The first is directly from the pin. The second path is from the pin through the associated flip-flop. Use of the pin name directly on the right hand side of an equation denotes the direct path (nonregistered input). Use of the pin name followed by an apostrophe denotes the path through the flip-flop (registered input).

When an M-Pin is defined as a direct flip-flop output, (i.e., /OD or /OJ) then no apostrophe is used. AMAZE assigns the associated Dn control line (under @I/O DIRECTION) a fixed value of "1". This permanently enables the output buffer. Feedback into the array occurs from the flip-flop's output path, thereby avoiding delay through the output buffer. However, the label assigned in this section is the inverse polarity of the output pin label because PLC42VA12 output buffers are inverting.

Example:

Example 1.

```
   @INTERNAL FLIP-FLOP LABELS
   "P0  P1  P2  P3  P4  P5    In this case the only used"
   N/C N/C N/C N/C REG4       "flip-flop is (P4), which has"
"been assigned the label (REG4)."
```

```
Example 2.
  @INTERNAL FLIP-FLOP LABELS
    N/C          "In this case the label REG4 has been assigned"
    N/C          "to the internal F/F P3."
    N/C    REG4
```

```
Example 3.
   @INTERNAL FLIP-FLOP LABELS
   "P0  P1  P2  P3  P4  P5  P6 P7"

   FF0  FF1  FF2 FF3 FF4 FF5 FF6 FF7  "All eight internal F/Fs are
                                       assigned labels."
```

```
Example 4.   For the PLC42VA12 only.
   @INTERNAL FLIP-FLOP LABELS
   "P0     P1  P2  P3   P4   P5   P6   P7    P8   P9"
   RIN1'  N/C N/C INT  N/C  FF5  N/C  N/C/  N/C /OUT1
```

In this case M0 was defined as a register input (RI) in the FNC field of the Pin Editor. With associate pinlabel "RIN1" (notice the apostrophe). P3 and P5 outputs are either not associated with a M-Pin (implies pin function of I, O, /O, or possibly B or /B) or, the associated output pins were assigned a function of /OD or /OJ and the pin labels are /INT or /FF5. M9 was assigned a function of /OD or /OJ with a pin label of out1.

3.5.12 @I/O Direction

Devices: PLS151, PLS153, PLHS153, PLUS153, PLS155, PLS157, PLS159, PLS179, PLC473, PLHS473, PLS173, PLUS173, PLHS18P8, PLHS501, PLC42VA12

Format:

LABEL = EXPRESSION ;

(For PLS151)[1]
LABEL = /(EXPRESSION) ;

Where LABEL is the signal (Dn or DMn) controlling the three-state drivers on bidirectional pins; and EXPRESSION can be either "0", "1", or a Boolean Expression.

Description:

The I/O Direction field is used to enter a Boolean Expression to designate I/O Control for the bidirectional pins being used.

Direction (Dn or DMn) labels only may be used for pins which have labels other than N/C in the pinlist screen. DMn labels are only applicable to PLC42VA12 designs. You must end the equation with a semicolon (;). The Dn labels are reserved for P-term control of the output pin buffers. See individual PLD data sheet schematics for Dn or DMn assignments.

This information is NOT required for the dedicated inputs (I) or outputs(O, /O). They are only needed when three-stateable pins(B, /B) are defined in the pinlist. Any "I" or "O" assignment in the pinlist screen for a bidirectional (B) pin has priority over information specified in the @I/O Direction Field; therefore, information for "I" or "O" assigned pins in the pinlist screen can be omitted from

[1] PLS151 output enable functions are controlled by the I/O direction and I/O steering functions. Refer to I/O steering Section (10.8) and the PLS151 data sheet.

the @I/O Direction field. If all PLD "B" pins are used as dedicated inputs (I) or outputs (O) as defined in the pinlist screen, then no additional I/O direction information is required and the "@I/O DIRECTION" title can be deleted from the file. If the "@I/O DIRECTION" keyword is left in the .BEE file without any information under it, BLAST will invoke a "WARNING 223" during compilation of the file only to alert the user of incomplete I/O direction specification. This warning does not affect the resultant PLD fuse file but is only to inform the user that the undefined "B–Pin" I/O information is taken from the assignments in the pinlist screen.

The output buffers of a PLC42VA12 are programmably controlled from either a dedicated OE pin (pin 13) or from a single product term (DMn). The default condition is control via pin 13 (OE). Therefore, connection to a product term (DMn) will only occur when DMn expressions are defined under the @I/O DIRECTION header. Remember, the pinlabel function assignments for pins intended to be tri–state controlled must be assigned B, /B, /BD or /BJ in the pinlist screen.

Example:

Example 1.

```
@I/O DIRECTION
     D0 = 1;                 " B0 is an output pin "
     D1 = 0;                 " B1 is an input pin "
     D3 = I0 * i1 * /I5 ;    " Output on B3 is enabled when this
                               term is true"
```

Example 2. For PLHS501 device:

@I/O Direction
DB7=0; "B7 (pin 18) is an input pin"

OE0=1; "O0 & O1 (pins 19 & 21) are outputs"
DB4=Enable; "B4 (pin 15) is an active output when enable input is true"

@I/O Steering
S3=1; "/B3 (pin 40) is an input pin"
S2=0; "/B2 (pin 39) is on output pin"

Note: The above I/O Direction and I/O Steering labels must be specified under the appropriate @I/O Direction and @I/O Steering fields in the Boolean equation entry (.BEE) file for three-state output control in the PLHS501 device. The control terms that enable all the XOR outputs and output only pins are also defined under @I/O Direction (XE0, XE1, XE2, XE3, OE0, OE1, OE2, OE3).

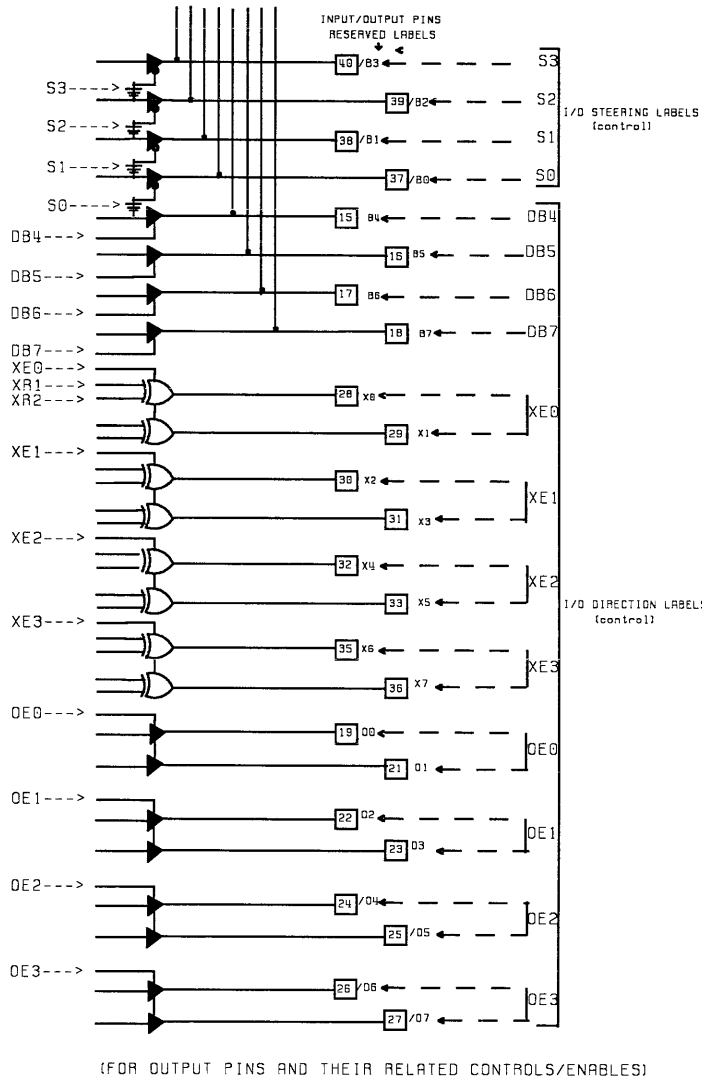


Figure 3–11 PLHS501 Reserved Label Map

3.5.13 @I/O Steering

Devices: PLS151, PLHS501

Format:

LABEL = EXPRESSION ;

LABEL , LABEL = EXPRESSION ;

For PLS151:

LABEL is the label of the three-state control signal of outputs Sn. An EXPRESSION is either "1" (logic high for permanent input) or "Dn" (for dynamic control by Dn signals).

For PLHS501:
An EXPRESSION is either "1" (for a permanent input) OR a "0" (for a permanent output).

Description:

A. The I/O Steering field is used to define the function of the bidirectional pins in a PLS151 design (B0 - B11).

   One may only use labels indicated in the Signetics PLD data manual (Sn), and only for the pins having labels other than N/C. The user must end the equation with a semicolon (;).

B. The I/O Steering field is also used to define four fuse controlled asserted low bidirectional pins in a PLHS501 design (/B0 - /B3).

Example:
For the PLS151:

```
@I/O STEERING

  S0, S2 = 1  ;      " B0 and B2 are Input pins"

  S7, S6, S4 = D1    " B7, B6, and B4 may be Input,"
                     " Output, or dynamically I/O"
                     " depending on the I/O"
                     " CONTROL (D1) definition"
                     " in I/O Direction Field."
```

Note: Since the (Sn) output enable gates are controlled by the control (Dn) terms as specified in the I/O steering field, control signals must also be specified in the I/O direction (Dn) field for the B-pins to become active outputs.

Example for the PLHS501:

@I/O Steering

S0, S2 = 1 ; "these are input pins"

S1, S3 = 0 ; "these are output pins"

Reference Figure 3-11. See also @I/O DIRECTION

3.5.14 @Logic Equation

Devices: All

Format:

```
For active high combinatorial outputs
    LABEL = EXPRESSION ;
For active low combinatorial outputs
    LABEL = /( EXPRESSION ) ;
For registered outputs
    LABEL   : SIDE = EXPRESSION  ;
For PML XOR outputs
    LABEL   : XR1 = EXPRESSION ;
              XR2 = EXPRESSION ;
```

where LABEL is the output pin label, assigned in the pinlist screen. However, for PLS155-9 and PLS179 devices, one must use the complement of the LABEL for a flip-flop. EXPRESSION is a Boolean equation, and SIDE is the input side label to the register, which can be R, S, J, K, D, or T, depending on the PLD selected. XR1 and XR2 are reserved words for the two inputs to each XOR output gate. All Boolean equations must be terminated with a semicolon (;).

Description:

The Logic Equation field is used to define the PLD output functions. The LABEL on the left-hand side of the equation must be one of the labels assigned to an output, bidirectional pin, or internal flip-flop. EXPRESSIONS can consist of multiple levels of parenthesis. BLAST will apply DeMorgan's theorem to translate corresponding Boolean Equations into Program Tables.

If the pinlist function (under "FNC" in pin editor screen) assignment for output "A" is "O" (output) or "B" (tri-stateable output), then the expression: A=/(a*b+c) will be DeMorganized to form the active low output function. If the pinlist function assignment for output "A" is "/0" (active low output) or "/B" (three-stateable output), then the expression: A = /(a*b+c) is mapped as written between the parenthesis and the polarity fuse for output "A" is programmed to implement the

inversion. DeMorganization of the expression may be undesirable in most cases as PLD product term usage increases dramatically with the length of the expression.

Rule of Thumb

For an active-low combinatorial output: Assign "/0" in the pinlist for the output function and use the "LABEL = /(EXPRESSION)" syntax in the @Logic Equation field.

Note that for a combinatorial output the LABEL on the left side of the equation must be the same as the PIN LABEL defined in the pinlist screen. The LABEL on the left side of the equation for a registered output with inverter must correspond to the output of the register and NOT the output pin. Thus, for parts such as PLS159 (with an inverter following the register), if the register output pin label is /REG, then the equation should be written with a LABEL of REG1. See example following for the PLS159. For 16V8/20V8 devices, if the output pin function is a "D", then the associated equation is "LABEL :D=EXP;". If the pin function is "/D" then the equation should be written "/LABEL :D=EXP;". In addition, for the PLC42VA12, if the pin function assignment is "OJ", then the associated equation is "LABEL:J=EXP; K=EXP; or T=EXP;". If the pin function is /OJ, then the associated equation is "/LABEL :J=EXP; K=EXP; or T=EXP;". BLAST will not prevent the user from writing J/K Boolean equations for S/R Register devices (PLS105, 167, 168). This may not invoke an error message during BLAST assembly but will result in unpredictable flip-flop operations in these devices if the condition J=K=1 occurs. Use care when writing Boolean equations with S/R Register-only PLDs.

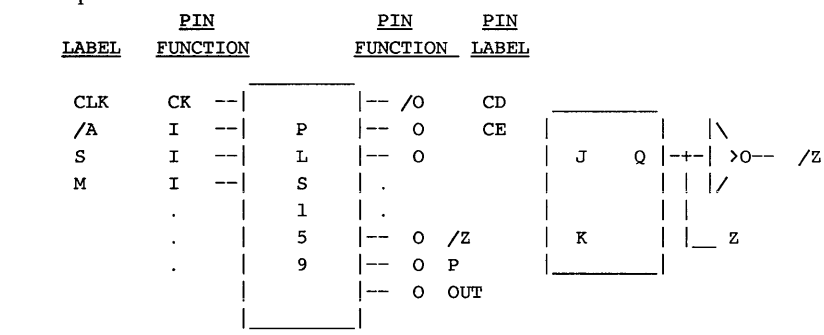
Avoid the use of LABELS "/C0" "/C1" in BLAST unless the user desires the two complement arrays of the PLUS405. Remember that "/C" is also reserved for the complement array of all other PLDs incorporating the feature.

When using the XOR outputs of PML, XR1 and XR2 may be any Boolean expression allowed in BLAST and also bracket notation is permitted. Constants 0 or 1 are allowed but are not needed. If only one input to the XOR output gate is to be active, with the other set to a constant for inversion or straight output, then the user may simply write a normal equation indicating polarity with a slash (/). BLAST will automatically set the constant input to the XOR output gate for the proper output polarity.

```
OPERATORS:          /  |  Complement
                [ ] ()  |  Expression Delimiter
                    *   |  AND (Product)
                    +   |  OR (Sum)
                        |
                        |  Note that the above Operators
                        |  are prioritized in the same
                        |  sequence as above.
                        |
                        |  ex.  X = ( / (A+B) * /D);
                        |  ex.  X = E*[/ (A+B)*D + [F*G]*H] ;
                    ;   |  End of Equation
                        |
                        |  ex.  X = B * C * /D
                        |           + /C * /B;
                        |       Y = /B * /D;
                        |
                    "   |  Comment Delimiter
                        |
                    =   |  Equality
                        |
                    :   |  Assigned to; after the
                        |  low-to-high clock transition for
                        |  register devices.
                        |
                        |  ex.  01 :  J = A * B;
                        |             K = C * /B;
                    ,   |  (Comma) Equivalence
                        |    ex.  X, Y, Z = /A * /B * C;
```

Example:

Example for the PLS159: USER'S PINLIST SCREEN




```
LOGIC EQUATION

 CD = /(/A * S + /A * M)   ;  "ASSERTED LOW COMBINATORIAL OUTPUT'
                              "POLARITY FUSE INTACT"
                              "Note /O function assignment and /(EXP.) syntax
                               of equation"
                              "Note that output label matches pinlist label"

 CE =  (/A * S + /A * M)   ;  "ASSERTED HIGH COMBINATORIAL OUTPUT'
                              "POLARITY FUSE BLOWN"

     Z  : J = S + M * /A   ;  "J/K register output"
          K = S + M * /A   ;  "Note /Z label inversion in pinlist indicating"
                              "inversion from register output to pin"
                              "this is basically a toggle function"
or
     Z  : T = S + M * /A   ; "direct toggle register implementation
                              on Z output"

     /out   : D = /A * S + M  ;    "note register output label slash
                                    is here and not in the pinlist"

     /P  : J = S * M * /A   ;
           K = S + M + /A   ;
           D = CE * /A + M * /S  ;
```

The above flip-flop (P) can be either J/K or D, depending upon the FC line. This feature allows P-term control of the flip-flop function: either "D" or "J/K - toggle" type flip-flop.

3.5.15 @Output Enable

Devices: PLS155, PLS157, PLS159, PLS179,
PHLS16L8, PLUS16L8, PLUS20L8

Format:

LABEL = EXPRESSION ; or
LABEL, LABEL = EXPRESSION ;

where LABEL can be either EA or EB. For the 155, 157, 159 and 179, EXPRESSION may be either 1, 0, or the label of the controlling pin as defined in the pinlist. For the 16L8 and the 20L8, EXPRESSION may be either 1, 0, or a single product term.

Description:

The Output Enable field is used to define the three-state control logic of output pins. The 155, 157, 159, and 179 series of devices have output pins divided into two banks. Each bank consists of two or more output pins whose three-state outputs are commonly controlled by terms EA or EB. These terms may permanently enable, disable or dynamically control (by connection to a device pin) the outputs of a specific bank. For these devices, EA (or EB) should be set equal to a 0 to enable or to a 1 to disable the outputs. If they are set equal to the label (given in the pinlist) of pin 11 for the 155, 157, 159 devices or pin 13 for a 179 then the associated bank of ouputs may be dynamically controlled by the device's output enable pin.

The 16L8 and 20L8 series of devices, have two three-state output pins (i.e., O7 and O0) which do not have feedback into the array. The product terms controlling O7 and O0 are EB and EA respectively. They may be set to a constant (i.e., 0 or 1) or a product term expression of input or bidirectional pin labels. Note that for these devices the sense of EA and EB is different than for the 155, 157, 159 and 179 devices. With the 16L8 and 20L8, when a term is set to a '1' this enables the output and a '0' disables the output.

Refer also to @I/O DIRECTION for information on controlling the bi-directional pins of these devices.

Example:

```
For 155, 157, 159, 179 series devices:
@OUTPUT ENABLE
 EA = 1;       "disabled outputs"
 EB = 0;       "enabled outputs"

@OUTPUT ENABLE
 EA, EB = ENAB; "ENAB is the label in the pinlist to the device's
                 output enable pin.  This is pin 11 for the 155, 157,
                 159 and pin 13 for the 179."

For 16L8 or 20L8 series devices
@OUTPUT ENABLE
 EA = 1 ; "enable output"
 EB = 0 ; "disable output"

@OUTPUT ENABLE
 EA = A*B*C; "single product term dynamically"
 EB = A*D;   "controlling output"
```

3.5.16 @Pinlist

Devices: All

Format:

```
LABEL(1)  1  FUNCTION(1)  comments ;
   .      .       .
   .      .       .
   .      .       .
LABEL(N)  N FUNCTION(N) comments ;
```

where LABEL is the name of up to 12 characters. N is the actual device pin number associated with the assigned LABEL. FUNCTION is the function of the pin.

Description:

This is a section in which to describe the device's pinout and pin functions. All pins of a device MUST be described. If some pins are not connected, then their label should be N/C. This section is automatically created by BLAST for any new design. It may be edited by the user, however it is strongly recommended that the user use the Pin Editor environment of BLAST to make pin assignments. The Pin Editor prompts for allowable pin functions and immediately warns of incorrect usage. It records edit changes in this section.

This section is a new feature of AMAZE 1.7. It replaces the fn.PIN of prior versions. Although AMAZE 1.7 will directly compile designs created with version 1.6 or 1.65, this section may easily be added to these older design files thereby allowing deletion of the .PIN file. To add this section to an older design file, simply invoke the pin editor and retype any pin label. Press ESC and answer 'Y' to the save file prompt. A @PINLIST section will be added to the .BEE file. The fn.PIN file may now be deleted as the @PINLIST section of the .BEE file will now be the source and destination of all subsequent pinlist edits.

Example:

For a 153 series PLA Device

```
@PINLIST

"<--------FUNCTION--------->   <--REFERENCE-->"
"PINLABLE     PINS # PIN_FCT    PIN_ID      OE_CTRL"
N/C            1       I         I0          -  ;
N/C            2       I         I1          -  ;
N/C            3       I         I2          -  ;
N/C            4       I         I3          -  ;
N/C            5       I         I4          -  ;
N/C            6       I         I5          -  ;
N/C            7       I         I6          -  ;
N/C            8       I         I7          -  ;
N/C            9       B         B0          D0 ;
GND           10       0V        GND         -  ;
N/C           11       B         B1          D1 ;
N/C           12       B         B2          D2 ;
N/C           13       B         B3          D3 ;
N/C           14       B         B4          D4 ;
N/C           15       B         B5          D5 ;
N/C           16       B         B6          D6 ;
N/C           17       B         B7          D7 ;
N/C           18       B         B8          D8 ;
N/C           19       B         B9          D9 ;
VCC           20       +5V       VCC         -  ;
```

3.5.17 @Register Load

Devices: PLS155, PLS157, PLS159, PLS179, PLC42VA12

Format:

LABEL = EXPRESSION ;

where LABEL can be LA (lower bank), or LB (upper bank) for the 155, 159 and 179 series. For the PLC42VA12 LABEL may also be LM0 or LM9. See PLD data sheet for bank configuration. An EXPRESSION is either "1" (clocked input from pin), "0" (register loaded from array only), or a single product term for dynamic control.

Description:

The Register Load field is used to define the control logic for direct register loading from the "F" or "M" pins. Only labels indicated as a comment after the header (LA, LB, LM0, LM9) in the BEE file may be used and only for the register banks which are used.

If all the required information is not supplied under this title line, the fuses remain intact which means that the register load function is disabled. If the user selects the input (I) function for a "F" or (RI) for an "M" pin in the pinlist, then the corresponding BANK of flip-flops must be controlled by LA or LB to implement the input F/F function.

It is the users responsibility to program Output Enable and/or the inputs (J, K) to flip-flops so that no interference from other signals arise when directly loading the flops. This means that the bank of registers designated as input (I) or (RI) function in the pinlist screen by the user must have the output disabled (Dn) in the @OUTPUT ENABLE field and the register load function Ln=1 (or controlled) in the @REGISTER LOAD field. For the 155, 159 and 179 series configured as J-K flops, the user must make sure that inputs J & K from the array are low (0)

because a high (1) from either source (array or pins) overrides a low and may conflict with the desired signal from the "F" pin. If the signals occur so that both J & K are 'high' then the flip-flop will toggle. This precaution is not applicable when the flip-flops are 'D-type' because of the inverter between J & K. Also this precaution is not necessary for PLC42VA12 designs, as this device contains circuitry that disables array outputs when directly loading a bank of flip-flops.

Example:

```
@REGISTER LOAD
     LA  = 1;      "ACTIVE, Register bank = input"
                   "All F/F controlled by LA (or LB) are
                    input registers for M-pins".

     LB  = 0;      "IDLE, Register bank = output"
                   "Default condition; bank of F/F are
                    output to the F-pins"
or
     LA  = A * B; "Register bank is loaded from the
                   F or M-pins only when inputs A & B
                   are 'high'"
```

3.6 State Transfer Entry

A Finite State Machine consists of a finite set of states and a set of transitions from state to state that occur on a set of input symbols. Two types of outputs are associated with state machines. Figures 3-12 and 3-13 show these two state machines and their State Diagrams.

1. Outputs associated with the state of the machines (Moore type machine).
2. Outputs associated with the transitions (the input condition and the present state) of the machine (Mealy type machine).

The state machine structure of Signetics PLD devices can be modeled into four different categories as shown in Figure 3-10. Signetics PLD devices are capable of adopting a Moore, Mealy, or Moore/Mealy type state machines. The output of a Mealy type state machine can be asychronous or synchronous with the clock.

Due to the different timing characteristics of the outputs, it was necessary to design a new symbol to distinguish synchronous (or outputs directly from registers) from asynchronous outputs (or outputs from combinatorial logic) to enhance the documentation. The new notation will only affect the registered or synchronous outputs associated with the transition of the machines.

An apostrophe (') was chosen to designate the synchronous/clocked output of a Mealy machine (Mealy output with registers). The apostrophe (') must follow every synchronized Mealy output symbol.

It also must be used on each output of a Moore machine only when the output is taken directly from a state register, and only when the label is used following a 'WITH' statement. If the state vectors directly define the output of a Moore machine and a label only follows a 'THEN' or '::' statement, no apostrophe is required. Refer to the @TRANSITIONS Section for information on 'WITH',

'THEN' and '::' statements. No apostrophe is required for either type of machine when the output is directly from the 'B' pins of a PLS155–159 or PLS179. This is true even if the 'B' output is fed directly from an 'F' pin register. The use of apostrophe's with output labels is shown in Figures 3–14.

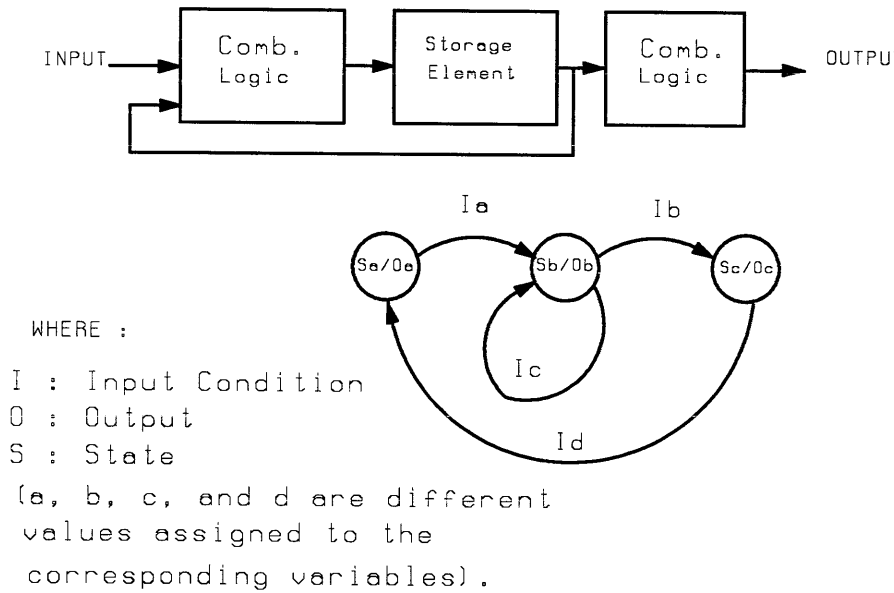


Figure 3–12 Moore State Machine Model

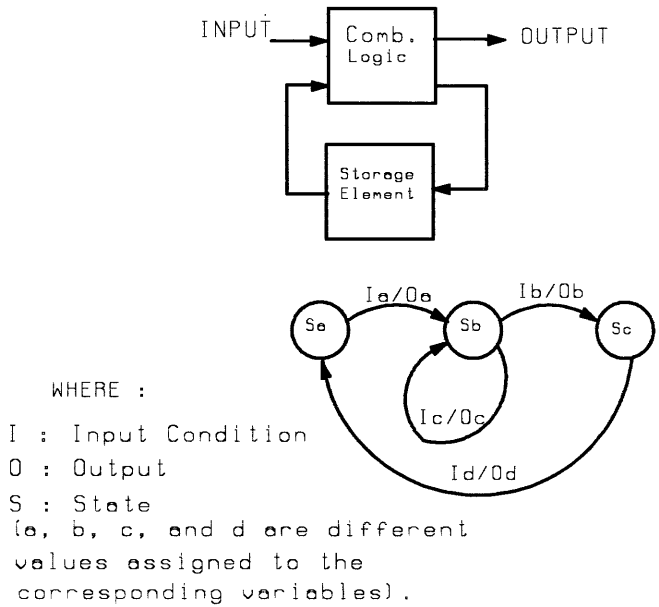


Figure 3-13 Mealy State Machine Model

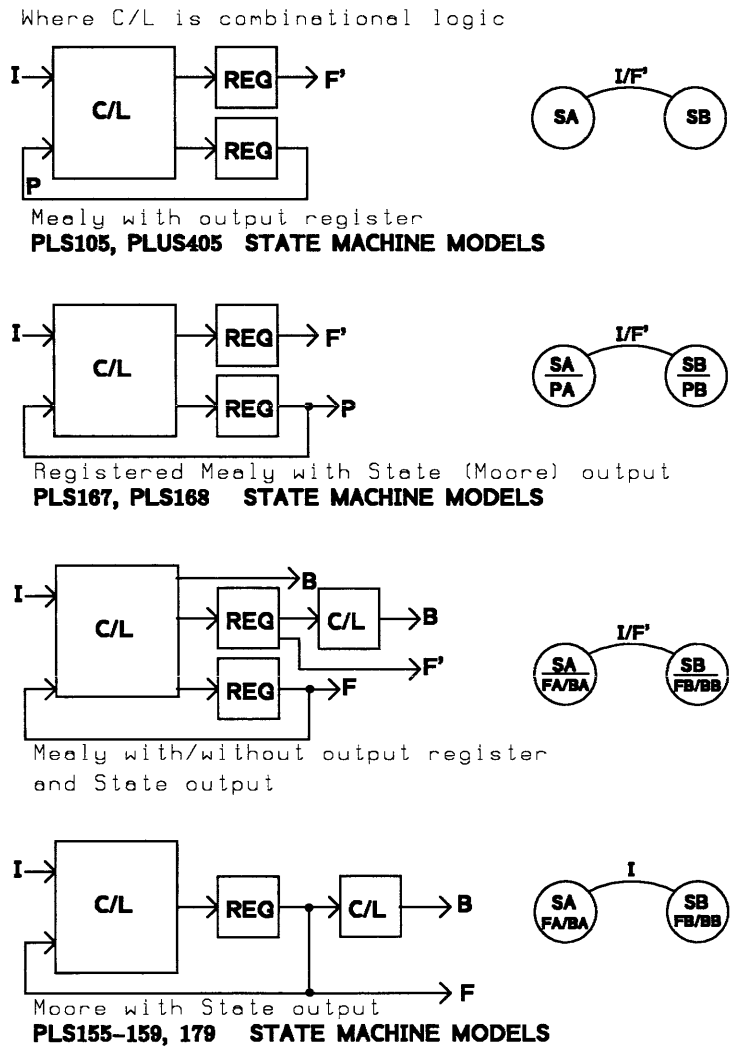


Figure 3-14 State Machine Models of Registered Devices

3.7 How to use State Equation Entry

After choosing Option 3 from the BLAST main menu, BLAST will display a template such as the one shown in Figure 3-12. This file has the SEE extension, such as:

"DESIGN_filename" .SEE

Design information can then be inserted into the template. The information must be entered under the proper title. The Input Vectors must be defined before the transitions, and the Device Selection field must be the first field in the file.

Title lines can be created but not duplicated providing they are not misspelled and the title follows an "@" sign. A warning will be issued for any unexpected, misspelled, or extraneous titles. The information entered on the title line will be ignored by BLAST. A title can be deleted if the corresponding information does not apply to the design, or the feature is not used.

All entries within the State transfer file (fn.SEE) are open formatted. Thus, blank character(s) or comments (with proper syntax) can be inserted at any point of an entry for clarification. No characters can be inserted within the characters of a label (i.e., IN1 is not the same as I N1).

```
@DEVICE SELECTION
@STATE VECTORS
@INPUT VECTORS
@OUTPUT VECTORS
@TRANSITIONS
```

Figure 3-15 An Empty .SEE File

BLAST allows use of more than one PLD for a state machine and also more than one state machine per SEE file. However, the user is required to select the device types, define the Boolean Equations (if used) and the pinlist information, for all of the devices being used in the state machine design. BLAST will automatically

partition the state machine into the designated devices, based upon matching I/O pinlabels between the device pinlist. All the design information (filename(s).BEE, filename(s).STD, and the filename.SEE) must reside in the design file directory. *The filename for a .SEE file may be the same as the filename of the .BEE and .STD files ONLY when there is ONE device per .SEE file.*

The wiring diagram of the design can be defined by using the individual PLD pinlists. The interconnection between devices must be defined by assigning the same label names to the connecting pins. External feedback on the same device can be accomplished (only if internal feedback is not available) by assigning the same label to the input and an output pin.

When using a state equation entry (fn.SEE) the BEE file is necessary only to define flip-flop types (i.e., D or JK. Refer to @Flip-flop mode) and output enable functions. It is not necessary to describe logic equations of functions described by the state equations. If an additional function is to be added to a state machine design and that function is easily described by a logic equation, then the BEE file may, if desired, contain some equations. They will be added to the fusemap generated from the SEE state equations.

Running the compiler with a SEE filename causes all BEE files associated with that SEE filename to be compiled. A simple state machine example is shown in Section 3.9. Additional examples are shown in Appendix A.

The following sections provide a description on the type and format of data to be entered in the State Equation fields.

3.8 .SEE File Headers

@Device Selection

@Input Vectors

@Output Vectors

@State Vectors

@Transitions

3.8.1 @Device Selection

Format:

FILENAME / DEVICE TYPE

where FILENAME is the name of the file where the Boolean and fuse table information exists (i.e., Fn.BEE, Fn.STD). DEVICE TYPE is the PLD device number. (i.e., PLS159)

Description:

The Device Selection field, allows specification of the device(s) to be used for the State Machine Design. If using more than one device for the design, the user must specify all the devices and their appropriate (.BEE) filenames, separated by at least one space or a carriage return. Each filename of multiple device designs must be different than the filename of the SEE file. However, for the case in which only one device is listed under this header, the .BEE file may have the same filename as this SEE file.

Specifications of PLD filenames in the @DEVICE SELECTION field of the State Equation Entry file will automatically include these devices in the state machine design when compiled with BLAST. BLAST will automatically create and handle the filename extensions, so do not include them in any "filename" entries of BLAST.

Example:

```
Example 1.
@DEVICE SELECTION
DLOW/PLS153   timer/PLS159  "2 devices on same line separated by a
space".

Example 2.
@DEVICE SELECTION
SM_PLD1/PLS105  "Pinlist, logic, fusemap of DEV #1)"
SM_PLD2/PLS105  "Pinlist, logic, fusemap of DEV #2)"
SM/PLD3/PLS168  "Pinlist, logic, fusemap of DEV #3)"
```

3.8.2 @Input Vector

Format:

Boolean Notation:

```
VEC_LABEL = EXPRESSION ;
```

Coded sets:

```
[ PIN_LABEL1, PIN_LABEL2 ... PIN LABELN ]
VEC_LABEL1 = NUMBER TYPE ;
VEC_LABEL2 = NUMBER TYPE ;
      .          .       .
      .          .       .
      .          .       .
VEC_LABELN = NUMBER TYPE ;
```

where:

VEC_LABEL	is the desired vector label to be used later in the @TRANSITIONS section.
EXPRESSION	is an expression consisting of PIN_LABELs and AND operations. Where: * is an AND function
PIN_LABEL	is a label given to a specific pin in the pinlist (section 3.3). Or, it may be a label defined in the .BEE file under @INTERNAL SR FLIP_FLOP LABELS section. It may be an input pin label, a bidirectional pin label, or an output pin label with a feedback connection.
NUMBER	is a binary, hex or octal number. There will be a direct correspondence between PIN_LABEL pins and this value. PIN_LABEL is the most significant bit and PIN_LABELN is the least significant. When using binary notation: 1 = 'high' 0 = 'low' – = 'don't care'
TYPE	defines the base of NUMBER. It may be in upper or lower case. Where: O is for octal H is for hexadecimal B is for binary

Description:

Input conditions may be assigned vector labels (VEC_LABEL) to be used in the @TRANSITIONS section. VEC_LABELS must start with an alpha character. A slash (/) is not allowed. If the negation of a vector is desired, then a new vector must be defined describing this new input condition. Pin labels must be used as they were defined on the pinlist. Both Boolean and Coded set formats may be used in this field. However, a coded set is terminated by another coded set or Boolean equation. Refer to Example 1.

For octal and hexadecimal assignment of a VEC_LABEL, the leftmost digits are truncated only if they are zeros. If the leftmost digits are not specified in the octal or hex NUMBER then they will be assigned don't cares. Refer to Example 2. and Example 3.

Binary format requires the number of digits used in NUMBER to equal the number of PIN_LABELs. Refer to Example 2.

Example:

Example 1. Combining Boolean Expression with Coded Set Format

```
@INPUT VECTOR
[b1, b2]
in1 = 11b;
in2 = 0-b;
in3 = /i1 * i15 + 12
in4 = -0b; "This vector is ignored"
```

The last vector "in4" will not be considered part of the [b1, b2] grouping, since it is separated by the Boolean equation of "in3".

Example 2. Right Justification of Octal and Hex NUMBERS

```
@INPUT VECTOR
[a3,a2,a1,a0]
inp1 = 7h;
inp2 = 3o;
```

This is the same as:

```
@INPUT VECTOR
[a3,a2,a1,a0]
inp1 = -111b;
inp2 = --11b;
```

Example 3. Invalid Hex and Octal NUMBERS

```
@INPUT VECTOR
[a3,a2,a1,a0]

inp1 = 8h;   "This is invalid since 8 hex requires
             four bits and only three (a2,a1,a0)
             are separated."
inp2 = 11o;  "This is also invalid since 11 octal
             requires four bits."
```

3.8.3 @Output Vector

Format:

Boolean Notation:

```
VEC_LABEL = LIST;
```

Coded sets:

```
[ PIN_LABEL1, PIN_LABEL2, ... PIN_LABELN ]
VEC_LABEL1 = NUMBER TYPE;
VEC_LABEL2 = NUMBER TYPE;
     "          "       "
     "          "       "
     "          "       "
VEC_LABELN = NUMBER TYPE:
```

where:

VEC_LABEL	is the desired vector label to be used in the @TRANSITIONS section. An apostrophe is required following the label if the outputs are direct from a register.
LIST	is a list of output pin labels separated by comas (,). An apostrophe is required following each output pin label if that pin is a direct output of a register.
PIN_LABEL	is a label given to a specific output pin or bidirectional pin in the pinlist (section 3.2.1). An apostrophe is required after the label if the output is obtained directly from a register.
NUMBER	is a binary, hex, or octal number. There will be a direct correspondence between PIN_LABEL pins and this value. PIN_LABEL1 is the most significant bit and PIN-LABELN is the least significant. When using binary notation: 1 = 'high' 0 = 'low' – = 'don't care'
TYPE	defines the base of NUMBER. It may be in upper or lower case. where: 0 is for octal H is for hexadecimal B is for binary

Description:

Outputs may be grouped and assigned vector labels to be used in the @TRANSITIONS section. VEC_LABELS must start with an alpha character. A slash (/) is not allowed. If the negation of a vector is desired, then a new vector must be defined describing this new input condition. Pin labels must be used as they were defined in the pinlist. Registered outputs may not be grouped with outputs from combinational logic.

Both Boolean and Coded Set formats may be used in this field. However, a coded set is terminated by another Coded Set or Boolean Expression. Refer to Example 1.

For octal and hexadecimal assignment of a VEC_LABEL, the leftmost digits are truncated only if they are zeros. If the leftmost digits are not specified in the octal or hex NUMBER then they will be assigned don't cares. Refer to Example 2 and Example 3.

Binary format requires the number of digits used in NUMBER to equal the number of PIN_LABELs. Refer to Example 2.

Example:

Example 1. Combining Boolean Expression with Coded Set Format

```
@OUTPUT VECTOR
[b1,b2]
out1 = 11b;
out2 = 0-b;
out3 = /i1, i15, i2   "Note labels separated by commas."
out4 = -ob;           "This vector is ignored."
```

The last vector "out4" will not be considered part of the [b1,b2} grouping, since it is separated by the Boolean equation of "out3".

Example 2. Right Justification of Octal and Hex NUMBERS

```
@OUTPUT VECTOR
[a3,a2,a1,a0]
out1 = 7h;
out2 = 3o;
```

This is the same as:

```
@OUTPUT VECTOR
[a3,a2,a1,a0]
out1 = -111b;
out2 = --11b;
```

Example 3. Invalid Hex and Octal NUMBERS

```
@OUTPUT VECTOR
[a2,a1,a0]
out1 = 8h;  "This is invalid since 8 hex requires four
            bits and only three (a2,a1,a0) are
            specified."
out2 = 11o; "This is also invalid since 11 octal
            requires four bits and only three are
            specified."
```

Example 4. Registered outputs

```
@OUTPUT VECTOR
[reg1, reg2]

out3' = 11b;
out2' = 10b;
out1' = 01b; "Note apostrophe in vector label."
```

3.8.4 @State Vector

Format:

Boolean Notation:

```
VEC_LABEL = EXPRESSION;
```

Coded sets:

```
[ PIN_LABEL1, PIN_LABEL2, ... PIN_LABELN ]
VEC_LABEL1 = NUMBER TYPE;
VEC_LABEL2 = NUMBER TYPE;
     "          "       "
     "          "       "
     "          "       "
VEC_LABELN = NUMBER TYPE:
```

where:

VEC_LABEL	is the desired vector label to be used later in the @TRANSITIONS section.
EXPRESSION	is a Boolean Expression consisting of of PIN_LABELs and AND operations. where: * is an AND function
PIN_LABEL	is a label given to a specific output pin with a feedback connection in the pinlist (section 3.2.1). Or, it may be a label defined in the .BEE file under @INTERNAL (SR) FLIP-FLOP LABELS section.
NUMBER	is the binary, hex, or octal number. There will be a direct correspondence between PIN_LABEL pins and this value. PIN_LABEL1 is the most significant bit and PIN_LABELN is the least significant. When using binary notation: 1 = 'high' 0 = 'low' - = 'don't care'
TYPE	defines the base of NUMBER. It may be in upper or lower case. where: O is for octal H is for hexadecimal B is for binary

Description:

State register outputs may be assigned vector labels to be used in the @TRANSITIONS section. VEC_LABELS must start with an alpha character. A slash (/) is not allowed. If the negation of a vector is desired, then a new vector must be defined describing this new input condition. An apostrophe is not required because state vectors always are outputs of registers.

Both Boolean and Coded Set formats may be used in this field; a coded set is terminated by another Coded Set or Boolean Expression. Refer to Example 1.

For octal and hexadecimal assignment of a VEC_LABEL, the leftmost digits are truncated only if they are zeros. If the leftmost digits are not specified in the octal or hex NUMBER then they will be assigned don't cares. Refer to Example 2 and Example 3.

Binary format requires the number of digits used in NUMBER to equal the number of PIN_LABELs. Refer to Example 2.

State registers may be used in only one state vector grouping. Refer to Example 4.

Example:

Example 1. Combining Boolean Expression with Coded Set Format

```
@STATE VECTOR
[b1,b2]
st1 = 11b;
st2 = 0-b;
st3 = /i1 * i15, i2
st4 = -ob;   "This vector is ignored."
```

The last vector "st4" will not be considered part of the [b1,b2] grouping, since it is separated by the Boolean equation of "st3".

Example 2. Right Justification of Octal and Hex NUMBERS

```
@STATE VECTOR
[a3,a2,a1,a0]
st1 = 7h;
st2 = 3o;
```

This is the same as:

```
@STATE VECTOR
[a3,a2,a1,a0]
st1 = -111b;
st2 = --11b;
```

Example 3. Invalid Hex and Octal NUMBERS

```
@STATE VECTOR
 [a2,a1,a0]
 st1 = 8h;   "This is invalid since 8 hex requires four
             bits and only three (a2,a1,a0) are
             specified."
 st2 = 11o;  "This is also invalid since 11 octal
             requires four bits and only three are
             specified."
```

Example 4. State registers may be used in only one state grouping.

```
@STATE VECTOR
 [q0,q4,a1,q7]
 state_00 = 0h;
 state_02 = 2h;
 state_04 = 9h;
 state_14 = eh;

 [q3,q5]
 st_0 = 00b;
 st_1 = 01b;
 st_3 = 11b;
```

3.8.5 @Transitions

Format:

'While-if-then-else':

```
WHILE [ present state ] : [ nonregistered Moore output ]
   IF  [ input1 ] THEN [ next state ]
   IF  [ input2 ] THEN [ next state ] WITH [ Mealy/Moore output ]
         .                .                .
         .                .                .
         .                .                .
   IF   [ inputN ] THEN [ next state ] WITH [ Mealy/Moore output ]
 ELSE  [ next state ] WITH [ Mealy/Moore output ]
```

'While-case-endcase':

```
WHILE [ present state ] : [ nonregistered Moore output ]
  CASE
       [ input1 ] :: [ next state ]
       [ input1 ] :: [ next state ] WITH [ Mealy/Moore output ]
          .              .                .
          .              .                .
          .              .                .
        [ input1 ] :: [ next state ] WITH [ Mealy/Moore output ]
 ELSE  [ next state ] WITH [ Mealy/Moore output ]
ENDCASE
```

Where:

PRESENT STATE — is Pin_label1 * Pin_label2 * ...Pin_labelN or Vec_label1 and where PIN_LABELS are state register output pin labels. VEC_LABEL is a state vector label.

NONREGISTERED MOORE OUTPUT — is Label1 * Label2 * ... LabelN and where LABELS are nonregistered output pin labels or nonregistered output vector labels. Therefore, " : [nonregistered Moore output]" is applicable only to the "B" outputs of PLS155-159 and PLS179 devices and the PLC42VA12 outputs configured as "O" or "B" functions. If this type of output is not used, or does not change from the previous state, and JK or RS flip-flops are used, then it is not necessary to put the colon and Moore output after the WHILE statement. This output value will appear on the device pins upon entry to the associated state.

INPUT	is Pin_label1 */+ Pin_label2 * ... Pin_labelN or Vec_label1 and where PIN_LABELS ar internal node labels, pin labels or complement array, VEC_LABEL is an input vector label, output vector label or a state vector label. A slash (/) may be used with this statement only to negate input, output, or internal register labels. A slash may not precede a vector label. If negation of a vector label is desired, then a new vector must be defined for that input condition. Empty brackets "[ ]" may be used to signify an always or don't care condition.
NEXT STATE	is Pin_label1 * Pin_label2 * ... Pin_labelN or Vec_label and where PIN_LABELS are registered output pin labels. VEC_LABEL is a state vector label. The change of state will occur only if the INPUT condition is true.
MEALY/MOORE OUTPUT	is Label1, Label2, ... LabelN and where LABELS are registered or nonregistered output vector labels. This output will occur only if the INPUT condition is true. Any registered output pin label or output vector labels must be followed by an apostrophe.

Description:

This section is where previously described pin labels and/or vectors are used to describe the functions of one or more state machines. Two formats - the "while-if-then-else" and the "while-case-endcase" are defined. Both formats perform the same function and which one to use is only a matter of personal preference. They may, if desired, both be used in a single .SEE file.

Note that the ELSE statement refers to complement array usage only. This feature is not available for devices which do not have a complement array. More than one ELSE statement may occur within the state machine transitions. Because of complement array usage, this statement may cause a longer propagation delay than through the AND/OR array. An alternative is to use another IF–THEN statement as shown in Example 2. However, the number of product terms may be much larger using IF–THEN instead of the complement array.

Example:

Example 1.

```
@TRANSITIONS
while [st_null]
    if [i7] then [s0]
while [s01] : [ready]
    if [i0-] then [s7] with [out1']
while [s7]
    else [s10] with [/a3',a2']
while [s8] : [busy]
    case
        [i4]::[s0]
        [i3]::[s7]
        [i2]::[s16] with [out2']
    else [s9]
  endcase
while [s9]
    if [i4] then [s8]
    if [i3] then [s0]
    if [i2] then [s7] with [out3']
while [s16]
    if [i1+i2+i3] then [s0] with [out5',out4']
```

Example 2.

```
          OPTION 1                          OPTION 2
  Using Complement Array          Not Using Complement Array

  WHILE [ S0 ]                    WHILE [ S0 ]
     IF [ I1 ] THEN [ S1 ]           IF [ I1 ] THEN [ S1 ]
     IF [ I2 ] THEN [ S3 ]           IF [ I2 ] THEN [ S3 ]
   ELSE [ S2 ]                       IF [ /(I1 + I2 ] THEN [ S2 ]

  WHILE [ S0 ]                    WHILE [ S0 ]
   CASE [ I1 ]  ::  [ S1 ]         CASE [ I1 ]  ::  [ S1 ]
        [ I2 ]  ::  [ S3 ]              [ I2 ]  ::  [ S3 ]
   ELSE [ S2 ]                          [/(I1 + I2 ]  ::  [ S2 ]
ENDCASE                         ENDCASE
```

Remember that the number of product terms used by Option 2 may be much larger than Option1.

3.9 State Machine Example

Consider a simple traffic light controller for a cross-section as shown in Figure 3-13. The notation used in BLAST will always assume that registers (J/K or S/R) will hold their previous values if they are not specified in the transition. Thus, the user must define transitions according to the type of flip-flops (J/K, S/R or D) being used in their design. Figure 3-17 show three state diagrams for the same design using different types of flip-flops. Also, a BLAST implementation of diagram A is shown in Figures 3-19 and 3-20. The responsibility of providing proper flip-flop control information (i.e., FC and/or Mx in .BEE file) is left to the user.

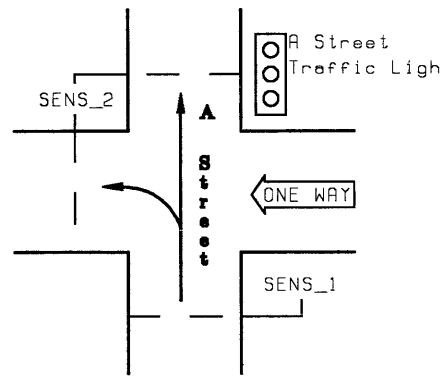


Figure 3-16 A Traffic Intersection

Initially, the A street traffic light is red. Any car in the A street that approaches the intersection has priority over the cross traffic. It will set the SENS-1 signal, which will turn the traffic light to green. When the car crosses the intersection (detected by the SENS-2 signal) and no car behind the first has set SENS_1, the

light will turn yellow a clock time after the last car has crossed SENS_1. The next clock will turn the light red, provided no car has tripped SENS_1. If one has, then the light will turn green from yellow.

Figure 3–17 shows three different types of transition diagrams for a Mealy machine with registered outputs depending on the type of registers being used in implementing the above example.

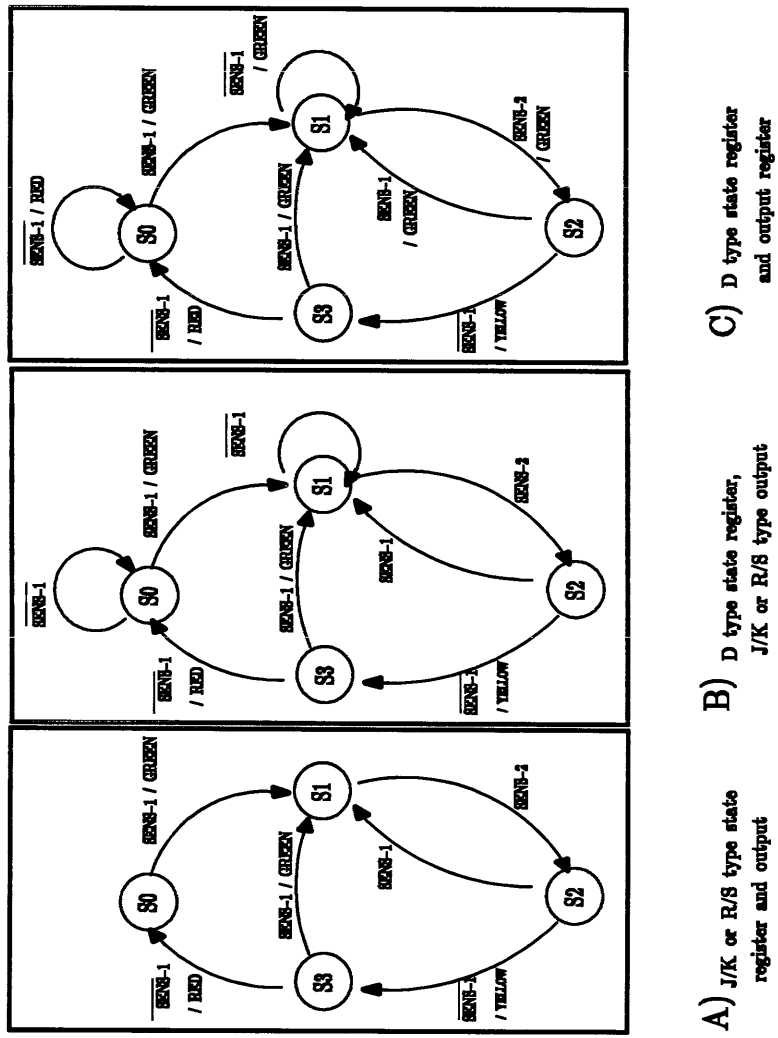


Figure 3–17 Three versions of a traffic light controller

```
File Name :  TRAFFIC
Date      :  11/14/1988
Time      :  14:25:55

@DEVICE TYPE
PLS159
@DRAWING
@REVISION
@DATE
@SYMBOL
@COMPANY
@NAME
@DESCRIPTION

"<--------FUNCTION--------->   <--REFERENCE-->"
"PINLABEL    PIN #  PIN_FCT   PIN_ID     DE-CTRL"
CLOCK         1       CK        CK         -  ;
SENS_1        2       I         I0         -  ;
SENS_2        3       I         I1         -  ;
START         4       I         I2         -  ;
N/C           5       I         I3         -  ;
N/C           6       /B        B0         D0 ;
N/C           7       /B        B1         D1 ;
N/C           8       /B        B2         D2 ;
N/C           9       /B        B3         D3 ;
GND          10       0V        GND        -  ;
N/C          11       /OE       /OE        -  ;
/GREEN       12       /O        F0         EA ;
/YELLOW      13       /O        F1         EA ;
/RED         14       /O        F2         EA ;
N/C          15       /B        F3         EA ;
N/C          16       /B        F4         EB ;
N/C          17       /B        F5         EB ;
Q1           18       /O        F6         EB ;
Q0           19       /O        F7         EB ;
VCC          20       +5V       VCC        -  ;

@COMMON PRODUCT TERM "CPT_label = (expression)"
@COMPLEMENT ARRAY "/C =(---) ;"
@OUTPUT ENABLE " EA, EB "
"permanently enable all outputs"
 EA = 0 ;
 EB = 0 ;
@I/O DIRECTION " D0 .. Dn "
@FLIP FLOP CONTROL " FC = --- ; "
@REGISTER LOAD " LA, LB "
```

```
@ASYNCHRONOUS PRESET/RESET " RA, RB, PA, PB "
@FLIP FLOP MODE  "M0 .. MN "
"set flip-flops used to J/K"
M0,M1,M2,M6,M7 = 1 ;
@LOGIC EQUATION
"no logic equation as all design information is in TRAFFIC.SEE"
```

Figure 3-18 Traffic Light BEE File

```
File Name:  TRAFFIC
Date:  11/14/1988
Time:  14:26:37

@DEVICE SELECTION
traffic/pls159
@STATE VECTORS
[Q1,Q0]
null = --b;
s0   = 00b;
s1   = 01b;
s2   = 10b;
s3   = 11b;
@INPUT VECTORS
@OUTPUT VECTORS
[/green,/yellow,/red]
grn_light' = 011b;
yel_light' = 101b;
red_light' = 110b;
@TRANSITIONS

while [null]
if [start] then [s0] with [red_light']

while [s0]
if [sens_1] then [s1] with [grn_light']

while [s1]
if [sens_1 ] then [s2]

while [s2]
if [sens_1] then [s1]
if [/sens_1] then [s3] with [yel_light']

while [s3]
if [sens_1] then [s1] with [grn-light']
if [/sens_1] then [s0] with [red_light']
```

Figure 3-19 Traffic Light SEE File

3.10 EDIT SCHEMATIC

This section of BLAST invokes a batch file which can be edited to invoke the user's external schematic capture package, such as OrCAD or FutureNet. The purpose of the batch file is to provide an easy method of invoking the schematic capture package and performing post-processing tasks to generate the required FutureNet format PINLIST file (Fn.FNN) for interface with the STBC (Schematic to Boolean Conversion) module of BLAST. The batch file name must be "EDIT_SCH.BAT".

If this option is selected in BLAST, a default text file appears on the screen with instructions for creating a typical "EDIT_SCH.BAT" file for interfacing OrCAD/SDT to STBC. However, this batch file can be modified to interface any schematic capture package to STBC provided the schematic capture package chosen generates a FutureNet Pinlist format file.

BLAST will pass the drive, directory path and filename to the batch file through a batch file variable (%1). This makes operation of the user's external schematic capture package automatic since the design filenames are passed through the batch file. Refer to the text file "EDIT_SCH.TXT" for an OrCAD interface example (in the installed directory of AMAZE).

In some cases, due to limited system memory, the schematic capture package must be invoked outside the AMAZE/BLAST shell. If you experience DOS error messages such as "Memory Limit" during the execution of "EDIT.SCH.BAT", this is a result of insufficient system memory to run AMAZE plus all of the schematic capture related files during the "Edit Schematic" process. If this kind of error message occurs, then exit AMAZE and run EDIT_SCH.BAT, including the drive and path of your schematic file from DOS.

i.e, EDIT.SCH <drive>: \file_path\schematic_filename.

3.11 Schematic To Boolean Conversion (STBC)

This module can be used, along with a schematic capture package, to enter the logic description in a schematic form and obtain the corresponding PLD fuse map.

The logic diagram must be drawn using a schematic entry software/hardware package provided by other vendors such as FutureNet, OrCAD/SDT, or any package which provides a FutureNet PINLIST file of the schematic drawing.

This module can be used on existing designs, in schematic form, to convert part or all of the discrete logic into a PLD. STBC automatically extracts the required information, regarding the section of the schematic which is specified by the user, and creates the corresponding Boolean equations, e.g., it partitions the schematics' area which is to be place into the PLD given the boundaries of the user.

When using the "Schematic-To-Boolean-Conversion" (STBC) module in BLAST, the PLD pinlist establishes boundaries of the logic from the design schematic to be implemented into the PLD (or PLD's). (The "PLD Pinlist"" should not be confused with the "FutureNet format PINLIST" mentioned above).

STBC will only write Boolean equations for the section of the schematic that uses the same signal names as indicated under the @PINLIST header of the BEE file. That is, the user must assign the same signal labels to the input, output, or bidirectional pins on the schematic, as in the @PINLIST section, or vice-versa. When using an existing design, where the user wants to map a section or the whole design into a PLD, they must make sure that pins and signal labels in the @PINLIST section (fn.BEE) and the pinlist file (Fn.FNN) from the schematic do exist and are identical. Note that a pin label with an overscore or bar "$\overline{\text{PINLABEL}}$" is accepted syntax for many external schematic capture packages but is not a legal definition for a pin label negation in BLAST. Slash (/) must be used for negation

of pin labels in the schematic drawing and in the BLAST pinlist screen (i.e., /PINLABEL).

FutureNet (Strides) or OrCAD SDT can create either a "netlist" or a "pinlist" FutureNet format output file (with default extension of PIN). The "pinlist" format must be specified for use with STBC. In addition, this FutureNet output file must be assigned the same filename as the PLD design in BLAST and must be given a ".FNN" extension (Fn.FNN). FutureNet's pinlist file must be copied to the design file directory before invoking BLAST STBC module. The user must have a completely filled out @PINLIST section of the BEE file and Fn.FNN (FutureNet pinlist) as well as Fn.STD in the design file directory in order for STBC to properly enter equations under the @LOGIC EQUATIONS header in the .BEE file.. Note that Fn.STD is automatically created by BLAST whenever a BLAST pinlist for a new design "Fn" is created by the user.

EXAMPLE: Schematic drawing created with FutureNet, OrCAD/SDT, or equivalent external schematic capture package and corresponding BLAST pin list drawing.

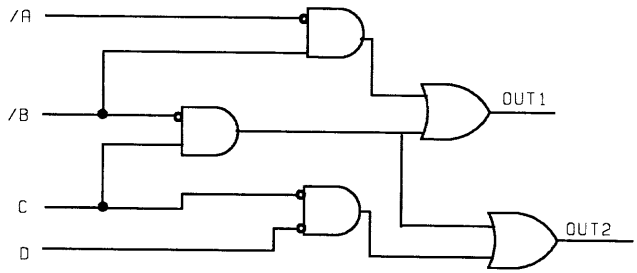


Figure 3-20 Schematic Drawing

```
                    BLAST    Fn.Fnn
                  PINLIST
LABEL   FNC      _______     FNC    LABEL
 /A      I      |       |    O     outl
 /B      I      |   P   |
  C      I      |   L   |    O     out2
  D      I      |   D   |
                |_______|
```

Figure 3-21 BLAST Pinlist Drawing

FutureNet refers to Data I/O FutureNet schematic capture packages: DASH II, DASH III, OR DASH IV. An equivalent schematic capture package capable of generating FutureNet format pinlist files may be used.

3.11.1 How to use STBC

When using this module, the following information files are needed:

AMAZE.CTL	Boolean description of the elements (74XX "starter" library provided with AMAZE).
filename.FNN	Interconnection of the elements (FutureNet format Pinlist) Created by external schematic capture package
PLSnnn.AMZ	PLD architecture, provided in AMAZE

Once these files are obtained (described in the following sections), invoke STBC by selecting option 5 (Schematic To Boolean Conversion) from the BLAST menu. Invoking STBC will cause equations to be entered under the @LOGIC EQUATIONS section of the design's BEE file. BLAST option 6 will then COMPILE the information. The following sequence must be taken to design in a schematic form:

1. Draw the schematic or call up an existing design in a schematic form.* Make sure the design rules specified in the following sections are met. Also make sure that all the elements in the drawing are properly specified in the AMAZE.CTL file. (See Section 3.11). The included AMAZE CTL file describes 74XXX devices and must be edited if 74S, 74LS 74F, 74HCT, etc., devices are used in the schematic.

   *See "EDIT SCHEMATIC" Section 3.10 for use of your external schematic capture package with BLAST.

2. Assign signal names to the nodes/signals which are going into or coming out of the PLD device. This step specifies the boundaries of the logic which is embodied in the PLD device.

3. Create the interconnection description file for the drawing (Fn.FNN). This procedure depends on the schematic entry package being used.

   A. For FutureNet DASH 2, 3, or 4; use the FutureNet "pinlist processor" to generate the pinlist. Rename this file to "fn.FNN" and copy to the design file directory. Do not confuse the FutureNet pinlist file (Fn.FNN) with the @PINLIST section of the BEE file.

B. For OrCAD/SDT Version 3.1 generate the interconnection description using the following three programs:

```
Annotate  filename /m /u
Ercheck   filename
Netlist   filename  filename.FNN FutureNet /s /p
```

(The /P specifies generation of a FutureNet "pinlist" format file.)

4. Exit the Schematic entry module, and invoke BLAST once the filename.FNN file creation has been completed.

5. Select the Pinlist Editor option. Use the same filename as specified in step 3. (fn.FNN) If no BLAST PLD pinlist exists a new set of information will be created. This step selects the PLD device.

6. Use the BLAST Pinlist Editor to specify the signal/pin names, their polarity, and functions for the PLD pins. Use the same labels as specified in the schematic. This step specifies the portion of the schematic being replaced with a PLD. Select output pin polarities to reflect the active polarity of the associated schematic label or signal.

7. Exit the Pinlist Editor. Invoke STBC by selecting option 5 (Schematic To Boolean Conversion) in the BLAST menu.

   STBC will use the information provided, and will enter equations in the BEE file, if no errors are encountered while processing the design. All the errors and warning messages during STBC are placed in the "filename.MSC" file. This file can be reviewed using DOS Command option in the BLAST menu. If errors occur in this step, take action to correct them using the above steps, otherwise continue with the next step.

8. Select the COMPILE option in the BLAST menu. Then select the desired option in the COMPILE menu. In this step a fusemap filename.STD will be created if the logic can be placed into the designated PLD.

   If any SIZE problem occurs, meaning the design will not fit into the requested PLD, try inverting the polaity of the output signal(s) in the Pinlist Editor. For those outputs generating excessive product terms or unusually large equations in the .BEE File and repeat steps 7 and 8. If this does not reduce product term usage, then the design can be reduced in the schematic, by specifying less logic. This can be done by going back to step 2 through 4 and/or step 6. Repeat steps 7 and 8 after any modifications.

   After a successful compilation, a PLD fusemap which will perform the same function as the schematic, will be created.

3.11.2 Schematic Entry Design Restrictions

1. The label (part name or part label) of the logic element(s) i.e., 7403, in the schematic must have an attribute of 3 for FutureNet DASH.

2. Make sure all the input/output signal names (PLD pin label), in the FutureNet DASH schematic, are assigned an attribute of 5.

3. For registers, both Q and /Q can be used in the logic, but only one of them can be used as the PLD output. In other words, the Q and /Q of the same register cannot connect to two different registered outputs of a PLD, since PLD's have one dedicated output pin per register.

4. STBC will place a "?" in the Boolean equations created, if there is an unassigned input or a wire connection problem in the schematic drawing.

5. Make sure no Reserved labels appear as signal names. These labels are specified for every device and they are listed during pinlist editing in BLAST. For PML designs, internal node labels default to INTERNAL (int) (e.g., INTERNAL1, INTERNAL2, etc.). Therefore, INTERNAL (int) are reserved names and may not be used elsewhere. Also avoid the labels Transitions, Vector, State, Input, and Output; these are also used by BLAST.

   However, if planning to use specific features of a device, one must use the pre-assigned name for the signal names as listed below:

   (i.e., /C or /C0, /C1 for Complement Array)

6. STBC supports different types of registers (i.e., J-K, D, R-S, or T), and will automatically configure the registers in the device accordingly. However, only PLD supported register types may be specified.

7. The input signal names for the lines connected to Vcc and Ground must be Vcc and GND respectively. For example, if connecting an input pin of an element to a constant high line (+5 volt), the signal name for that pin must be "Vcc". A constant low input would be assigned the signal name "GND". Remember, the attribute for Vcc and GND must be 5 when using FutureNet..

8. Schematic signal names with a bar on top (for inversion) are not accepted by BLAST. The inversion syntax for BLAST is slash "/".

9. Due to the architecture of some of the PLDs, internal feedbacks without intermediate pin labels are not possible to implement. If using feedbacks in the design, make sure they originate at a PLD pin. In other words, only feedback signals from nodes such as:

   A. An output pin signal
   B. Output of a register
   C. Complement array "/C"

10. STBC will not assign labels to output and input registers used in the design. However, if a register does not have a pin label associated with it, it can be considered as an internal register.

11. If the BLAST pinlist editor does not allow assignment of the same pin labels to implement an external connection, insert a dummy element in the schematic and assign two different signal names to it's input and output.

12. Although XOR gates may be used in a schematic, they will be implemented by STBC in NAND gates. If the output XOR gates of a PLHS501 are to be used, then the .BEE equations will have to be edited. Refer to Section 3.5.14 for the proper equation syntax to use for XOR output gates.

3.12 Schematic Element Description File

AMAZE.CTL describes the functions of logic elements (TTL device library) in the schematic used as input for STBC. In other words, AMAZE.CTL is the file which contains information regarding the Boolean equivalent function of elements used in the schematic. The user is provided with an AMAZE.CTL file containing the functional description of basic SSI, MSI primitives in the TTL library of the schematic capture package used. If using any other elements/library not covered by this file, one can simply add the element, modify the existing element, or create a new file. Each logic element is described in the AMAZE.CTL file as a function of the element's pin numbers or labels. The AMAZE.CTL file is located in the sub-directory STBCLIB located in the AMAZE directory.

The user must make sure that all the elements used in the schematic are properly defined in this file. AMAZE.CTL is an ASCII text file and can easily be edited

with a text editor or word processor such as PC-WRITE included with the AMAZE package. Note that for example, 74LS138 will not be recognized by STBC unless the 74LS138 logic description is written into the AMAZE.CTL file. However, a 74138 description is included in the supplied AMAZE.CTL file and will be recognized without editing the AMAZE.CTL file.
There are four types of information in AMAZE.CTL which are used to describe function of the elements and their pins:

1. Three-stateable elements
2. Register elements
3. Pure combinatorial Boolean Logic without the above elements (i.e., AND,OR), or in general, any elements (decoders, multiplexers, etc.) for which one can write Boolean equations for them.
4. Pin function attribute (i.e., input, output, three-state).

The above information is defined under the appropriate heading/title.

@Combinatorial Logic
(files for simple gates and decoders)

This file is open formatted (spacing between labels, arguments, delimiters, names, ...etc. is not restricted). However, spaces between characters of a label or name is not allowed. For example, "IN0" is not the same as "I N 0", but "IN0=i1*/I2;" is the same as "IN0 = I1 * /I2 ;".

The headings/titles must be as specified (i.e., @Combinatorial Logic). No extra blanks or different spelling is allowed, however, the sequence of the information may be changed. Information on the same line as the heading line is ignored.

3.13 AMAZE.CTL File Headers

@Attribute

@Combinatorial Logic

@Registers

@Tri States

3.13.1 @Attributes

Format:

pin function ; attribute number , attribute number , ... ;

comma (,) is the *attribute number* separator, and a semicolon (;) is the *pin function* separator. *Pin function* can take on one of three values: INPUTS, OUTPUTS, or BIDIRECTIONS. For the appropriate attribute numbers refer to the FutureNet pin attribute number.

Description:

In this section one can define all possible attributes that are used in the schematics/designs. Attributes are used to define the function of pins of the elements (i.e. 7400 input and output pins) in the schematics. Thus, the attributes should be separated into three different categories:

i. Inputs (defined under an "INPUTS :" section)
ii. Outputs (defined under an "OUTPUTS :" section)
iii. Bidirectionals (defined under a "BIDIRECTIONS :" section)

Example:

```
@ATTRIBUTE       " The info. on this line is ignored"
  INPUTS :         23, 24 ;
  OUTPUTS :        21, 25 ;
  BIDIRECTIONS :   22, 26 ;
```

When creating schematics with FutureNet* the user must properly assign the element and signal input/output labels.

The attribute for the element name (i.e., 7404 or 74ls14 must be 3.

The attribute for the signals must be 5.

The attribute for the PLD pin labels in the schematic must also be 5.

* FutureNet refers to Data I/O–FutureNet DASH–II, III, or IV Schematic capture package.

3.13.2 @Combinational Logic

Format:

type name : *output pin name* = *Boolean equation* ;

comma (,) is the *output pin name* separator, and a semicolon (;) is the *type name* separator. A slash (/) before the pin label indicates an Active Low function. Parenthesis and thus multilevels of nesting is allowed in the *Boolean equation* area.

Description:

In this section (under @COMBINATIONAL LOGIC heading), the user must define the logic function of all the elements used in the design and/or library. The logic shall be defined in a Boolean form with the same syntax as described in the previous sections of this manual (under Boolean Equation Entry). In general, any element that has a Boolean function between its inputs and outputs can be defined in this section. The equations must be written in terms of the elements' pin labels.

Example:

@COMBINATIONAL LOGIC

```
7400 : 3 = /(1*2) ,
       6 = /(5* 4) ,
      11 = /(12 * 13),
       8 - /(9*10) ;

74LS55 : 8 = /((1*2*3*4) + (10 *11*12*13)) ;
```

Note: If a 74LS00 is used in the schematic, then a logic description for the 74LS00 gate needs to be added to the AMAZE.CTL file. This can be easily accomplished by using a text editor (such as PC-WRITE) to edit or copy the existing 7400 logic description to the 74LS00 (or simply change the "7400" name to "74LS00". In this case, "7400" will no longer be recognized by STBC in the .CTL file).

3.13.3 @Registers

Format:

type name : pin function designator = pin label (or pin number) ;

A minimum of one space or carriage return is the *pin function designator separator* for the same *register*, colon (:) is the *registers* definition separator, and Semicolon (;) is the *type name* separator. A slash (/) before the *pin label* indicates an Active Low function.

Description:

The registers are defined under @REGISTERS heading, where the user define the name of the element(s), the pin labels, and the function associated with the pins. The following functions are supported in this section:

- o Clock (designated by "C")
- o Asynchronous Preset (designated by "AP")
- o Asynchronous Reset (designated by "AR")
- o Input (designated by "J", "K", "R", "S", "T", or "D")
- o Output (designated by either "Q" for true output, or "N" for complemented output)
- o Synchronous Load (designated by "L").
- o Flip-flop Control (designated by "F").

Example:

```
@REGISTERS

  7474    :  C= 3  D= 2  AR =  1  Q=5  N=6  AP= 4
             C=11  D=12  AR = 13  Q=9  n=8  AP=10 ;

  74107   :  C= /12  J=1  AR=/13
             Q=   3  N=2   K=  4
             C=  /9  J=8   K= 11
             AR=/10  Q=5   N = 6 ;
```

3.13.4 @Tri States

Format:

type name : *pin function designator* = *pin label (or pin number)* ;

A comma (,) is the three-state's component definition separator, and a semicolon (;) is the *type name* separator. A slash (/) before the *pin label* indicates an Active Low function. Example: three-state devices are octal buffers, transceivers, and; the quad buffer 74125.

Description:

The three-stateable elements are defined under @Tri States. Under this heading the name and the pin labels of these elements must be defined such that they match those defined in the schematic. Three types of functions can be associated with every pin:

- o Input (designated by "I")
- o Output (designated by "O")
- o Control Line (designated by "C")

Example:

```
@Tri States     " Information on this line is ignored "

 74240 :  C = /19     I = 17     O =  /3 ,
          C = /19     I = 15     O =  /5 ,
          C = /19     I = 13     O =  /7 , "Note comma (,) used
          C = /19     I = 11     O =  /9 ,  to separate internal
          C = /1      I =  2     O = /18 ,  components (BUFFERS).
          C = /1      I =  4     O = /16 ,  Semicolon (;) ends the
          C = /1      I =  6     O = /14 ,  description)."
          C = /1      I =  8     O = /12 ;

 74241 :  C = 19      I = 17     O =  3 ,
          C = 19      I = 15     O =  5 ,
          C = 19      I = 13     O =  7 ,
          C = 19      I = 11     O =  9 ,
          C = /1      I =  2     O = 18 ,
          C = /1      I =  4     O = 16 ,
          C = /1      I =  6     O = 14 ,
          C = /1      I =  8     O = 12 ;
```

3.14 Compile

The design information will generate a PLD Fuse Map by selecting Option 6 in the BLAST main menu. The COMPILE menu is shown in (Figure 3–22) where two major Compile Options are given, Sum Of Products Equations and Superimposition. When compiling, if any error is detected, the line where the error was found and the previous line are printed or listed, with a marker pointing to a specific column. The marker indicates that the error was found at or to the left of the marker position.

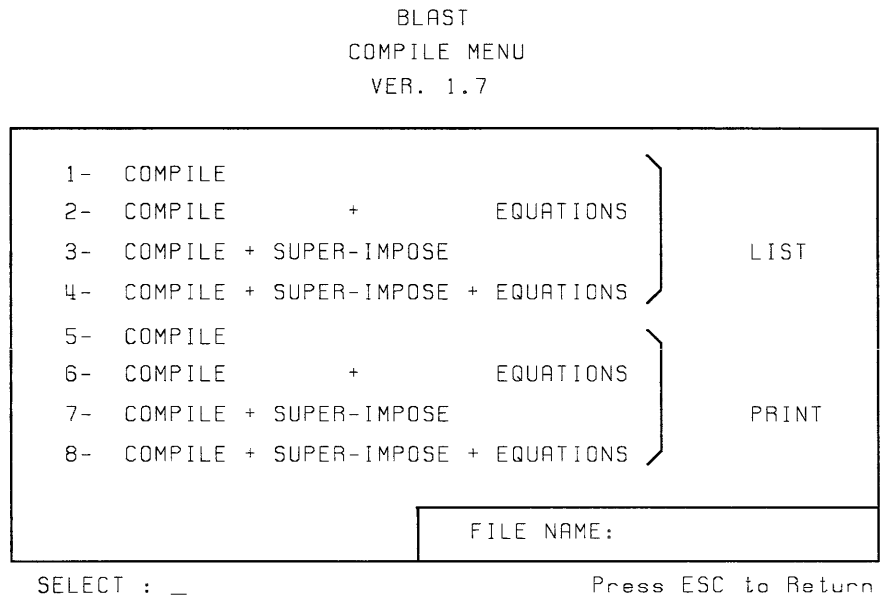


Figure 3–22 Compile Menu

When Compiling a State Equation file, all the information regarding the state machine (all the .BEE, .STD files specified in the Device Specification field of the .SEE file) and the .SEE file must reside in the design file directory.

3.14.1 Sum Of Products

The listing or print-out of the expanded version of the Boolean equation is a Sum Of Products form which may be requested during compilation. This option allows examination of the equation for any possible minimization. Note that BLAST will automatically ignore any logic redundancies, and no logic minimization routine is done during COMPILE.

3.14.1.1 Sum Of Products for PML with Bracket Notation

PML devices, (such as the PLHS501) may use bracket notation which provides a different way of presenting the sum of products format for a particular output equation. Refer to Section 3.15 for information on PML Bracket Notation. Essentially, the portion of the equation inside the bracket is treated (and thought of) as a separate equation. AMAZE may expand what is within the brackets, however, individual product terms within the brackets are not combined with any product terms outside the brackets. Even if the same product term exists both inside and outside the brackets, it will remain split in both places, even after assembling in BLAST. This can be seen in Figures 3-23 and 3-24.

Note that item 1 in both figures points to the same product term both inside and outside the brackets. Items 2 and 3 are the left and right hand side bracket respectively.

```
@DEVICE TYPE
PLHS501
@DRAWING
@REVISION
@DATE
@SYMBOL
@COMPANY
@NAME
@DESCRIPTION
@COMMON PRODUCT TERM
@INTERNAL NODE
@I/O DIRECTION
@I/O STEERING
@LOGIC EQUATION
```

```
                        1                       3
     ↓                  ↓          ↓            ↓
X= (a*b)+[b*c*d+a*e*(f+g+b*h)+(a+b*g*h+e)*(b+c*d+a*e)]+(a+c+d+e)*(f+g*h+b)+a*b+c*g;
         ↑
         2
```

Figure 3–23 Test Case Showing PML SOP

```
Displaying equations in SUM of PRODUCTS form....
                        1
      ↓                                          ↓
X= (a*b) +[(b*c*d)+(a*e*f)+(a*e*g)+(a*b*e*h)+  (a*b) +(a*c*d)+(b*g*h)
          ↑                                    3
          2                                    ↓
+(b*c*d*g*h)+(a*b*e*g*h)+(a*e)+(b*e)+c*d*e)]+ a*f + a*g*h + c*f +c*g*h + b*c +
  d*e*f + d*e*g*h + b*d*e + a*b*c*g;
```

Figure 3–24 Compiled Result

The fuse table resulting from this one complex equation is shown in the FTE section 4.3.2 of this manual.

3.14.2 Superimposition

If using a programmed PLD device which contains an old fuse pattern, and the same device is to be used with the new design or a modification of the existing design, then the Superimposition option may be employed. BLAST will then try to overlay the modifications in the new design onto the previous pattern. BLAST also deletes any product terms on the old pattern which would interfere with the new design. This allows iterative design and programming, adding and testing new functions to the PLD on a step-by-step basis.

Use the Superimposition option only when the design changes are complete, and the pattern existing in the "fn.STD" file matches to the pattern in the physical device. This is usually true when the device is programmed iteratively, since the most recent additions or changes are presently programmed into the PLD. It is possible to reuse a PLD for a completely new design unrelated to its existing programmed function, given that the inputs/outputs, polarities and control functions are unchanged. This can be accomplished by uploading (via programmer) the fusemap of the existing programmed PLD, assigning the same filename to the uploaded fusemap as used in BLAST for the new design, and invoking "superimpose" to overlay the new changes over the existing fusemap. Superimpose will display the message: "Old Device" to indicate the present device may be used; the "New Device" message indicates a lack or conflict of p-terms or functions and a unprogrammed PLD must be used.

3.15 Print

A hard-copy of the design work may be obtained by selecting Option 7 (PRINT) in the BLAST main menu. The PRINT menu will be displayed as shown in Figure 3-25. The files may be printed individually or as a complete package. When option 6, (ALL THE DESIGN INFORMATION), is chosen, a copy of all the BLAST information, and the simulator .LOG file (if they exist in the design file -ardirectory) will be printed. Selection 4, Edit Schematic, calls the "EDIT.SCH.BAT" file to allow printing the schematic via the utilities in your External Schematic Capture Package, if used.

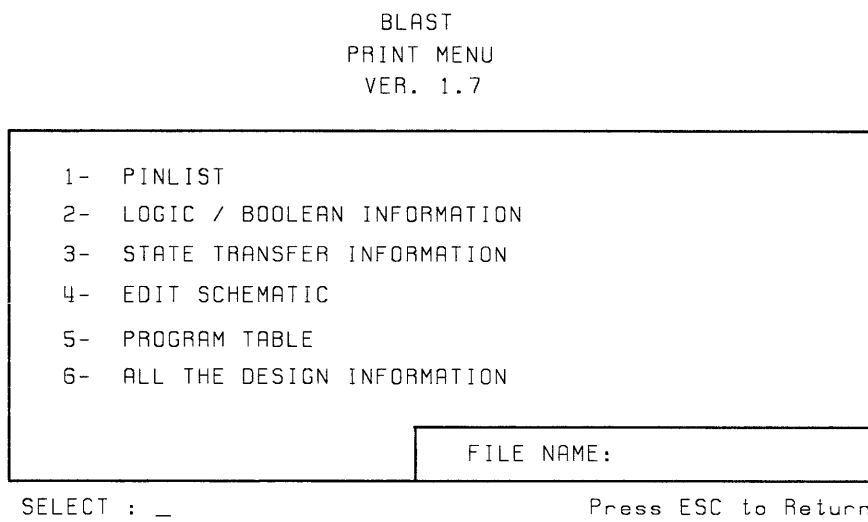


Figure 3-25 Print Menu

3.16 Bracket Notation for the PML Family of Devices

Bracket notation allows the logic designer to implement Boolean equations to any "depth" (stacking gates several levels deep – up to approximately 73 gate levels at maximum).

This allows optimization of the PML structure by making the appropriate tradeoff between the following two directions:

1. Minimize gate delay by use of a two gate level implementation of a Boolean Expression, (either the Sum of Products or Product Of Sums), or
2. Reduce the number of Product Terms used, by adding gate levels appropriately. By actually adding an additional gate level, or several, a reduction may be achieved in the number of total product terms required for a particular implementation of a particular Boolean Expression. This is particularly appropriate for multiple function synthesis.

Example 1:
Implement the following equation in the PLHS501:

$Z = a*b*c*d + \overline{b}*\overline{d}*e + (a+b+c)*(d+e+f)+c*d*e*f$

Let X = (a+b+c)*(d+e+f) ; requires 2 terms

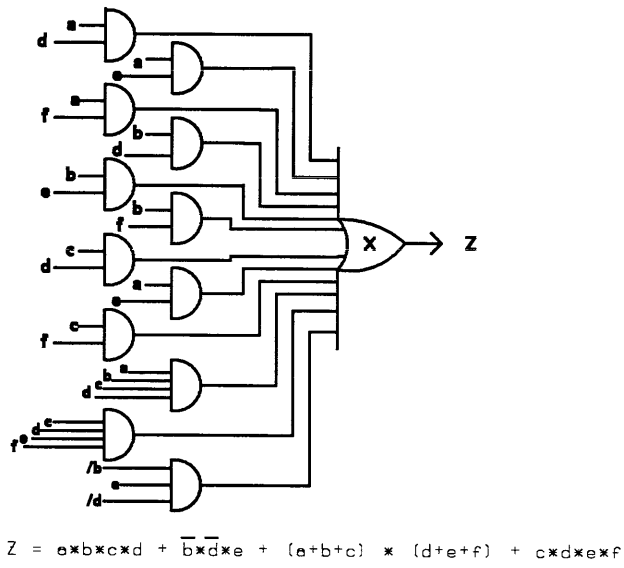
Then X also is equal to:

X = a*d + a*e + a*f + b*d + b*e + b*f + c*d + c*e + c*f ;
requires 9 terms

Therefore,

$Z = a*b*c*d + \overline{b}*\overline{d}*e + X + c*d*e*f$;

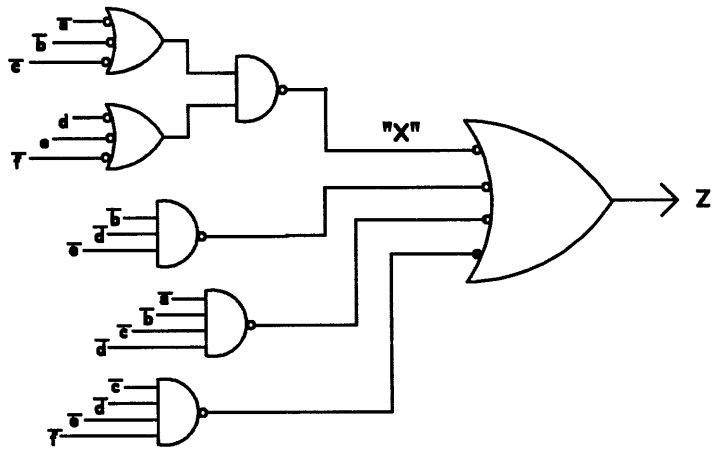
One typical Sum of Products two level implementation of this expression results in thirteen gates being used (nine gates with two inputs, two gates with four inputs, one gate with three inputs and one gate with 12 inputs (Figure 3-26).



$Z = a*b*c*d + \overline{b}*\overline{d}*e + (a+b+c) * (d+e+f) + c*d*e*f$

Figure 3-26 Two Level SOP Configuration

If the first implementation with an additional gate level is allowed (and the associated additional gate delay), then the following circuit results in a total of seven gates only. See Figure 3-27.




```
Z = a*b*c*d + b*d*e +[(a+b+c)*(d+e+f)] + c*d*e*f ;
```

Figure 3-27 Bracket Controlled Multilevel Configuration

This implementation results in three three-input gates, three four-input gates and finally one two-input gate. It also exploits the available complemented inputs available through Signetics input architecture.

The net conclusion is that the additional gate level in this example (which adds one gate delay) reduces the number of product terms used from thirteen down to seven.

An additional way to view the capabilities of the bracket notation are shown in the following example:

Example 2:

```
Z = ((a+b) * (c+d) * (e+f)) * ((g+h+i) * (j+k+l))
```

Expansion of Z to a two level structure would result in -

(8 P–terms) X (9 P–terms) = 72 P–terms

By judicious bracketing this may be partially expanded as follows:

```
Z = [(a+b) * (c+d) * (e+f)] * [(g+h+i) * (j+k+l)]
```

This will result in 19 terms used when fully expanded. Careful use of brackets permits a designer to trade off gate depth (or speed) with the number of product terms used.

Brackets do not prevent AMAZE from expanding what is within the brackets. Brackets prevent AMAZE from combining what is within the brackets with those terms outside of the brackets. Therefore, for the most control over the circuit that is actually fused within the device, write equations for a NAND gate implementation and use brackets around the product terms.

PTE

Users Guide

4. Program Table Editor

4.1 Description

PTE is a special editor for PLD Program Tables. This module allows creation, modification, or printing a PLD Standard file in a format similar to the PLD Program Table shown in Signetics PLD Data Manual. It allows comments in the header area of the table. An on-line HELP display is provided during the operation of this module.

During the Program Table editing session, the entered information is checked for validity and the cursor will not move to the next position when a non-valid character is entered. The output of PTE is a PLD standard (.STD) file which can be used by other AMAZE modules.

PTE supports the following editing functions:

- [→][↑][←][↓]: Moves the cursor position.
- RETURN : Moves the cursor to the beginning of the next line.
- TAB : Moves the cursor to the next functional field.
- SHIFT TAB : Moves the cursor to the previous functional field.
- PgUP : Moves to the previous page of the Table, if any.
- PgDn : Moves to the next page of the Table, if any.
- Home : Moves to the first page of the Table.
- End : Moves to the last page of the Table.
- Ctrl + A : Aborts the editing session without saving the information.
- Ctrl + B : Blanks the present line of table by changing all the fuses into 0 or A state. This command changes the fuses to BLOWN (PLD virgin) state for high speed (PLHS) parts.

- Ctrl + D : Changes the fuses in the present line into - or . state. This means UNBLOWN for High Speed (PLHS) parts and BLOWN for other (PLS or PLC) parts.
- Ctrl + E : Saves the present information and Exits the editing session.
- Ctrl + P : Prints the Program Table (make sure printer is on-line).
- Ctrl + O : Deletes the OR section of the product term. This can improve speed performance of some PLD's; see Data Sheet(s)).
- Ctrl + R : Refreshes the present screen if any system messages have appeared on the screen.
- Cntl + S : Save the file.
- Ctrl + T : Returns to the COMMENT section of the file.
- ESC : Exits PTE and prompts the user whether or not to save the edit changes (duplicates the CTRL+A and CTRL+E functions).

PTE will NOT display the product terms which are in the default state (–) for High Speed products (i.e. PLHS473), or in the virgin (0) state for all the other types of products (i.e. PLS153, PLC473, or PLS105).

4.2 How to Use PTE

When PTE is invoked, it requests the name of the file to be edited. PTE will then look for "fn.STD", where fn is the user defined filename, in the design file directory. If such a file does not exist, PTE requests if a new file is to be created. For new designs a menu of all the available devices is displayed, and the user is asked to select the desired device.

For new designs, the following section will be displayed:

```
Cust/project -
Date         -
Rev/I. D.    -
Comments:
```

The cursor will be positioned at the top of the screen. The user may skip through fields by pressing the RETURN key. Note that the RETURN key will keep any information entered previously. A list of available editing commands may be viewed by pressing the asterisk (*) key. To exit this mode, press ESC or position the cursor on the last line of the comment field (line immediately above bar) and press RETURN.

The screen will be refreshed and the following prompt will appear under the header/comment section:

Continue to Program Table? (Y:edit table; N:edit header; Esc:exit) [Y]

For existing designs, the program will skip to this point with information previously entered displayed above in it's appropriate field.

An affirmative response, or (RETURN as shown in the bracket above), will invoke the Program Table editing session. Pressing [N] will position the cursor at the top of the current screen to re-edit the header/comment section. If at any point during the program table editing one must return to this section, simply press [Ctrl] and [T] keys together.

During the PTE session, the Program Table heading and a maximum of 16 product terms and/or control terms are displayed at a time. Nonvalid entries are not accepted in this session.

Pressing the asterisk (*) key will display a list of available editing commands while in the PTE table.

NOTE : *Modification of a design with PTE which was previously created with BLAST, will NOT cause modifications to the original BLAST equations.*

The Input/Clock option for the PLUS405 cannot be changed to "CLOCK" (H) until the Input/Clock column (I6) of each product term is in the don't care (-) state. When "CLOCK" option (H) is selected, PTE will then allow the user to make any changes to the Input/Clock column. Failure to define the I6 column as don't cares will result in an error message. Once the I6 column is defined, PTE will deny any changes unless the clock option is defined as L. The Input/Clock column of all product terms is automatically changed to the proper state as the Input/clock fuse is changed.

The PTE editor cannot be used to edit PLE's (i.e., PROM's). Other editors may be used to edit the Standard file of PLE's, although Boolean equation entry of new information through BLAST is recommended.

Fusemaps for PLS155-159 or PLS179 will be interpreted in their most general form (as J-K FFs, next state syntax: H, - in PTE) when uploaded through DPI. This may result in user confusion when they were initially described as D-FFs (next state syntax: A, . in PTE). The results will be correct and consistent regardless of initial syntax. In other words, "H" is functionally identical to "A" and "-" is functionally identical to ".".

4.3 Fuse Table Editor

4.3.1 Description

FTE is a subset of PTE and a special editor for the PLHS501. It is automatically called when PTE is invoked on a PLHS501 design. This module is designed to create a new table or modify an existing table. Commands are entered by selection from Pull-Down menus.

During the FTE session the information is checked for validity and nonvalid characters are omitted. The output of FTE is a PLD Standard (.STD) file which can be used by other AMAZE modules.

It is strongly recommended that the user only use FTE for design verification. This is because the 72 NAND gate "Fold Back" array cannot be uniformly described in a Program Table format, as easily as for other PLDs. This means that modification efforts can be very hazardous to the net health of the design unless the user is an expert with FTE.

```
[4][5][6][7]:  Moves the cursor.  Will scroll the display.
RETURN       :  Moves the cursor to the beginning of the next line.
PgUP         :  Moves up one page or to the top of the file.
PgDN         :  Moves down one page.
HOME         :  Moves to the start of the line.
END          :  Moves to the end of the line.
*or F10      :  Moves cursor to PULL-DOWN main menu.
<ESCAPE      :  Exits each level of PULL-DOWN.
```

TABLE 4-1 FTE Cursor Control

4.3.2 How to use FTE

It is recommended that the user only use FTE for verification of a design and not for modification.

PML EXAMPLE continued from Section 3.2.6.1.1. Reference: fusemap starting next page.

FTE is entered by invoking PTE and selecting a new or existing filename of a PLHS501 design. When FTE is invoked, the user will be asked for the name of the file to be edited. If such a file does not exist, FTE asks if a new file is to be created. If the answer is "Y" then a new PLHS501 file will be created. If the file already exists, it will be read in and checked for:

1. Correct device name
2. Correct characters in each line of the file.

Appropriate error messages will be displayed if errors are detected; press (ESC) after reading the message and before the file is displayed. Characters are checked as they are entered and if they are illegal characters for that position, they are ignored. Characters may not be added past the end of the line.

The screen will scroll as new lines are needed. At all times the cursor position line and column will be displayed in a window in the upper right-hand corner of the FTE screen. On the top line of the window will be the labels for the row and column.

```
File
       Save file       - Save the file
       Quit            - Quit FTE and lose edits
       Print           - Copy fusemap to Filename.LST
       Copy            - Copy one file to a differently
                         named file or disk drive
```

```
Search
       Find            - Find a pattern
       Next            - redo previous Find
       Top of file     - move cursor to top of file
       End of file     - move cursor to end of file
       Line            - move cursor to a specific line
       Column          - move cursor to a specific column
Window
       Select          - move to other selected window
       Open            - open 2nd window
       Close           - close window containing cursor
```

TABLE 4-2 FTE Main and Sub-Menu Editing Functions

Editing commands, defined and shown in Table 4-2, may be invoked within FTE two ways. The first, and longer method, is started by pressing <F10> or "*". This will cause a menu line to be displayed on the top of the display. By typing the first letter of an entry, or moving the cursor with the arrow keys to cover an entry and pressing <Return>, a sub-menu will be displayed. To execute a function press the first letter of an entry, or position the cursor with the arrow keys and press <Return>. Exit from the main or a sub-menu by pressing <ESC>.

The second method, which must be performed only while the cursor is within the editing area, consists of typing the control character sequence associated with the desired function shown in Table 4-3. Using the control character sequence will not work when the cursor is in a menu. For reference only, the control character sequences are also listed within FTE along side their corresponding sub-menu selections.

```
File
        Save file       - ctrl-k ctrl-s
        Quit            - ctrl-k ctrl-x
        Print           - ctrl-k ctrl-w
        Copy            - ctrl-k ctrl-c
Search
        Find            - ctrl-q ctrl-f
        Next            -        ctrl-l
Go-to
        Top of file     - ctrl-q ctrl-r
        End of file     - ctrl-q ctrl-c
        Line            - ctrl-o ctrl-n
        Column          - ctrl-o ctrl-i
Window
        Select          - ctrl-o ctrl-g
        Open            - ctrl-o ctrl-o
        Close           - ctrl-o ctrl-y
```

TABLE 4-3 FTE Editing Functions Control Character Sequences

Exit from FTE by choosing the QUIT or SAVE commands from within the FILE menu. If file changes have been made followed by the QUIT command being executed, the user will be promptd to save the file. Exiting may occur with or without saving old files. Upon exiting with SAVE, the "OLD" file will be remained to fn.BAK.

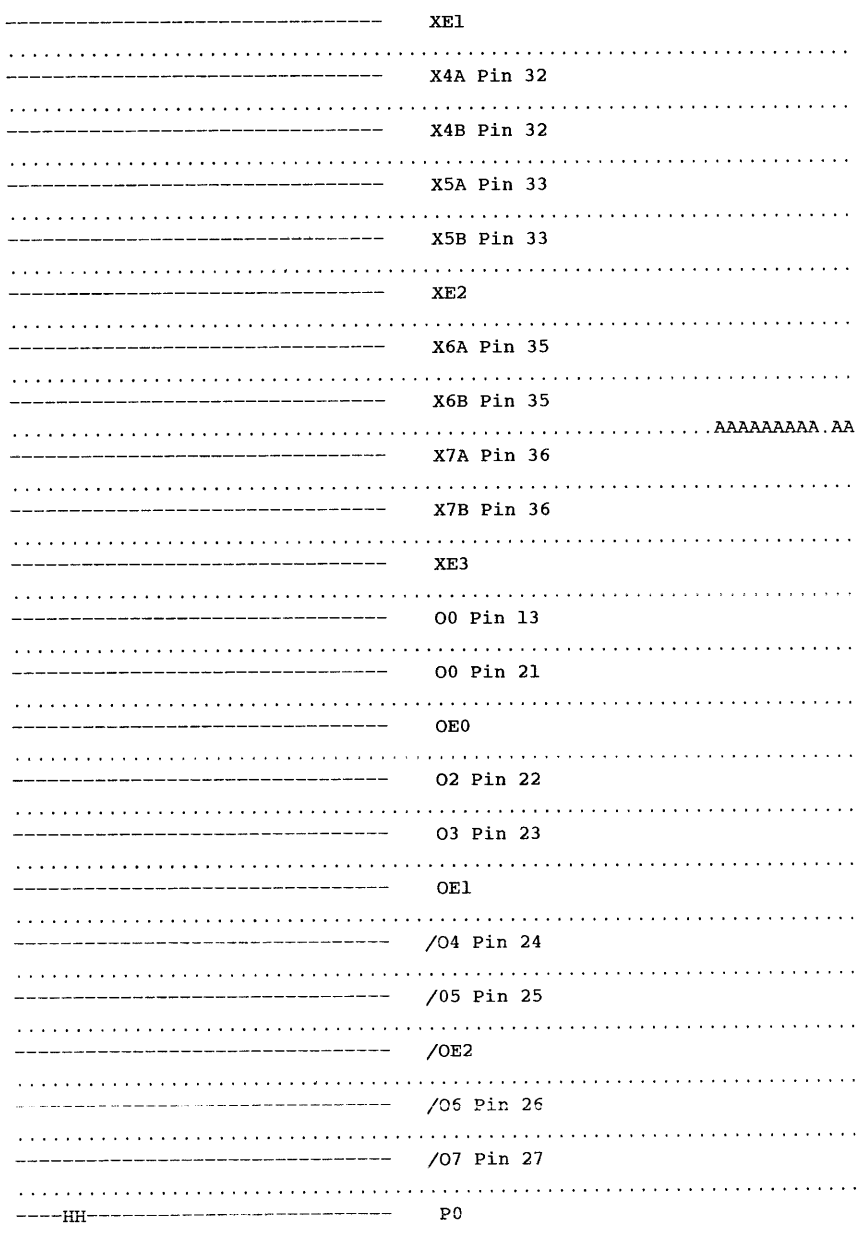
Note: *Modification of a design with FTE which was previously created with BLAST, will NOT cause modifications to the original BLAST equations.*

4.3.3 FTE Default Labels

The labels for the rows and columns are in the file EDITLAB.MSG These labels have a maximum length of ten symbols. The line for the "label" may be changed (with maximum length of ten). The total number of lines in the three fields may NOT be changed. The "start field" and "end field" lines start with an "@" and may not be changed.

A fuse table is printed out on the next eight pages. This "FUSEMAP" is the result of the SOP example in Section 3.2.6.1.1.

```
--------------------------------  /B0 Pin 37
................................................................................
--------------------------------  /B1 Pin 38
................................................................................
--------------------------------  /B2 Pin 39
................................................................................
--------------------------------  /B3 Pin 40
................................................................................
--------------------------------  B4 Pin 15
................................................................................
--------------------------------  DB4
................................................................................
--------------------------------  B5 Pin 16
................................................................................
--------------------------------  DB5
................................................................................
--------------------------------  B6 Pin 17
................................................................................
--------------------------------  DB6
................................................................................
--------------------------------  B7 Pin 18
................................................................................
--------------------------------  DB7
................................................................................
--------------------------------  X0A Pin 28
................................................................................
--------------------------------  X0B Pin 28
................................................................................
--------------------------------  X1A Pin 29
................................................................................
--------------------------------  X1B Pin 29
................................................................................
--------------------------------  XE0
................................................................................
--------------------------------  X2A Pin 30
................................................................................
--------------------------------  X2B Pin 30
................................................................................
--------------------------------  X3A Pin 31
................................................................................
--------------------------------  X3B Pin 31
................................................................................
```


XE1
X4A Pin 32
X4B Pin 32
X5A Pin 33
X5B Pin 33
XE2
X6A Pin 35
X6B Pin 35
AAAAAAAAA.AA
X7A Pin 36
X7B Pin 36
XE3
O0 Pin 13
O0 Pin 21
OE0
O2 Pin 22
O3 Pin 23
OE1
/O4 Pin 24
/O5 Pin 25
/OE2
/O6 Pin 26
/O7 Pin 27
HH
P0

```
------------------------------    P1
.....................................................................A..
------------------------------    P2
.....................................................AAAAAAAAAA.....A
H----H-------------------------    P3
.....................................................................
-----H--------------------HH----    P4
.....................................................................
H--H---------------------------    P5
.....................................................................
---H----------------------HH----    P6
.....................................................................
---HH--------------------------    P7
.....................................................................
HHH----------------------------    P8
.....................................................................
-HH----------------------HH----    P9
.....................................................................
-HH-H--------------------------    P10
.....................................................................
-H-HHH---------------------H----    P11
.....................................................................
--HHH--------------------------    P12
.....................................................................
HH---H-------------------------    P13
.....................................................................
-H---H----------------------H----    P14
.....................................................................
-H--HH---------------------H-----    P15
.....................................................................
--HH-H-------------------------    P16
.....................................................................
----H--------------------HH----    P17
.....................................................................
--HHH--------------------HH----    P18
.....................................................................
-H--HH-------------------HH----    P19
.....................................................................
-H----H------------------------    P20
.....................................................................
-H--H--------------------------    P21
.....................................................................
-HHH---------------------------    P22
.....................................................................
------------------------------    P23
.....................................................................
```

---------- P24
. . . . . .
---------- P25
. . . . . .
---------- P26
. . . . . .
---------- P27
. . . . . .
---------- P28
. . . . . .
---------- P29
. . . . . .
---------- P30
. . . . . .
---------- P31
. . . . . .
---------- P32
. . . . . .
---------- P33
. . . . . .
---------- P34
. . . . . .
---------- P35
. . . . . .
---------- P36
. . . . . .
---------- P37
. . . . . .
---------- P38
. . . . . .
---------- P39
. . . . . .
---------- P40
. . . . . .
---------- P41
. . . . . .
---------- P42
. . . . . .
---------- P43
. . . . . .
---------- P44
. . . . . .
---------- P45
. . . . . .
---------- P46
. . . . . .

```
--------------------------------   P47
................................................................................
--------------------------------   P48
................................................................................
--------------------------------   P49
................................................................................
--------------------------------   P50
................................................................................
--------------------------------   P51
................................................................................
--------------------------------   P52
................................................................................
--------------------------------   P53
................................................................................
--------------------------------   P54
................................................................................
--------------------------------   P55
................................................................................
--------------------------------   P56
................................................................................
--------------------------------   P57
................................................................................
--------------------------------   P58
................................................................................
--------------------------------   P59
................................................................................
--------------------------------   P60
................................................................................
--------------------------------   P61
................................................................................
--------------------------------   P62
................................................................................
--------------------------------   P63
................................................................................
--------------------------------   P64
................................................................................
--------------------------------   P65
................................................................................
--------------------------------   P66
................................................................................
--------------------------------   P67
................................................................................
--------------------------------   P68
................................................................................
--------------------------------   P69
................................................................................
```

```
--------------------------------    P70
.....................................................................
--------------------------------    P71
.....................................................................
HLLL      /B ENABLE

COMMENT:
The last line above indicates that the PLHS501 output steering fuses are to be
programmed as (Pins 37-40):
H   L     L    L
^    ^     ^    ^
|     |     |    |_/B0
|     |     |_/B1
|     |_/B2
|_/B3
```

SIM

Users Guide

5. PLD Functional Simulator (SIM)

5.1 Description

The PLD Functional Simulator (SIM) is a functional simulator designed for PLD devices. It will display the reaction of a programmed device to a set of input vectors. Figure 5–1 depicts the structure of this simulator.

The PLD logic description for the simulator can be generated through one of several AMAZE software modules. This description is in the form of a *fn.STD* file.

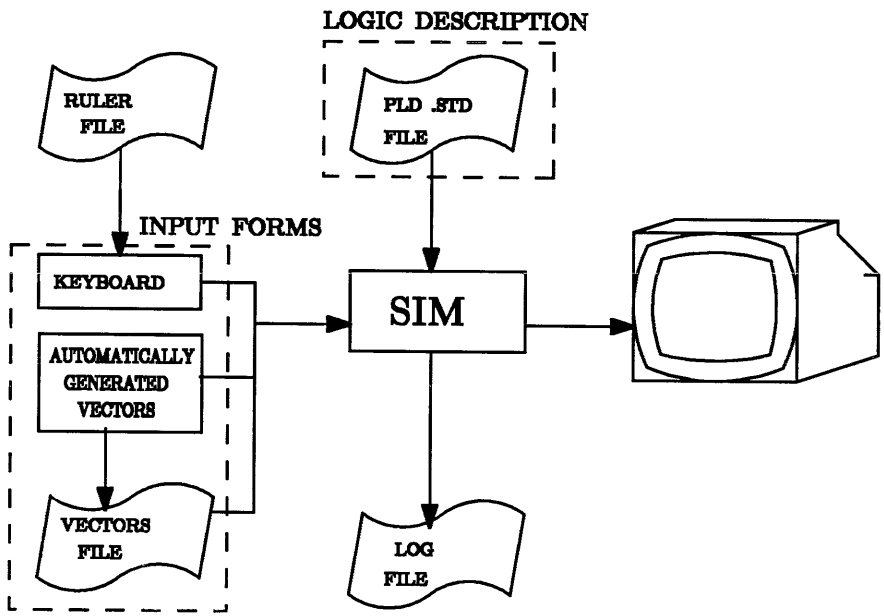


Figure 5–1 Overview of SIM

SIM begins by displaying the menu shown in Figure 5–2. SIM allows three different modes of input vector entry for simulation of the PLD device.

Interactive simulation is the operation of entering input vectors and being given the device responses, one vector at a time. Vector set simulation allows application of a file set of input vectors in the design file directory to the design mode. Automatic vector generation will automatically create a set of input vectors with a heuristic algorithm using the device pattern.

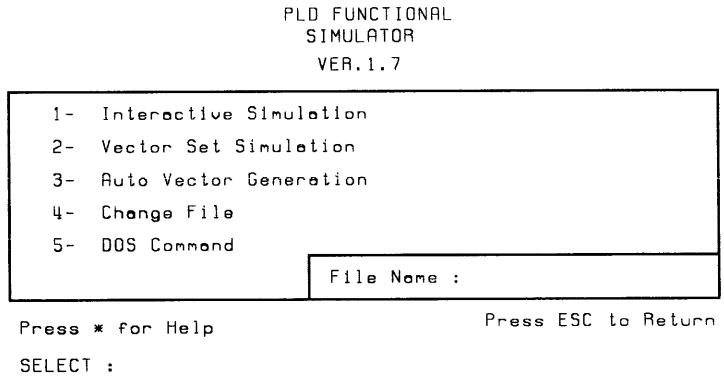


Figure 5–2 SIM Main Menu

One can switch between options 1 through 3 for the same pattern while the pattern filename is the same (specified in the lower right corner of the menu).

The user is promptd for the pattern filename upon first entering the simulator. The simulation pattern filename can be changed using Option 4 in the menu. For the filename, enter the same filename used in creating the PLD pattern (.STD file); do not include any extension when specifying the filename. The pattern will be checked for errors and any detected problems will be displayed on the screen. For debugging

purposes, a TRACE option is provided to show how a specific output is generated by indicating which product terms were activated by the input vector (see section 5.5).

Online HELP can be invoked by entering an asterisk (*) at any point within the PLD simulator. Once a menu selection is made, the user will be promptd by the following question:

LOG Function desired? (Y/N) [N]

The LOG file is the documentation of the simulation session and is used to create the PLD test vectors (in JEDEC format). A description of the LOG file is given in Section 5.6. Push the return key (a null entry) to bypass LOG if it is not needed. On the other hand, if a history of the simulation session is desired to be saved in a LOG file, type the letter "Y" (upper or lower case). SIM will then ask for a header or comments to be placed at the beginning of this file. Type any LOG file header or comment lines desired (up to 65 characters per line). A null entry will end the comments.

If the user chooses to bypass the LOG Function or has completed entering the LOG header information, SIM will then enter the operating mode requested.

Upon selecting interactive mode (Selection 1), SIM will first instruct the user on the specific input character sequence for the PLD Device being simulated. The user will also be supplied a list of special functions available for the device being simulated. This information can be repeated by requesting HELP (with an "*") at any input vector prompt. The input vector prompt is:

INPUT VECTOR:

When the correct number of inputs have been entered after the input vector prompt, SIM will display the projected response of the PLD device (with output functions listed from MSB–LSB or as specified in the RULER file) and prompt for another input. To quit the simulation, enter Q at the "Input Vector" prompt and control will be returned to the SIM menu. At this point, one may return to the simulation in interactive or vector set mode, with or without LOG, or may exit to return to the AMAZE main menu.

5.2 Ruler File

When SIM first asks for an input vector after choosing menu option 1 – interactive mode, it will display a heading for the input vector expected. This heading is called a RULER. One may enter input vectors with any spacing desired but the sequence must be shown on the ruler. The ruler may be toggled between a 'custom' and a default display by entering an 'R' instead of a normal input vector at any time during the simulation session.

The custom ruler information is held in an ASCII text file, 'Filename.RUL' (see Figure 5–3). If this file does not exist upon entering SIM, selection of option 1, 2, or 3 from the menu will cause one to be created using pin names given in the pinlist. The order in which input vectors are input during interactive mode may be changed by reordering lines in the ruler file. Also, as the device pin numbers in the ruler file are what identifies a pin, the pin label in the ruler file may be any ASCII text. It does not have to match the label given in the pin list.

CAUTION! Since SIM reads the ruler file upon entry, altering the ruler file by invoking an editor using option 5 – DOS Command, will not update the display if option 1 is immediately chosen. It is necessary to either leave SIM and return, or choose option 4 – Change Filename and keep the same filename.

Also, SIM does not check if @PINLIST assignments changed. So after altering pins either edit the ruler file or delete it and a new ruler file will be created upon invoking menu option 1, 2 or 3.

The Ruler file is only used to display the inputs and outputs on the screen during "interactive simulation", and it has no effect on the input sequence of vector (Fn.VEC) files or the pin sequence of the LOG file. The Ruler file is split into two sections (Figure 5-3), the INPUT section and OUTPUT section. @INPUT and @OUTPUT are the headings for these sections. Under both headings, every line corresponds to one input or output pin of the device.

In some instances, SIM will write the label of a combinatorial "B" pin in both the @INPUT and @OUTPUT fields of the Ruler (.RUL) file. This will cause the label to appear at both the "Input Vector" and "Output Vector" prompts during interactive simulation, which may not be desired for cosmetic reasons. Using a text editor to remove a B-Input from the @Output field and an output B-Pin from the @Input field will correct the situation of the wrong input and output labels appearing next to the prompts during interactive simulation (this is a cosmetic situation only).

```
PIN    PIN     PRE-ASSIGNED
NBR.   LABEL   (FIXED) VALUE
@INPUT
 08    INPUT7    :  H          |
 07    INPUT6    :  L          |
 06    INPUT5    :  L          |
 05    INPUT4    :  H          |
 04    INPUT3                  |
 03    INPUT2                  |                      BBBB BBBB BB
 02    INPUT1                  |                      OOOO OOOO OO
 01    INPUT0                  |                 IIII UUUU UUUU UU
                               |                 NNNN TTTT TTTT TT
 19    BOUTPUT9                |                 PPPP PPPP PPPP PP
 18    BOUTPUT8                |                 UUUU UUUU UUUU UU
 17    BOUTPUT7                |                 TTTT TTTT TTTT TT
 16    BOUTPUT6                |                 3210 9876 5432 10
|   Input Vector
 15    BOUTPUT5                |
 14    BOUTPUT4                |
 13    BOUTPUT3                |
 12    BOUTPUT2                |
 11    BOUTPUT1                |
 09    BOUTPUT0                |
 @OUTPUT                       |
 19    BOUTPUT9                |                      BBBB BBBB BB BBBB BBBB BB
 18    BOUTPUT8                |                      OOOO OOOO OO OOOO OOOO OO
 17    BOUTPUT7                |                 IIII UUUU UUUU UU UUUU UUUU UU
 16    BOUTPUT6                |                 NNNN TTTT TTTT TT TTTT TTTT TT
                               |                 PPPP PPPP PPPP PP PPPP PPPP PP
 15    BOUTPUT5                |                 UUUU UUUU UUUU UU UUUU UUUU UU
 14    BOUTPUT4                |                 TTTT TTTT TTTT TT TTTT TTTT TT
 13    BOUTPUT3                |                 3210 9876 5432 10 9876 5432 10
 12    BOUTPUT2                |  Output Vector:
 11    BOUTPUT1                |
 09    BOUTPUT0                |
|                              |     Corresponding Input Vector &                |
|A:fn.RUL file                 |     Output Vector Rulers . This half of the     |
|                              |     figure is how the display will look in      |
|                              |     SIM and is not part of the fn.RUL file.     |
```

Figure 5-3 Ruler file with Input and Output Vector Headings

5.3 .RUL File Headers

@Input

@Output

5.3.1 @Input

Format:

```
PIN     LABEL  :  VALUE
PIN     LABEL
```

Where PIN is the device input pin associated with LABEL described in the pinlist. VALUE is optional and will fix the value (H or L) of the pin throughout the simulation.

Description:

Input sections contain three types of information, Pin number, Pin label, and Preassigned Value (if desired). Pin labels may contain 1 to 12 characters and must be separated by at least one blank from other entries in the same line (i.e. Pin number and assigned value). By Preassigning a value (H or L) to an input pin, the user will be setting that pin to that value throughout the simulation. The Preassigned Value is optional and it's useful for pins which are not being used or for pins with a fixed function such as an Output enable pin. A ':' should be used between the pin label and the Preassigned Value with at least one space on either side of the colon. Input pins which were given a pre-assigned value in the .RUL file, or were not listed in the @Input section will not show up in the Ruler heading during interactive simulation. Bidirectional pins may be listed in this section.

Example:

```
@INPUT
1 CLOCK
2 SENSE_1
3 SENSE_2
4 START
5 OUTPUT_EN :H
```

5.3.2 @Output

Format:

PIN LABEL

Where Pin is the device output pin associated with LABEL described in the pinlist.

Description:

Output sections contain two types of information pin numbers and pin labels. Pin labels may contain 1 to 12 characters and must be separated by at least one blank from other entries on the same line. Output pins not listed in this section will not show up in the Ruler heading. Bidirectional pins may be listed in this section.

Example:

```
@OUTPUT
12 /GREEN
13 /YELLOW
14 /RED
18 Q1
19 Q0
```

5.4 Interactive Simulation

Upon beginning interactive simulation, SIM will display the specific sequence of characters required for each input vector (called Ruler heading – see Section 5.2) and a set of applicable help and control messages. The Help information will be repeated if '*' is entered at any time during Input Mode.

In general, the user must enter the inputs, bi-directional bits, clock or chip enable bits and any other bits needed for proper setting of the device. All device inputs must be in H or L characters (upper and lower case allowed), while bidirectional lines can be given as H, L or dash ("-"). The dash is used when a line is expected to be an output. Characters may be spaced as desired for grouping or readability.

```
ENTER H or L characters corresponding to each I or Q
  and H, L or "-" characters corresponding to each B
  Use spaces as you wish for readability.

A "/" will repeat the equivalent character from the last valid input.
A "=" will repeat the remaining characters from the last valid.
Enter "D" to display LOG file so far.
Enter "I" to toggle timing display screen prompt.
Enter "M" to run a set of vectors from external file.
Enter "P" to show previous input and result.
Enter "Q" to exit run mode.
Enter "R" to toggle ruler.
Enter "T" to toggle trace mode.
Enter "W" to toggle warning message mode.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                IIIIIIII BBBBBBBBBB } DEFAULT RULER FILE (FOR PLS153)
                76543210 9876543210 }     "       "     "          "
INPUT VECTOR:  LLLLLLLL ---------- <-USER INPUT VECTORS (H, L, -)
               "
               "     INPUTS <=B(I/O=> TRACE TERMS
               "765432F10 9876543210
OUTPUT VECTOR: 000000000 11LLLLLLHL ; <- SIMULATOR OUTPUT VECTOR

TIMING DISPLAY (Y/N) (N)          [NOTE: INPUT CONDITIONS: 0, 1]
                                  [      OUTPUT CONDITIONS: H, L]
```

Special functions can be invoked by entering the following characters:

/ will repeat a character from the last valid input
= will repeat the rest of the last valid input.

The above characters can be placed anywhere in the input vector.

Timing relationships of output pin signals may be displayed during interactive simulation. To display the signals type Y when prompted followed by a carriage return (CR) after the output vector is displayed. The prompt message for a timing display may be turned off by entering an "I" at the input vector prompt.

Each timing display is representative of only the current input vector. Timing displays do not show cumulative information from previous displays.

When simulating a device that has internal registers and a dedicated clock line, a small up-arrow will appear on the timing scale line. This arrow displays where the active clock edge occurred in the display. Remember that an input vector of "H" on a dedicated clock line is interpreted by the simulator as a low-high-low pulse. The clock will not occur in the simulation until the inputs to the clocked register are stable.

Figure 5-4 is the timing of the output signals at the "B" pins for a PLS153.

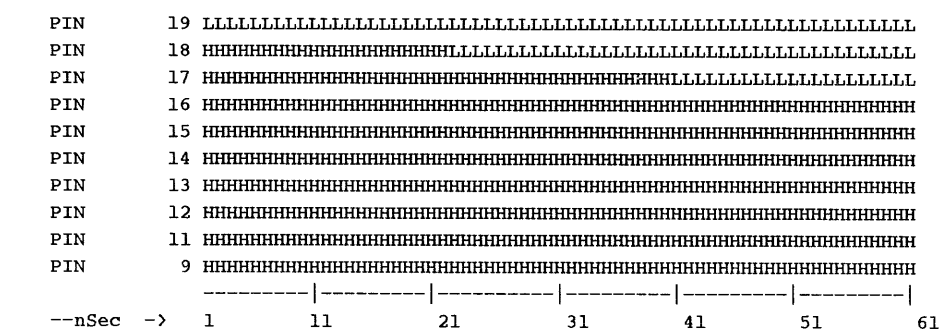


Figure 5-4 B-Pin timing for a PLS153

When simulating a device that contains registers built from gates, the user should keep three points in mind. First, due to feedback, some of the gates will have unknown outputs upon first entering the simulator. This is common to any simulator. Also, due to the feedback, SIM will take a relatively long time to enter. It will display the message "Outputs are not stabilized within 251 nsec". The very

first vector applied should force these registers into a known state, either set or reset (i.e., each register should be designed with a set or reset). If not, any subsequent vector will continue to propagate the unknown condition and the "output not stable" message will continue.

Secondly, SIM cannot know that a flip flop has been constructed, so the clock line must be explicitly toggled. For example, if set "L" in one vector, it needs to be set "H" and then "L". Lastly, the data input to a flip flop should not change in the same input vector as the active edge of the clock signal or else an unstable condition may occur. If so, the flip flop must be put into a known condition with a set or reset.

For some sequential devices the state registers may be forced to a specific state by entering an "F" followed by the correct number of H and L characters for the number of state registers. If this is used, any active LOG file will contain a special message declaring a forced state change (with the key characters "+"). Since this function is not a part of the actual device operation, the LOG file should not be used to create test vectors for device testing (test vectors created with the "F" function may result in vector fail messages on PLD Programmers).

Figure 5–5 shows examples of some input vectors and commands.

```
HHHHLLLLLHLHLHHHHL          (full 17 characters input
                               setting all inputs and PE bit)
 hhhh llll hlhl hhhh l      (same input with lower case
                               characters and spacing)
////HLHL///////H/           (repeat some characters and
                               change others)
hhl=                        (set first three bits and
                               repeat all others)
d                           (Display the LOG file so far)
p                           (show previous input and
                               output)
*                           (HELP - show input vector
                               character options)
f hhhlhl                    (Force state to HHHLHL)
=                           (repeat all of last input
                               vector)
S                           (Show the internal registers
                               on the output pins)
M                           (Run a set of vectors stored in
                               an external file.)
Q                           (Quit manual mode and return to
                               SIM menu)
```

Figure 5–5 Examples of input vectors/commands for PLS105 Simulation

Note: The timing display cannot be invoked for PROMs.

5.5 Trace Function

TRACE provides a debugging tool in the PLD simulator. It can be activated or deactivated by entering a "T" at any "INPUT VECTOR" prompt. When active, TRACE will indicate which product terms of the fuse pattern were used in deriving the final output. The activated product term numbers are displayed after the resultant output. The control terms (i.e. FC, D0, D1,...) are displayed with a preceeding letter 'C'. The numbering sequence for control terms are from top to bottom as specified in the Program Table. For example, for the PLS155, C1 refers to the FC term, C2 refers to LB term, and C11 refers to D0 (refer to individual PLD Data Sheets or PTE for the desired PLD).

5.6 Log Files

An optional output file may be created for documentation or for translation to test vectors for a device tester or PLD Programmer. This is the LOG file and will have the same filename as the pattern file and an extension of .LOG. When requesting this operation, the simulator will prompt for comments to be placed in this file. One may enter as many lines of comments as desired with up to 76 characters per line. Comments are ended upon entering an empty (null) line. SIM will automatically add a quote sign and space at the beginning of each line and will insert additional lines to indicate pins or functions of the LOG data. The data will reflect the input vectors and the resultant device outputs along with the previous and next states for sequential devices. Clock and enable information is also included. Inputs to the device are recorded in 1 and 0 format while outputs remain in H and L format.

The LOG file is necessary if functional testing of the PLD at programming is desired. DPI (Device Programmer Interface) uses the "Fn.LOG" to create and append the test vectors to the PLD fusemap file for transfer to the PLD Programmer in JEDEC format; see Section 6 for DPI details and operation.

```
PLS153 dflow153.STD
"This LOG file was created from a user input file
"7/17/84    TMS
"
" INPUTS (=B(I/O)=) TRACE TERMS
" 76543210 9876543210
"
  11101110 LH...HH110  ;
  11101110 LH...HL101  ;
  11101110 LH...LL001  ;
  11101110 LH...LH010  ;
  11101111 HL...LH101  ;
  11111110 LH...10HL1  ;
  10111110 LH........ ;
"
" -X-X---- ----------       I/O CONTROL LINES
"          OOIIIBBBBI       DESIGNATED I/O USAGE
"          OOIIBBBBII       ACTUAL I/O USAGE
"
"  PINLIST
" 08 07 06 05 04 03 02 01 19 18 17 16 15 14 13 12 11 19 ;
Note: input conditions  1, 0.

      output conditions H, L.
```

All lines beginning with (") are comments.

Figure 5-6 Log File

The sequence of inputs and outputs are fixed in the LOG file and cannot be changed. A period represents a tri-stated output and an X represents an unknown or unresolved output. A file containing an X should not be used to create test vectors.

The various fields of LOG data will be delimited by spaces and each line of simulation data will end with a semicolon (";") followed by numbers referring to the product terms of the fuse pattern that matched the input vector if TRACE was active. A '+' at the end of a line in the LOG file indicates that there were more activated terms which could not be listed on the same line.

Some special functions, such as presetting the PLS105 or setting the state of the PLS159 require special clocking for test and in these cases appropriate lines are added in the LOG file. For this reason, there may not be an exact line-to-line relationship between the simulation inputs and the LOG file.

For full simulation of sequential devices SIM allows the direct presetting of the state registers. Since this does not happen in reality in some PLD devices (i.e. PLS105, PLS167, PLS168), the LOG file will contain a message that there was a forced state change and the file should not be used to create test vectors. A blank comment line (with a quote sign in the first column) will indicate the end of the input and output vector information.

Next in the LOG file will be three additional comment lines to provide information on I/O control and usage of bi-directional pins. The first line will contain a series of dashes and "X" characters under several of the bit lines. The dash represents an input or an I/O line capable of controlling the direction of one or more I/O pin, while the "X" indicates a bit whose value does control I/O direction in this pattern. The next line will contain characters to indicate the designated direction of each I/O line as I (input), O (output) or B (bi-directional). The third line will indicate the actual usage of each I/O pin for this simulation session. A comparison of the second and third lines will indicate if the applied input vector sequence exercised all the capabilities of the design.

LOG files for devices without I/O pins will have blank lines in this area. The next few lines, following a blank comment line, will contain a listing of the device pin numbers in the order they appear within the LOG data above. Functions in the LOG listing for which there are no device pins (such as state levels) will appear as "00" in the pinlisting. Figure 5-6 shows an example of a LOG file.

5.7 External Vector Set Files

An External Input Vector File can be used to enter a sequence of input vectors as a group during the vector entry operation. Vector Set Simulation may be used to develop test vectors, step sequential devices to a known state or simply demonstrate device function.

Before writing an Input Vector File, it will be necessary to determine the proper input character string sequence. This is because SIM does not use the custom Ruler file (fn.RUL) from the pinlist information for external vectors. SIM requires external vectors to be input according to the default Ruler heading. Therefore, the easiest method to view a default heading is to invoke the simulator and choose option 1 - Interactive Simulation. Type an 'R' to toggle the Ruler to it's default display. Note the ordering of inputs and bidirectional pins by their data sheet names.

Vectors are written using the characters H, L, -, /, or =. Their meaning is the same as for interactive input vectors, defined in Section 5.4. Blank characters may be inserted into the input character string for grouping or readability. Comments are not allowed within the body of the input character string, but they may be placed on the same line provided the proper number of H, L, -, /, or = characters to satisfy all input pins did occur. Lines beginning with a quote character (") are interpreted as comment lines. Please refer to figure 5-7 for a sample external input vector file.

Any errors detected in this file will be displayed on the screen. Each line of entry is treated as if it were a manual entry and is run by the simulator one line at a time. Thus it is possible to use all the special functions indicated in Section 5.4. SIM will stop reading the external file when it encounters a line beginning with a "Q". This will return control to the simulator menu where the user may continue the simulation manually or from another external input vector file. SIM will always read external input vector files from the beginning. Anything after the "Q" will not be seen. When the "Vector Set Simulation" (option 2) is selected from the simulator menu, the user is prompted for the name of the input vector file; the user must specify both the filename and the extension (i.e., FN.EXT) if one exists.

```
"Line designations:"
"     I6 = /INH     Inhibits outputs (makes all inputs)
"     I4 = DIR      PB =) PA when high, PA =) when low
"     Il = CLK      High to operate
"     I0 = CROSS    High to cross, low for direct

"     B9 = QTRUE    High to indicate direct operation
"     B8 = QFALSE   High to indicate crossed operation
"     B4-B3 = PB    (1-0)
"     B2-Bl = PA    (1-0)
" All other pins unused
"
" Inputs      I/O's          Function
"
"7654321 99876543210
"
 /H/L/HL --------HH     = Expect PB to be HH )PA) =) PB direct)
 ///////  ///////HL    = Expect PB to be HL
 ///////  ///////LL    = Expect PB to be LL
 ///////  ////////LH   = Expect PB to be LH
 ///////H  ////////HL = Expect PB to be LH (PA =) PB crossed)
 ///H///L /////HL--    = Expect PA TO BE HL (PB =) PA direct)
 /L////H/ /////----    = Expect PA and PB to be off (Inhibited)
quit
```

Figure 5-7 External Input Vector File

5.8 Auto Vector Generation

The Auto Vector Generation option in the SIM menu will automatically create a set of input vectors optimized according to the fuse pattern. These vectors are saved in the *filename*.VEC file. The simulator will then run the generated test vectors against the fuse pattern and saves the results in the *filename*.LOG file.

Auto Vector Generator will flag any input condition which can cause an indeterminant state in the device; these input vectors are automatically taken out of the generated vector list. To determine which product term(s) caused the condition, use the Interactive mode with the Trace On, force the present state (indicated when the condition was flaged) and then apply the appropriate input condition. Automatic vectors are generated by "Walking" ones and zeroes through the inputs, and even though this produces acceptance test results for combinatorial designs, it is by no means exhaustive. For this reason, the vectors generated may not be the best or the most optimized set of vectors for the pattern [especially with sequential (state machine) designs]. You may want to enhance the test vectors by adding input vectors to the generated vectors by editing the "Fn.VEC" file.

Choosing this option for devices with buried registers (PLS155, 157, 159, 179, and PLUS405) will invoke a prompt asking if supervoltage vectors are to be used. These devices have a diagnostic mode which is entered by pulling an assigned pin to +10 volts. Once in this mode, the internal registers may be preset to a value applied to the output pins. Supervoltage vectors refer to this diagnostic mode of presetting registers. Not all device programmers support this feature, even though they may be capable of accepting test vectors, so use with care.

DPI

Users Guide

6. Device Programmer Interface

6.1 Description

The Device Programmer Interface (DPI) allows the user to pass a PLD fusemap from a host computer to a logic device programmer (Download), or from a logic device programmer to a host computer (Upload).

DPI supports JEDEC and Signetics H/L formats and a low cost Programmer format. Figure 6-1 shows the DPI main-menu. DPI will use the Default information as defined in the AMAZE.DEF file. If this file does not exist, a set of default values will be assigned to the Format and communication configuration (i.e. Baud rate, data bits parity). The user may temporarily change the Format and Communication Baud rate, while in DPI, or use Option 6 (Set Default) of the AMAZE main menu to change all of them permanently. BLAST "Compile" menu also has an option to generate the PLD JEDEC Fusemap File (Fn.JED) without test vectors if your PLD programmer requires only a JEDEC file instead of a serial port "Download". This option allows JEDEC file generation without using DPI.

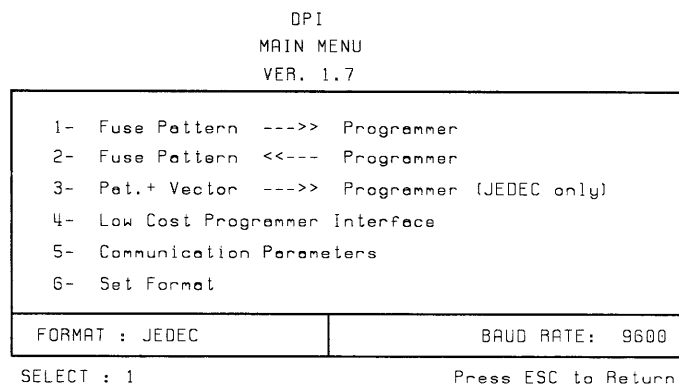


Figure 6-1 DPI Main Menu

6.2 How to Use DPI

6.2.1 Downloading (Fuse Pattern --> Programmer)

When the Download option is chosen (Option 1), the user is asked to enter the standard fusemap filename (fn.STD file) to be downloaded. DO NOT specify the extension. This file must reside in the design file directory.

For device programmers which support functional testing of programmed devices, Option 3 (Pat.+Vector -->Programmer) in the menu may be selected. It will cause DPI to download PLD Test Vectors created by the AMAZE SIMULATOR with a .LOG extension along with the PLD fusemap (.STD file). This operation can only take place in JEDEC format. Both the STD and the LOG files must reside in the design file directory.

6.2.2 Uploading (Fuse Pattern <-- Programmer)

When uploading a fusemap from a device programmer (Option 2), the user must specify both the filename and the device type which is selected from a menu. DPI will create a file in the design file directory with the specified filename and STD extension. The filename can be up to eight characters long. Make sure the information from the Device Programmer is sent AFTER the following prompt appears:

The host is ready to receive programmer data.

When downloading (or uploading), the file which was sent (or received) from the programmer will be filename.JED for JEDEC format and filename.SIG for Signetics H/L format. The file (and test vectors if option 3 download) is automatically written to the design file directory as a compatibility feature to the user.

PLC105, PLUS405A, and PLHS501 are supported by JEDEC format only (not the Signetics H/L format). Most commercial PLD programmers support the AMAZE DPI default format which is JEDEC Standard.

Note: For a PLHS18P8 device either dots (.) or (/) are allowed in the "OR" array (they mean the same). DPI puts SLASHs when passing files in either direction. This only is a note to users editing PLHS18P8 program tables with the Program Table Editor (PTE).

6.2.3 Low Cost Programmer Interface

When Option 4 in the DPI menu is chosen, DPI changes the Baud rate to 9600, and displays the menu shown in Figure 6–2. For options 1 to 3 the device must reside in the programmer socket. The VERIFY option verifies the pattern in the device against the pattern in the fn.STD file in the design file directory.

It's recommended to verify the patterns after Options 1 and 2, due to noise that might occur during Uploading and Downloading, on the communication lines.

Invoking the Low Cost Programmer Interface without first connecting the low–cost programmer to the RS–232 serial port will cause the MS–DOS computer to wait until the serial port connection is made. The Low Cost Programmer interface only communicates with COM1. COM1 should be set for 8–bits/character, 1 stop–bit, no parity.

The PLHS501 is also supported by its own low cost programmer which will be available with this release.

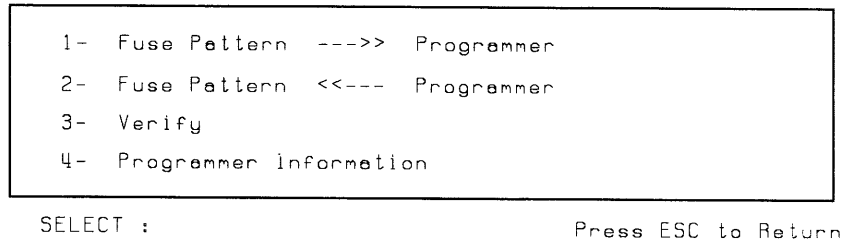


Figure 6–2 Low Cost Programmer Interface Menu

Note: The low cost programmer interface supports communication with the following low cost programmers:

For 20 pin PLD support (PLS151, PLS153, PLS155, PLS157, PLS159, only).

Curtis Electro Services, Inc.
Box 4098
Mountain View, CA 94040
Phone: (415) 964–3846

For PLHS501 support.

Strebor Data Communication, Inc.
1008 North Nob Hill Drive
American Fork, Utah 84003
Phone: (801) 756–3605

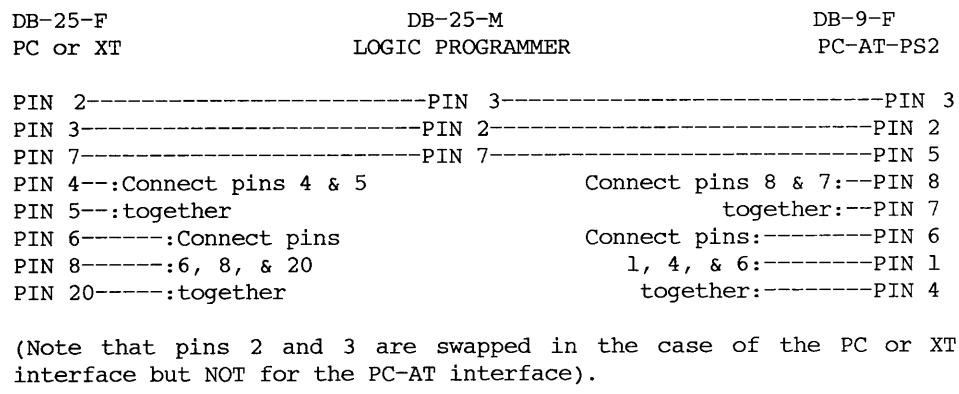
6.3 DPI/PTP Serial Communication

The PC must be equipped with an Asynchronous Communication Port -- either COM1 or COM2 .

The DPI/PTP default for COM1 is as follows:

EVEN parity
1 Stop Bit
7 Data Bits

The Programmer should be configured in the same mode as default. The defaults can be modified using option 6 (set default) in the AMAZE main menu to match desired programmer settings, the assigned serial communication parameters are stored in the AMAZE.DEF file. Figure 6-3 shows the MINIMUM pin configuration for the communication cable. A shielded cable is recommended to insure reliable RS-232 data transfers and reduce RFI interference. Note however, that when interfacing an XT to the Strebor programmer, there should be no swap of lines 2 and 3 as shown in Figure 6-3. In other words, pin 2 connects to pin 2 and pin 3 to pin 3. When interfacting an AT to the Strebor however, pin 2 connects to pin 3 and pin 3 connects to pin 2.



(Note that pins 2 and 3 are swapped in the case of the PC or XT interface but NOT for the PC-AT interface).

Figure 6-3 Minimum Pin Configuration for Communication Cables

When downloading or uploading in JEDEC format, note that the EA and EB fuse numbers for PLS159 are different for different Commercial Device Programmers. The fuse numbers used in this package are shown in Figure 6–4 –Format A (EB: 2106 and 2107 – EA: 2104 and 2105). Some Commercial Device Programmers use the fuse numbers shown in Format B (EB: 2104 and 2105 – EA : 2106 – 2107).

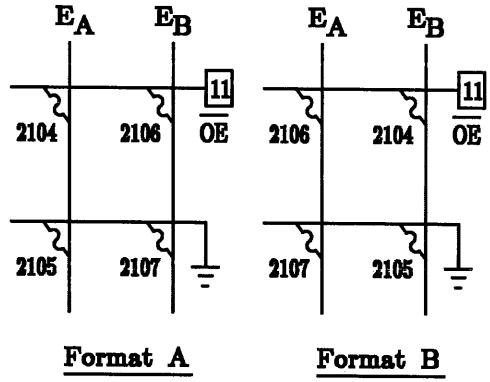


Figure 6–4 Different JEDEC Numbers for PLS159

Consult the Device Programmer Manual for the fuse numbers used. Example:

DATA I/O:	Format A
STAG:	Format B

Thus, when downloading or uploading a PLS159 pattern using JEDEC format, DPI software will prompt for the configuration that the device programmer supports. This will automatically switch the fuse numbers if it is required.

Low cost programmer users must insure that the RS-232 interface connections between the computer and the low cost programmer have been established before attempting to run DPI with the LCP.

PTP

Users Guide

7. PAL to PLD Conversion (PTP)

7.1 Description

PTP converts PAL designs/patterns into corresponding Signetics supplied PLD device designs/patterns. PTP supports two modes of operations:

1. PAL physical device into PLD physical device (via commercial PLD programmer).
2. PAL fuse pattern into PLD fuse pattern (via files read/written from/to the design file directory.

Every time PTP is run, it creates a Fn.PRT file in the design file directory. This file contains the PAL (fusemap) pattern and the corresponding PLD (fusemap) pattern, and provides documentation for the conversion process.

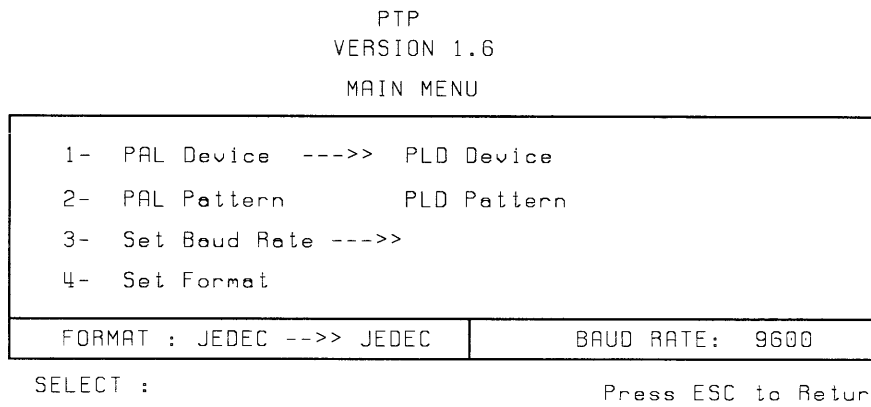


Figure 7–1 PTP Main Menu

7.2 How to Use PTP

7.2.1 PAL Device to PLD Device

In this mode the user is to obtain a PLD device functionally compatible with specified PAL device. The following steps are required for this mode of operation: (PTP is completely menu driven and prompts the user for all steps.)

1. Load the PAL device into the Device Programmer, as if preparing to program more devices from a master.

2. Send the fuse information to the host computer, when instructed by PTP[1] (usually an "output" or "upload" function of the PLD programmer. On Data I/O model 29 series, the command is "SELECT EC" and then "START").

3. Users of PTP with Data I/O Model 29 series programmers must clear the programmer memory (SELECT A4) before changing Program-&-Test Adapters; failure to do this may cause PTP to abort execution.

4. Receive the corresponding PLD pattern from the computer, when instructed[2] (usually an "input" or "download" function of the PLD programmer. On Data I/O model 29 series, the command is "SELECT EB" and then "START").

5. Program the PLD device. On Data I/O model 29 series: "COPY->RAM->DEVICE->START-->START".

---

[1] If too much time elapses before the PAL fusemap is sent from the programmer (or the serial connections have not been established, a "Timeout" error will result for the selected com port; if this happens, restart the process from Step 2.

[2] The PLD pattern sent file to the programmer will be filename.JED for JEDEC format and filename.SIG for Signetics' H/L format. This file will also be written to the design file directory.

A copy of the corresponding PLD STANDARD file has been saved in the design file directory under the following name:

fn.STD where fn is the filename specified by the user

This file may be used with other AMAZE modules for future modification. The PLD fuse pattern may be edited with PTE and downloaded to the programmer with DPI if altering of the PLD fuse pattern is desired.

The PTP communication format supports JEDEC, SIGNETICS H/L, and PAL's FUSE PLOT. All combinations for using PTP have been included in the PTP Format menu. PTP will use the default information, provided in the AMAZE.DEF file. If this file does not exist, a default set will be assigned to the Format and communication configuration. You can temporarily change the Format and Baud rate, while in PTP, or use Option 6 (Set Default) in the AMAZE main menu to change them permanently. Note that the serial communication defaults set by the user for DPI are also used for PTP.

7.2.2 PAL Pattern to PLD Pattern

In this mode of operation, a PAL pattern in the FUSE PLOT (- and X) or JEDEC format can be converted to its corresponding PLD STANDARD pattern. If the PAL pattern can fit into the PLHS18P8, then the PLD selection menu will be displayed (Figure 7-2).

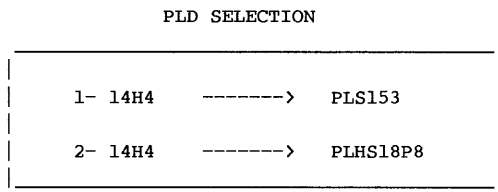


Figure 7–2 PLD Selection Menu

The 14H4 device is an example; any 20 pin nonregistered PAL device may be used here as selected by the user from the PAL menu.

The input file (fuseplot or JEDEC format) has to have the .PAL extension, and it must exist in the design file directory . The corresponding PLD pattern will have the same filename as the PAL filename, with the .STD extension. This allows DPI to be used to download the PLD fusemap (.STD PLD file) to the PLD programmer). PTP will also create the PLD JEDEC programmer file (.JED PLD File), so using DPI is optional for PLD programming.

If editing the fuse pattern with PTE, please note that PTP for the PLHS18P8 device puts dots (.) in the unusable sections of the OR array, unlike the DPI module which uses SLASHs (/) for the same function. This will not cause problems with DPI when downloading, as DPI will accept both formats.

7.2.3 PTP/DPI Serial Communication

Refer to Section 6.3.

UTILITIES

8. UTILITIES

The series of stand-alone modules listed below is provided:

1. TOJED
2. FROMJED
3. TOSIG
4. FROMSIG

These modules will read/write files via the default drive unless otherwise specified.

8.1 TOJED

This module creates a JEDEC file containing the fuse information and test vectors (if applicable). From the program table (.STD) file and the test vector (.LOG) file if specified.

Format :

TOJED *parameter1 parameter2 parameter3*
parameter1 : Input Fuse Pattern filename (Standard Format)
parameter2 : Output filename (JEDEC Format)
parameter3 : Input Vector filename (.LOG format)

If parameter3 is not provided, the output file will only contain the fuse pattern information.

Example :

TOJED examp.STD examp.JED ex-1.LOG

```
In the above example, the JEDEC correspondence of the fuse
pattern
(examp.STD), and Test vectors (ex-1.LOG) is stored in
examp.JED
```

TOJED test.STD tt1.JJD

```
In the above example, the JEDEC correspondence of the fuse
pattern  (test.STD) is stored in ttl.JJD.
```

8.2 FROMJED

This module converts a JEDEC file (.JED) into a PLD Standard fuse pattern (.STD).

Format :

FROMJED *parameter1 parameter2 parameter3*
parameter1 : PLD full device name
parameter2 : Input filename (JEDEC Format)
parameter3 : Output filename (Standard format)

The test vector portion of the input JEDEC file will be ignored by this module.

Example :

FROMJED PLHS153 examp.JED exp.STD
In the above example, the JEDEC information (examp.JED) is converted into PLHS153 fuse pattern in Standard format which is stored in exp.STD.

8.3 TOSIG

This module creates a SIG. H/L correspondence of the specified standard file (.STD).

Format :

TOSIG *parameter1 parameter2*
parameter1 : Input Fuse Pattern filename (Standard Format)
parameter2 : Output filename (SIG. H/L Format)

Example :

TOSIG examp.STD examp.SIG
In the above example, the SIG H/L correspondence of the fuse pattern (examp.STD) is stored in examp.SIG.

8.4 FROMSIG

This module converts a SIG. H/L file into a PLD Standard fuse file (.STD).

Format :

FROMSIG *parameter1 parameter2 parameter3*
parameter1 : PLD full device name
parameter2 : Input filename (SIG. H/L Format)
parameter3 : Output filename (Standard format)

Example :

FROMSIG PLS105 examp.SIG exp.STD
In the above example, the SIG H/L information (examp.SIG) is converted into PLS105 fuse pattern in Standard format which is stored in exp.STD.

APPENDIX A

Examples

9. Appendix A

The examples used in this chapter were chosen only to illustrate the capabilities of different sections of BLAST for different products. A brief description of the design examples are given before different sections of BLAST are presented.

9.1 Data Flow Controller Using PLS153

A Data Flow Controller (DFC), consists of a network which controls the direction and destination of data among clusters of networks and/or systems, (i.e., CPUs and memory banks or ports of two different systems).

The DFC is to be able to direct the data from any port on one side, to any port on the other side of the network simultaneously, providing that there is only one single link between two ports. The direction of flow is also controlled through the DFC.

Consider two two-bit I/O ports as two ports of two different networks which are capable of transferring data from one network to another network in different configurations and directions. The flow of data (i.e., from network A to network B) is controlled by an asynchronous signal (DIR), while the transfer of data can be done in parallel mode (PAR) or cross mode (CROSS). The PAR and CROSS are chosen to be synchronous signals. Figure 9-1 depicts such a network. A D-Latch is used to generate the PAR or CROSS signals.

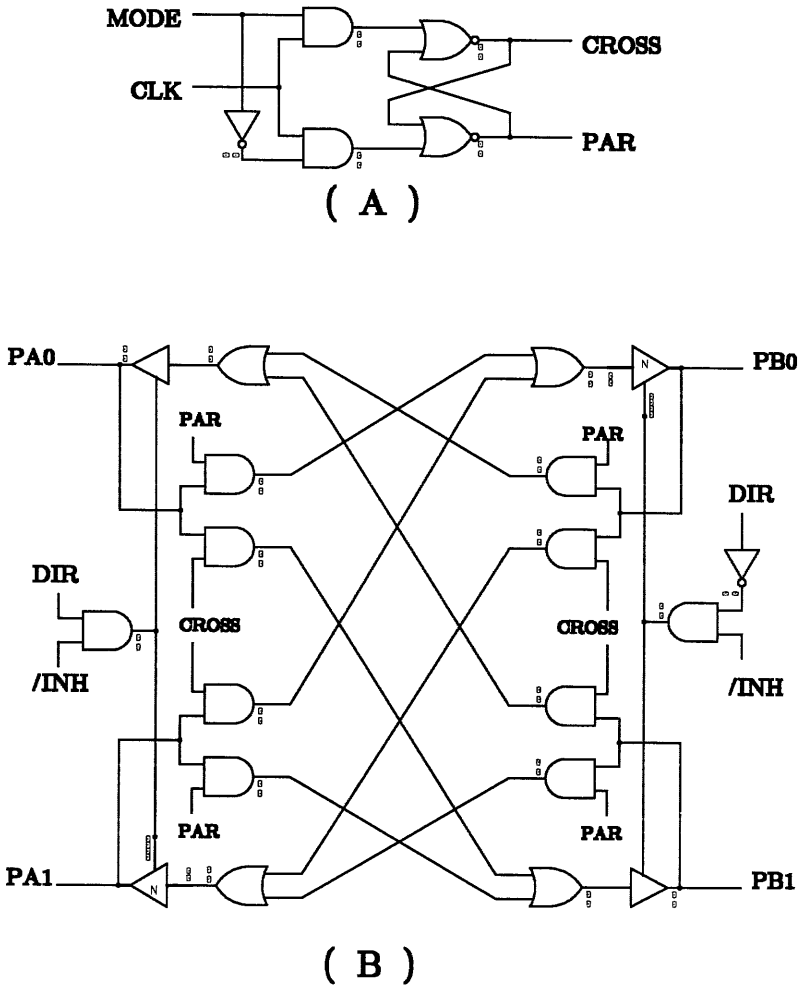


Figure 9–1 Two 2–bit Port Controller

```
################## P I N L I S T ##################

      LABEL     ** FNC **PIN --------- PIN** FNC **     LABEL
D               ** I   **  1-|       |-20 ** +5V **VCC
CLK             ** I   **  2-|       |-19 ** /0  **QTRUE
N/C             ** I   **  3-|   P   |-18 ** /0  **QCOMPL
N/C             ** I   **  4-|   L   |-17 ** B   **N/C
DIR             ** I   **  5-|   S   |-16 ** B   **N/C
N/C             ** I   **  6-|   1   |-15 ** B   **N/C
/INH            ** I   **  7-|   5   |-14 ** B   **PB1
N/C             ** I   **  8-|   3   |-13 ** B   **PB0
N/C             ** I   **  9-|       |-12 ** B   **PA1
GND             ** 0V  ** 10-|       |-11 ** B   **PA0
                           ---------

@DEVICE TYPE                        ******PLS153******

@DRAWING                            ***DATA FLOW 1 ****

@REVISION                           ******REV. 1 ******

@DATE                               ****FEB 28, 87*****

@SYMBOL                             *******DFC*********

@COMPANY                            *****SIGNETICS*****

@NAME                               *JOE USER**

@DESCRIPTION

            **********************************
            *                                *
            *     DATA FLOW CONTROLLER       *
            *                                *
            * * * * * * * * * * * * * * * * *
```

Figure 9–2 Data Flow Controller

Examples

```
      if D = H then                  D  CLK  I  CROSS   PAR
         CROSS connection          -----------I--------------
                                    H       I    H      L
      if D = L then                 L       I    L      H
         PARallel connection        X    L  I  CROSS   PAR
                                         I
```

2 - Asynchronous DIRection input.

3 - An INHibit line, which disables the crosstalking between the two ports when it goes to high state. (/INH)

```
/INH    DIR    CROSS    PART    I    DATA TRANSFER
 ------------------------------I----------------------------
  L      X       X       X      I    NO CONNECTION
                                I
  H      H       H       L      I   PB0  T0  PA1, PB1 TO PA0
                                I
  H      H       L       H      I   PB0 TO PA0, PB1 TO PA1
                                I
  H      L       H       L      I   PA0 TO PB1, PA1 TO PB0
                                I
  H      L       L       H      I   PA0 TO PB0, PA1 TO PB1
                                I
```

```
@COMMON PRODUCT TERM

"The outputs of the D - Latch are given common
symbols as:  "

     CROSS = QTRUE;
     PAR = QCOMPL;

@I/O DIRECTION

"If the flow direction is PORT A ---  PORT B
then :"

     D3 = /INH * /DIR:
     D4 = /INH * /DIR;
```

```
"If the flow direction is PORT B ---  PORT A
then : "

     D1 = /INH * DIR;
     D2 = /INH * DIR;

"OUTPUT OF D-LATCH:  "
     D8=1;
     D9=1;

@LOGIC EQUATION

"--------------------------
  DEFINING THE D-LATCH
--------------------------"

QTRUE =  /(CLK * /D + QCOMPL);
QCOMPL =  /(CLK * D + QTRUE);

"--------------------------------------
   DEFINING THE DATA PATHS
--------------------------------------'

PA0 = PB0 * PAR + PB1 * CROSS;
PA1 = PB1 * PAR + PB0 * CROSS;

PB0 = PA0 * PAR + PA1 * CROSS;
PB1 = PA1 * PAR + PA0 * CROSS:

"Note that:     PAR = QCOMPL
            &   CROSS + QTRUE "
```

Examples

```
   Cust/Project -                          *JOE USER*
   Date         -                          ****FEB 28,87 ****
   Rev/I.  D.   -                          ******REV. 1 *****
   PLS153                                !      POLARTY      !
---                                      !-------------------
  T  !                                   !L:L:H:H:H:H:H:H:H:H!
  E!-----------------------------------------------------------
  R!        I       !        B(i)       !        B(o)       !
  M!-----------------------------------------------------------
---!7_6_5_4_3_2_1_0!9_8_7_6_5_4_3_2_1_0!9_8_7_6_5_4_3_2_1_0!
  0!- - - -,- - H L!- -,- - - -,- - - -!A .,A A A .,. . . A!
  1!- - - -,- - - -!- H,- - - -,- - - -!A .,A A A .,. . . A!
  2!- - - -,- - H H!- -,- - - -,- - - -!. A,A A A .,. . . A!
  3!- - - -,- - - -!H -,- - - -,- - - -!. A,A A A .,. . . A!
  4!- - - -,- - - -!- H,- - - -,H - - -!. .,A A A .,. . A A!
  5!- - - -,- - - -!H -,- - - H,- - - -!. .,A A A .,. . A A!
  6!- - - -,- - - -!- H,- - - H,- - - -!. .,A A A .,. A . A!
  7!- - - -,- - - -!H -,- - - -,H - - -!. .,A A A .,. A . A!
  8!- - - -,- - - -!- H,- - - -,- - H -!. .,A A A .,A . . A!
  9!- - - -,- - - -!H -,- - - -,- H - -!. .,A A A .,A . . A!
 10!- - - -,- - - -!- H,- - - -,- H - -!. .,A A A A,. . . A!
 11!- - - -,- - - -!H -,- - - -,- - H -!. .,A A A A,. . . A!
 12!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 13!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 14!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 15!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 16!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 17!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 25!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 26!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 27!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 28!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 29!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 30!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 31!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!A A,A A A A,A A A A!
 D9!- - - -,- - - -!- -,- - - -,- - - -!
 D8!- - - -,- - - -!- -,- - - -,- - - -!
 D7!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!
 D6!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!
 D5!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!
 D4!- L - L,- - - -!- -,- - - -,- - - -!
 D3!- L - L,- - - -!- -,- - - -,- - - -!
 D2!- L - H,- - - -!- -,- - - -,- - - -!
 D1!- L - H,- - - -!- -,- - - -,- - - -!
 D0!0 0 0 0,0 0 0 0!0 0,0 0 0 0,0 0 0 0!
```

```
N I N D N N C D Q Q N N N P P P P N Q Q N N N P P P P N
/ N / I / / L  T C / / / B B A A / T C / / / B B A A /
C H C R C C K  R O C C C 1 0 1 0 C R O C C C 1 0 1 0 C
               U M               U M
               E P               E P
                 L                 L.
```

9.2 Timer/Decoder Using PLS159

The Timer/Decoder consists of a 4-stage binary synchronous counter and a Decimal Count Detector (Decoder). The flip-flop outputs are buffered to produce active low binary count outputs. The outputs are used to drive the binary LED display.

The flip-flop outputs are also buffered to produce the "Q" and "/Q" which are then decoded to generate a 4-bit Decimal Count Detector at 0, 5, 20, and 15. The decoder output is to drive a 4-bit LED display; thus an active low output is used. Also the output buffers are controlled by the output enable signal, DECON.

The counter may be directly preset to produce a count of 1111 (15). The binary display should be all lit and the decimal display for 15 will be lit, as well. Again, the counter may be directly reset to produce a count of 0000 (0). The binary display will be dark and the decimal display for 0 will be lit.

The logic design of the Timer/Decoder is depicted in Figure 9-3.

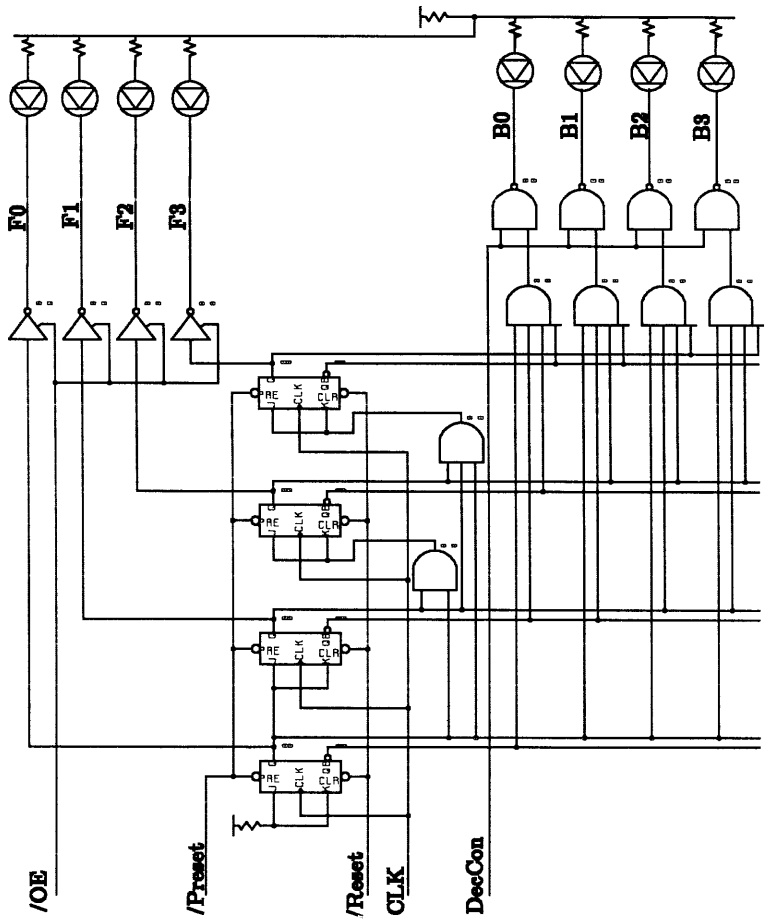


Figure 9–3 4 Bit Timer/Decoder

```
LABEL    ** FNC  **PIN          _______   PIN ** FNC **LABEL
CLK        ** CK   **    1-|           |  -20  ** +5V **VCC
/PRESA     ** I    **    2-|           |  -19  ** B   **N/C
/RESA      ** I    **    3-|     P     |  -18  ** B   **N/C
DECCON     ** I    **    4-|     L     |  -17  ** B   **N/C
N/C        ** I    **    5-|     S     |  -16  ** B   **N/C
B0         ** /B   **    6-|     1     |  -15  ** O   **F3
B1         ** /B   **    7-|     5     |  -14  ** O   **F2
B2         ** /B   **    8-|     9     |  -13  ** O   **F1
B3         ** /B   **    9-|           |  -12  ** O   **F0
GND        ** OV   **   10-|___________|  -11  ** /OE **/OE
```

Figure 9–4 Timer/Decoder PINLIST (TIMER.PIN)

```
@DEVICE TYPE
                                        ******PLS159******
@DRAWING
                                        *******Timer******
@REVISION
                                        **********A*******
@DATE
                                        *******2/28/87****
@SYMBOL
@COMPANY
                                        *****Signetics****
@NAME
                                        *****JOE USER****
@DESCRIPTION
```

Figure 9–5 Boolean Equation Entry File (TIMER.BEE)

The Timer is a 4 stage synchronous binary counter with binary and selected decimal outputs. It is designed on one PLS159 PLD device and requires external signals for clocking controls. Its outputs may be used to drive

LED displays.

(TIMER.BEE File)

@COMMON PRODUCT TERM

"No common product term was needed for this design."

Examples

@COMPLEMENT ARRAY

"The Complement Array acts as the Binary Count Controller."

```
/C = /(/DECCON) ;
```

@FLIP FLOP CONTROL

"The registers used in this design are always of the J-K type. Thus: "

```
FC = 1;
```

@FLIP FLOP MODE

"All the registers are controlled by the FC line."

"Note that the registers not used in this design are not to be defined. And the default to the virgin state (intact)."

```
M0, M1, M2, M3 = FC;
```

@I/O DIRECTION

"The I/O pins are controlled by the complement array (/C)."

```
D0 = /C;
D1 = /C;
D2 = /C;
D3 = /C;
```

@OUTPUT ENABLE

"The four bit register outputs are always ENABLED."

```
EA = 0;
```

"This information is unnecessary since the PLS159 F/F outputs are by default enabled".

(TIMER.BEE file)

```
@REGISTER LOAD

"Register load function for this design is not
 required since the only registers used are
 defined to be output in the PINLIST page."

@ASYNCHRONOUS PRESET/RESET

"The registers' PA (preset) and RA (reset) lines are
 externally controlled by the complements of the
 /PRESA and /RESA signals."

        PA = PRESA;
        RA = RESA;

@LOGIC EQUATION

"The Toggle option for the registers are used to
 form a 4-bit binary counter.  Note that the
 negation of the labels are used on the LHS of
 the equations."

/F0: T = 1;
/F1: T = /F0;
/F2: T = /F0 * /F1;
/F3: T = /F0 * /F1 * /F2;

"Decimal count detector is designed using the
 B's outputs.  Note that the expression on the
 RHS of the equation are negated since these
 outputs are defined to be Active Low in the
 PINLIST page."

B0 = /(F0  *  F1 *  F2 *  F3);
B1 = /(F0  * /F1 * /F2 *  F3);
B2 = /(F0  * /F1 *  F2 * /F3);
B3 = /(/F0 * /F1 * /F2 * /F3);
```

(TIMER.STD File)
Program Table for Timer/PLS159 as promptd by BLAST or viewed by the Program Table Editor (PTE).

Examples

```
   Cust/Project -  ***JOE USER******
   Date         -                          ******2/28/87******
   Rev/I.  D.   -                          *********A********
   PLS159              ! F/F TYPE      ! E(b)=  !E(a)= !POLARTY!
---                    ----------------!-----------------------
  T  !                 !A:A:A:A:A:A:A:A!   0    !   .  !L:L:L:L!
  E!-------------------------------------------------------------
  R  !   I   !  B(i) !      Q(p)      !      Q(n)      !  B(o) !
  M!C!-------------------------------------------------------------
---!_!3_2_1_0!3_2_1_0!7_6_5_4_3_2_1_0!7_6_5_4_3_2_1_0!3_2_1_0!
  0!A!- L - -!- - - -!- - - -,- - - -!0 0 0 0,- - - -!. . . .!
  1!-!- - - -!- - - -!- - - -,- - - -!0 0 0 0,- - - 0!. . . .!
  2!-!- - - -!- - - -!- - - -,- - - H!0 0 0 0,- - 0 -!. . . .!
  3!-!- - - -!- - - -!- - - -,- - H H!0 0 0 0,- 0 - -!. . . .!
  4!-!- - - -!- - - -!- - - -,- H H H!0 0 0 0,0 - - -!. . . .!
  5!-!- - - -!- - - -!- - - -,L L L L!0 0 0 0,- - - -!. . . A!
  6!-!- - - -!- - - -!- - - -,L H L H!0 0 0 0,- - - -!. . A .!
  7!-!- - - -!- - - -!- - - -,H L H L!0 0 0 0,- - - -!. A . .!
  8!-!- - - -!- - - -!- - - -,H H H H!0 0 0 0,- - - -!A . . .!
  9!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 10!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 11!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 12!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 13!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 14!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 15!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 16!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 17!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 26!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 27!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 28!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 29!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 30!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 31!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 Fc!-!- - - -!- - - -!- - - -,- - - -!
 Pb!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Rb!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Lb!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Pa!-!- - - L!- - - -!- - - -,- - - -!
 Ra!-!- - L -!- - - -!- - - -,- - - -!
 La!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 D3!.!- - - -!- - - -!- - - -,- - - -!
 D2!-!- - - -!- - - -!- - - -,- - - -!
 D1!-!- - - -!- - - -!- - - -,- - - -!
 D0!-!- - - -!- - - -!- - - -,- - - -!
```

```
N B / / B B B B N N N N F F F F N N N N F F F F B B B B
/ I R P 3 2 1 0 / / / / 3 2 1 0 / / / / 3 2 1 0 3 2 1 0
C N E R         C C C C         C C C C
C S E
O A S
  N   A
```

TABLE 9-1 Program Table for Timer .PLS159

9.3 State Equation Entry Example

In this section, the design and implementation of a state machine depicted in Figure 9-6 is shown. This state machine requires 4-Inputs (D0 to D3), 4-State bits (ST0 to ST3), 9-Synchronous Outputs (A0 to A8), and 3-Asynch. Outputs (ERROR, BUSY, READY). For this machine one can use two PLS159 as shown in Figure 9-6.

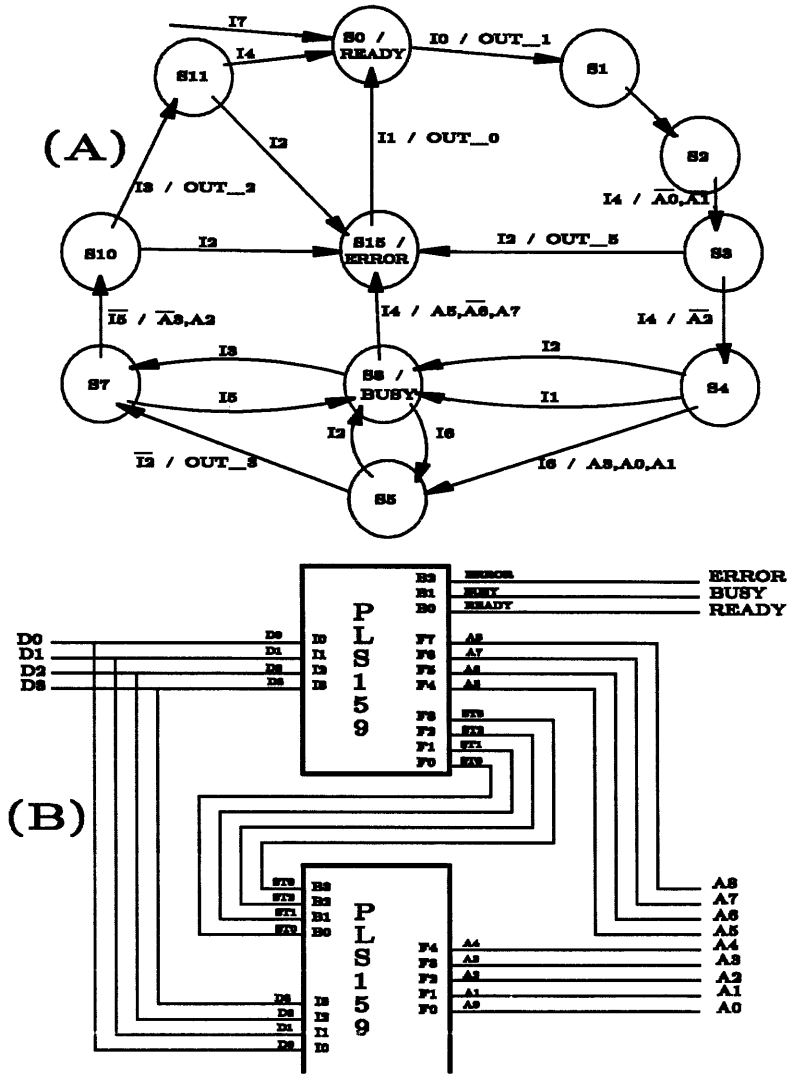


Figure 9–6 State Machine Design Example

```
 ********************MYDSGN**************************

 @Device selection

mydsgnl / PLS159
mydsgn2 / PLS159  "Note that this filename is used in the
                   .BEE, .PIN & .STD file"

@INPUT VECTORS

[id0, id1, id2, id3]
In_Null = ----B;
i0 = 0h;
i1 = 1h;
i2 = 2h;
i3 = 3h;
i4 = 4h;
i5 = 5h;
i6 = 6h;
i7 = 7h;

@OUTPUT VECTORs
"--------------------"

[a0, a1, a2, a3, a4, a5, a6, a7, a8]
out_0' = 0000 0000 0b;
out_1' =  1111 1111 1b;
out_2' =  1111 0000 0b;
out_3' =  ---1 100- -b;
out_5' =  10-- --1- 1b;
out_6' =  -110 01-- 1b;

@STATE VECTORS
"--------------------"

[st3, st2, st1, st0]

st_null = - - - - b;
s0 = 0h;
s1 = 1h;
s2 = 2h;
s3 = 3h;
s4 = 4h;
s5 = 5h;
s6 = 6h;
s7 = 7h;
s8 = 8h;
```

```
     sl0 = ah;
     sll = bh;
     sl5 = fh;     @transitions
"_________________"

WHILE [ st_null ]
     IF [ i7 ] THEN [ s0 ]

WHILE [ s0 ] : [ready]
     IF [ i0 ] THEN [ sl ] with [out_l' ]

WHILE [ sl ]
     IF [ IN_Null ] THEN [ s2 ]

WHILE [ s2 ]
     IF [ i4 ] THEN [ s3 ] with [ /a0', /al' ]

WHILE [ s3 ]
     IF [ I2 ] THEN [ sl5 ] with [ out_5' ]
     if [i4 ] then [ s4 ] with [ /a2' ]

WHILE [ s4 ]
     IF [ I2 ] THEN [ s8 ]
     IF [ Il ] THEN [ s8 ]
     IF [ I6 ] THEN [ s5 ] with [ A3, A0, Al ]

WHILE [ s5 ]
     IF [ I2 ] THEN [ s8 ]
     ELSE [ s7 ] with [ out_3']

WHILE [ s7 ]
     IF [ I5 ] THEN [ s8 ]
     ELSE [ sl0 ] with [ /a3', a2' ]

WHILE [ s8 ] : [busy]
 CASE
     [ I4 ] :: [ sl5 ] with [ a5', /a6', a7' ]
     [ i3 ] ::  [ s7 ]
     [ i6 ] ::   [ s5 ]
 endcase

WHILE [ sl0 ]
     if [ I3 ] then [ sll ] with [ OUT_2' ]
     if [ i2 ] then [ sl5 ]

WHILE [ sl5 ] : [ error ]
```

```
     IF [Il ] THEN [ s0 ] with [ out_0' ]

WHILE [ sll ]
     if [ i4 ] then [ s0 ]
     if [ i2 ] then [ sl5 ]
```

Figure 9–7 MYDSGN

```
##################  PINLIST ##################


      LABEL     ** FNC **PIN  _______  PIN** FNC **     LABEL
 N/C            ** CK  **  1-|       |-20 ** +5V **VCC
 ID0            ** I   **  2-|       |-19 ** O   **A8
 ID1            ** I   **  3-|   P   |-18 ** O   **A7
 ID2            ** I   **  4-|   L   |-17 ** O   **A6
 ID3            ** I   **  5-|   S   |-16 ** O   **A5
 READY          ** O   **  6-|   1   |-15 ** O   **ST3
 BUSY           ** O   **  7-|   5   |-14 ** O   **ST2
 ERROR          ** O   **  8-|   9   |-13 ** O   **ST1
 N/C            ** /B  **  9-|       |-12 ** O   **ST0
 GND            ** 0V  ** 10-|_______|-11 ** /OE **N/C
```

Figure 9–8 MYDSGN1 PINLIST

```
################## PINLIST ##################


      LABEL     ** FNC **PIN  _______  PIN** FNC **     LABEL
 N/C            ** CK  **  1-|       |-20 ** +5V **VCC
 ID0            ** I   **  2-|       |-19 ** B   **N/C
 ID1            ** I   **  3-|   P   |-18 ** B   **N/C
 ID2            ** I   **  4-|   L   |-17 ** B   **N/C
 ID3            ** I   **  5-|   S   |-16 ** O   **A4
 ST0            ** I   **  6-|   1   |-15 ** O   **A3
 ST1            ** I   **  7-|   5   |-14 ** O   **A2
 ST2            ** I   **  8-|   9   |-13 ** O   **A1
 ST3            ** I   **  9-|       |-12 ** O   **A0
 GND            ** 0V  ** 10-|_______|-11 ** /OE **N/C
```

Figure 9–9 MYDSGN2 PINLIST

```
@DEVICE TYPE
 PLS159
@DRAWING
@REVISION
```

```
@DATE
@SYMBOL
@COMPANY
@NAME
@DESCRIPTION
@COMMON PRODUCT TERM
@COMPLEMENT ARRAY
@I/O DIRECTION
@FLIP FLOP CONTROL
@OUTPUT ENABLE
@REGISTER LOAD
@ASYNCHRONOUS PRESET/RESET
@FLIP FLOP MODE
 M0,M1,M2,M3,M4,M5,M6,M7 = 1;  " Set the registers to J-K "
@LOGIC EQUATION
```

Figure 9–10 Example of MYDSGN1

```
@DEVICE TYPE
 PLS159
@DRAWING
@REVISION
@DATE
@SYMBOL
@COMPANY
@NAME
@DESCRIPTION
@COMMON PRODUCT TERM
@COMPLEMENT ARRAY
@I/O DIRECTION     @OUTPUT POLARITY
@FLIP FLOP CONTROL
@OUTPUT ENABLE
@REGISTER LOAD
@ASYNCHRONOUS PRESET/RESET
@FLIP FLOP MODE
 M0,M1,M2,M3,M4 = 1;  "Set the registers to J-K"
@LOGIC EQUATION
```

Figure 9–11 Example of MYDSGN2

```
                ***************MYDSGN1***************
   Cust/Project -
   Date         -
   Rev/I.  D.
PLS159                ! F/F TYPE      ! E(b)= ! E(a)= !POLARTY!
---                   ----------------!-----------------------
T  !                  !.:.:.:.:.:.:.:.!   0   !   0   !L:H:H:H!
E  !-------------------------------------------------------------
R  !    I    ! B(i) !      Q(p)      !      Q(n)      ! B(o) !
M  !C!-------------------------------------------------------------
---!_!3_2_1_0!3_2_1_0!7_6_5_4_3_2_1_0!7_6_5_4_3_2_1_0!3_2_1_0!
  0!-!H H H L!- - - -!- - - -,- - - -!. . . .,H H H H!A . . .!
  1!-!- - - -!- - - -!- - - -,H H H H!. . . .,. . . .!A . . A!
  2!-!L L L L!- - - -!- - - -,H H H H!L L L L,H H H H!A . . .!
  3!-!- - - -!- - - -!- - - -,H H H L!. . . .,H H L H!A . . .!
  4!-!L L H L!- - - -!- - - -,H H L H!. . . .,H H L L!A . . .!
  5!-!L H L L!- - - -!- - - -,H H L L!L . L .,L L L L!A . .  !
  6!-!L L H L!- - - -!- - - -,H H L L!. . . .,H L H H!A . . .!
  7!-!L H L L!- - - -!- - - -,H L H H!. . . .,L H H H!A . . .!
  8!-!H L L L!- - - -!- - - -,H L H H!. . . .,L H H H!A . . .!
  9!-!L H H L!- - - -!- - - -,H L H H!. . . .,H L H L!A . . .!
 10!A!L H L L!- - - -!- - - -,H L H L!. . . .,L H H H!A . . .!
 11!-!- - - -!- - - -!- - - -,H L H L!. . H H,H L L L!A . . .!
 12!A!H L H L!- - - -!- - - -,H L L L!. . . .,L H H H!A . . .!
 13!-!- - - -!- - - -!- - - -,H L L L!. . . .,L H L H!A . . .!
 14!-!- - - -!- - - -!- - - -,L H H H!. . . .,. . . .!A . A .!
 15!-!L L H L!- - - -!- - - -,L H H H!. L H L,L L L L!A . . .!
 16!-!H H L L!- - - -!- - - -,L H H H!. . . .,H L L L!A . . .!
 17!-!L H H L!- - - -!- - - -,L H H H!. . . .,H L H L!A . . .!
 18!-!H H L L!- - - -!- - - -,L H L H!H H H H,L H L L!A . . .!
 19!-!L H L L!- - - -!- - - -,L H L H!. . . .,L L L L!A . . .!
 20!-!L H L L!- - - -!- - - -,L L L L!. . . .,. . . .!A A . .!
 21!-!H L L L!- - - -!- - - -,L L L L!H H H H,H H H H!A . . .!
 22!-!L L H L!- - - -!- - - -,L H L L!. . . .,H H H H!A . . .!
 23!-!L H L L!- - - -!- - - -,L H L L!. . . .,L L L L!A . . .!
 24!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 31!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 Fc!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Pb!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Rb!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Lb!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Pa!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 La!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 D3!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 D2!.!- - - -!- - - -!- - - -,- - - -!
 D1!-!- - - -!- - - -!- - - -,- - - -!
```

Examples

```
DO!-!- - - -!- - - -!- - - -,- - - -!
     I I I I N E B R A A A A S S S S A A A A S S S S N E B R
     D D D D / R U E 8 7 6 5 T T T T 8 7 6 5 T T T T / R U E
     3 2 1 0 C R S A       3 2 1 0       3 2 1 0 C R S A
               O Y D                               O Y D
               R  Y                                R  Y
```

```
          ***************MYDSGN2***************
Cust/Project -
Date         -
Rev/I.  D.   -
PLS159                ! F/F TYPE      ! E(b)= ! E(a)= !POLARTY!
---                   ----------------!-----------------------
T  !                  !A:A:A:.:.:.:.:.!   0   !   0   !L:L:L:L!
E  !-------------------------------------------------------------
R  !    I    !  B(i) !      Q(p)      !      Q(n)      !  B(o) !
M  !C!-------------------------------------------------------------
---!_!3_2_1_0!3_2_1_0!7_6_5_4_3_2_1_0!7_6_5_4_3_2_1_0!3_2_1_0!
  0!-!L L L L!L L L L!- - - -,- - - -!0 0 0 L,L L L L!A A A A!
  1!-!L L H L!L L H L!- - - -,- - - -!0 0 0 .,. . H H!A A A A!
  2!-!L H L L!L L H H!- - - -,- - - -!0 0 0 .,. . H L!A A A A!
  3!-!L L H L!L L H H!- - - -,- - - -!0 0 0 .,. H . .!A A A A!
  4!-!L H H L!L H L L!- - - -,- - - -!0 0 0 .,L . L L!A A A A!
  5!A!L H L L!L H L H!- - - -,- - - -!0 0 0 L,L . . .!A A A A!
  6!-!- - - -!L H L H!- - - -,- - - -!0 0 0 L,L . . .!A A A A!
  7!A!H L H L!L H H H!- - - -,- - - -!0 0 0 .,. . . .!A A A A!
  8!.!- - - -!L H H H!- - - -,- - - -!0 0 0 .,H L . .!A A A A!
  9!-!H H L L!H L H L!- - - -,- - - -!0 0 0 H,L L L L!A A A A!
 10!-!H L L L!H H H H!- - - -,- - - -!0 0 0 H,H H H H!A A A A!
 11!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 12!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 13!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 14!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 15!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 16!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 17!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 18!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 25!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 26!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 27!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 28!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 29!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 30!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 31!0!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!0 0 0 0,0 0 0 0!A A A A!
 Fc!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Pb!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Rb!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Lb!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 Pa!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 La!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 D3!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 D2!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
 D1!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
```

```
D0!.!0 0 0 0!0 0 0 0!0 0 0 0,0 0 0 0!
     I I I I S S S S N N N A A A A A N N N A A A A A S S S S
     D D D D T T T T / / / 4 3 2 1 0 / / / 4 3 2 1 0 T T T T
     3 2 1 0 3 2 1 0 C C C         C C C         3 2 1 0
```

9.4 PML EXAMPLE

The following example illustrates a basic approach to verifying a design implemented in PML architecture.

You will find seven equations in this example, although one (Z) is trivial since it reduces to one variable.

The equation for "X" has a bracket nested within another bracket.

```
**********************PINLIST*********************|
         Left                                Right      |      L A B E L
LABEL** FNC **  PIN _________ PIN  ** FNC **  Label|Length-12 Char,Max
VCC   ** +5V **   8-|         |-46  ** +5V **VCC    |A-Z, 0-9, /; -, _
A     ** I   **   9-|         |-45  ** I   **N/C    |  N/C - NOT USED
B     ** I   **  10-|         |-44  ** I   **N/C    |
C     ** I   **  11-|    P    |-43  ** I   **N/C    | F U N C T I O N
D     ** I   **  12-|    L    |-42  ** I   **N/C    |
E     ** I   **  13-|    H    |-41  ** I   **N/C    |
F     ** I   **  14-|    S    |-40  ** I   **M      |
G     ** I   **  15-|    5    |-39  ** I   **P      |
X     ** O   **  16-|    0    |-38  ** /O  **T      |
N/C   ** B   **  17-|    1    |-37  ** /0  **N/C    |
S     ** B   **  18-|         |-36  ** 0   **Z      |
N/C   ** O   **  19-|         |-35  ** 0   **N/C    |
GND   ** OV  **  20-|_________|-34  ** OV  **GND    |
```

Figure 9-12 First Half of PLHS501 Pinlist

BLAST will assemble the PLHS501 pin list and Boolean equations (.BEE file) into the .STD file in such a format as to automatically invoke the fuse table editor (FTE) through the "PTE" selection of the AMAZE Main Menu (the PLHS501 fuse table is shown on the next six pages as it would be viewed/edited with FTE). The default pattern of 32 dashes (-) followed by 72 dots (.) symbolize the 32 input

buffers (H,L,–) and the 72 NAND gate feedback (A,.) which are all of the inputs to an individual NAND gate in the device. (Refer to PTE/FTE sections for more detail.)

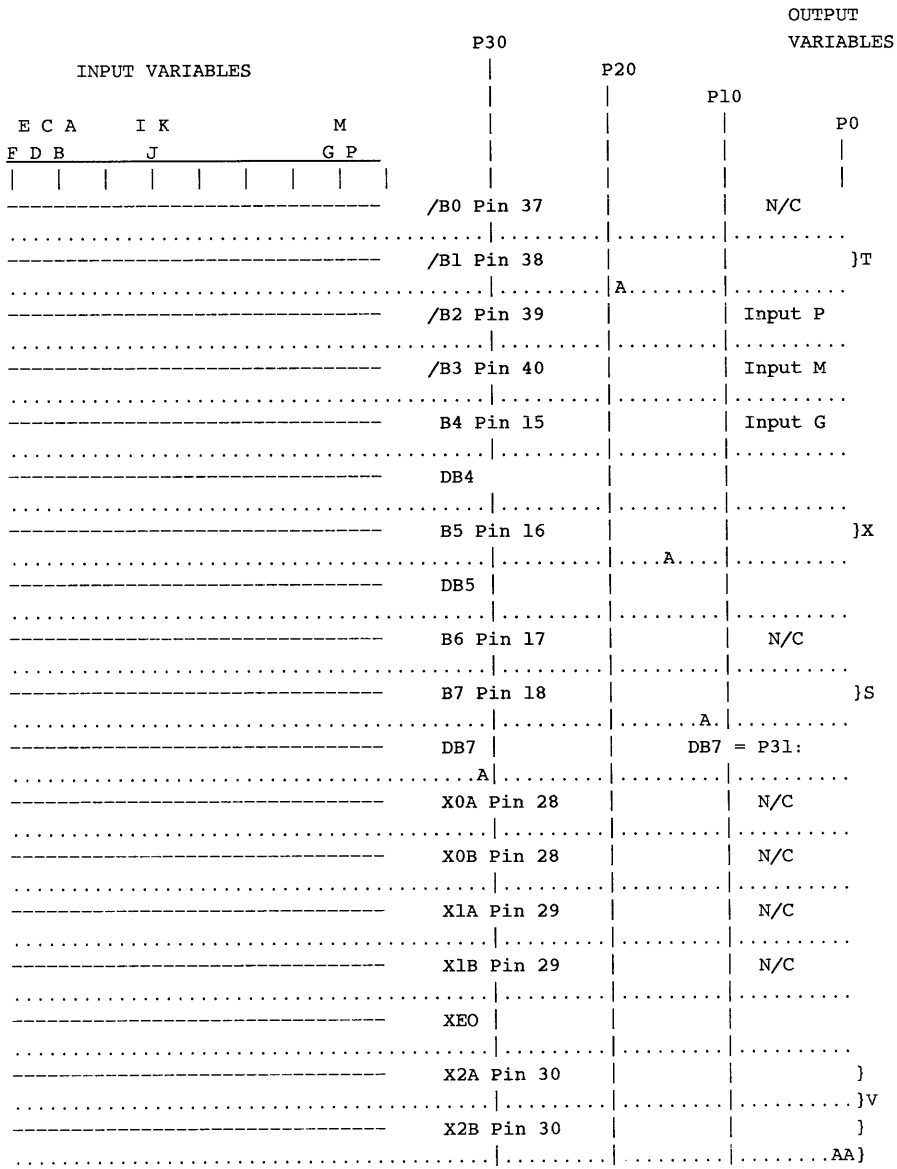
```
**********PINLIST**********|
      BOTTOM                        TOP       |     L A B E L
LABEL** FNC ** PIN  _______  PIN ** FNC **  LABEL|LENGTH- 12 CHAR,MAX
W    ** 0   **  21-|       |- 7  ** I **N/C      |A-Z, 0-9, /; -, _
N/C  ** 0   **  22-|       |- 6  ** I **N/C      |  N/C - NOT USED
N/C  ** 0   **  23-|       |- 5  ** I **N/C      |
Y    ** /0  **  24-|   P   |- 4  ** I **N/C      | F U N C T I O N
N/C  ** /0  **  25-|   L   |- 3  ** I **N/C      |
N/C  ** /0  **  26-|   H   |- 2  ** I **I        |
N/C  ** /0  **  27-|   S   |- 1  ** I **J        |
N/C  ** 0   **  28-|   5   |-52  ** I **K        |
N/C  ** 0   **  29-|   0   |-51  ** I **N/C      |
V    ** 0   **  30-|   1   |-50  ** I **N/C      |
N/C  ** 0   **  31-|       |-49  ** I **N/C      |
N/C  ** O   **  32-|       |-48  ** I **N/C      |
N/C  ** O   **  33-|_______|-47  ** I **N/C      |
```

Figure 9–13 Second Half of PLHS501 Pinlist

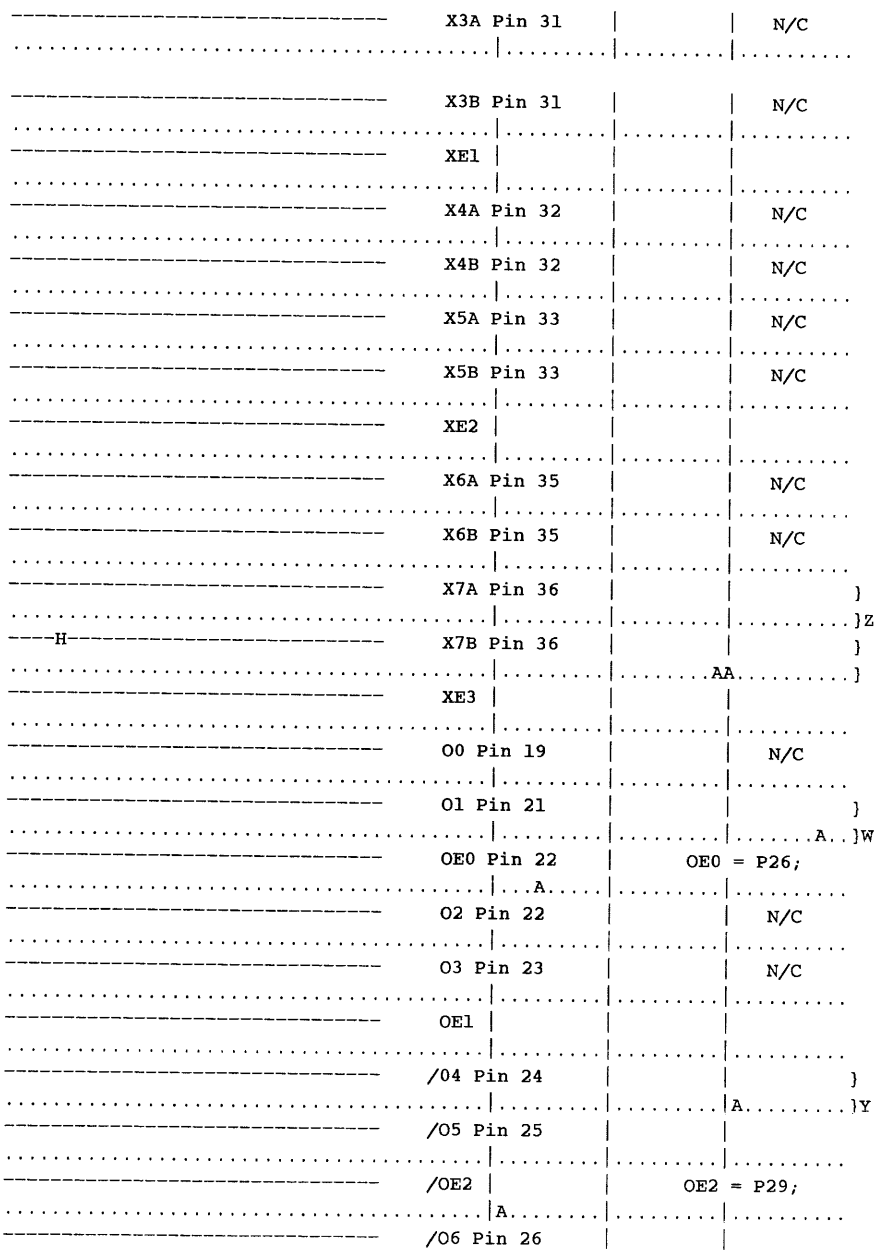
```
File Name:  TTS10-12

@DEVICE TYPE
PLHS501
@DRAWING
@REVISION
1
@DATE
2/28/87
@SYMBOL
@COMPANY
Signetics
@NAME
JOE USER
@DESCRIPTION
@COMMON PRODUCT TERM
cpt1 = a*b*c;
cpt2 = b*d*g;
@INTERNAL NODE
@I/O DIRECTION
oe0 = P*/A*/F + I*M;
oe2 = A*B*/E + /P + M;
db7 = i*/J + /K + P;
@I/O STEERING
s1 = 0;
s2 = 1;
s3 = 1;
@LOGIC EQUATION
V = E*cpt1 + D*G ;
W = F + A*C + cpt2 ;
X = A + [B + cpt1 + E*[C + a*cpt2]]+b*c*d*e ;
Y = /( A + cpt1) ;
Z = /( B + cpt2) ;
s = /C*F + /G*B ;
t = /( /a*/j*b + /k*f +m ) ;
```

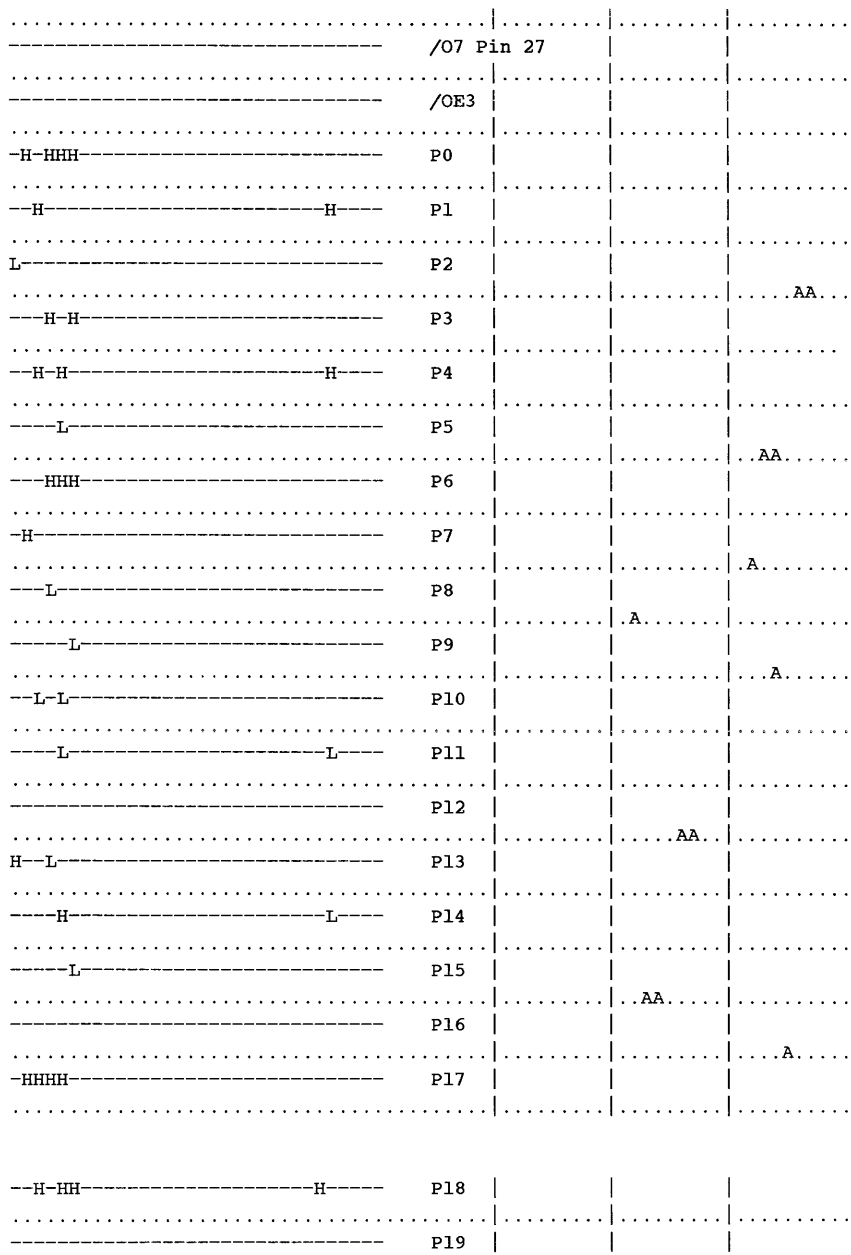
Figure 9-14 .BEE FILE FOR TTS10-12

TEST CASE TTS10-12
FUSE TABLE

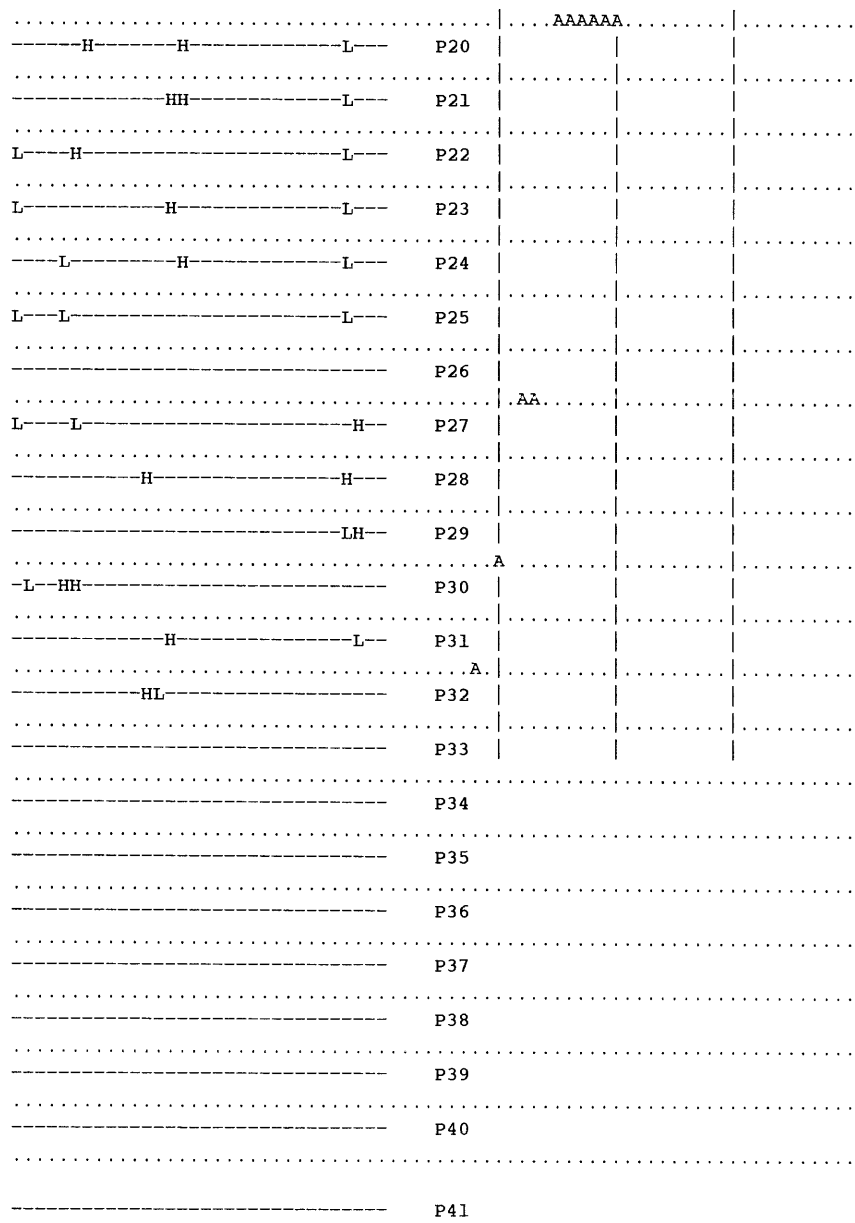


Examples


X3A Pin 31
N/C
X3B Pin 31
N/C
XE1
X4A Pin 32
N/C
X4B Pin 32
N/C
X5A Pin 33
N/C
X5B Pin 33
N/C
XE2
X6A Pin 35
N/C
X6B Pin 35
N/C
X7A Pin 36
X7B Pin 36
Z
H
XE3
O0 Pin 19
N/C
O1 Pin 21
W
OE0 Pin 22
OE0 = P26;
O2 Pin 22
N/C
O3 Pin 23
N/C
OE1
/O4 Pin 24
Y
/O5 Pin 25
/OE2
OE2 = P29;
/O6 Pin 26


/O7 Pin 27
/OE3
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16
P17
P18
P19

Examples


P20
P21
P22
P23
P24
P25
P26
P27
P28
P29
P30
P31
P32
P33
P34
P35
P36
P37
P38
P39
P40
P41

```
------------------------------  P42
.....................................................................
------------------------------  P43
.....................................................................
------------------------------  P44
.....................................................................
------------------------------  P45
.....................................................................
------------------------------  P46
.....................................................................
------------------------------  P47
.....................................................................
------------------------------  P48
.....................................................................
------------------------------  P49
.....................................................................
------------------------------  P50
.....................................................................
------------------------------  P51
.....................................................................
------------------------------  P52
.....................................................................
------------------------------  P53
.....................................................................
------------------------------  P54
.....................................................................
------------------------------  P55
.....................................................................
------------------------------  P56
.....................................................................
------------------------------  P57
.....................................................................
------------------------------  P58
.....................................................................
------------------------------  P59
.....................................................................
------------------------------  P60
.....................................................................
------------------------------  P61
.....................................................................
------------------------------  P62
.....................................................................
------------------------------  P63
.....................................................................
------------------------------  P64
.....................................................................
```

Examples

```
--------------------------------    P65
.....................................................................
--------------------------------    P66
.....................................................................
--------------------------------    P67
.....................................................................
--------------------------------    P68
.....................................................................
--------------------------------    P69
.....................................................................
--------------------------------    P70
.....................................................................
--------------------------------    P71
.....................................................................
LHLL      /B ENABLE
```

TABLE 9–2 Fuse Table

TEST CASE: TTS10-12

P-Term #	Value	P-Term #	Value
P0	/(a*b*c*e)	P17	/(b*c*d*e)
P1	/(g*d)	P18	/(g*a*b*d)
P2	F+/P3+/P4	P19	/(P20*P21*P22*P23*P24*P25)
P3	/(a*c)	P20	m+/k+/a
P4	/(g*b*d)	P21	m+/k+/j
P5	/(/b*P6*P7)	P22	m+/a+f
P6	/(a*b*c)	P23	m+/j+f
P7	/(e*P8)	P24	m+/k+b
P8	c+/P18	P25	m+b+f
P9	a+/P6	P26	/(P27*P28)
P10	b+d	P27	/P+a+f
P11	g+b	P28	/(m*i)
P12	/(P13*P14)	P29	/p+m+/P30
P13	c+/f	P30	/a+/b+e
P14	g+/b	P31	/(/p*k*P32)
P15	a+/P16+/P17	P32	j+/i
P16	/P5		

TABLE 9-3 P-Term Values

TTS1-12
CONTROL AND OUTPUT VARIABLE VALUES

```
OEO = P26 = /(P27*P28) = /P27 + /P28 ;

┌──────────────────────────┐
│ OEO = p*/a*/f + m*i ;    │
└──────────────────────────┘

OE2 = P29 = /P + M + /P30 ;

┌──────────────────────────┐
│ OE2 = /p + m + a*b*/e ;  │
└──────────────────────────┘

DB7 = P31 = p + /k + /P32 = p + /k + /j*i ;

┌──────────────────────────┐
│ DB7 = p + /k + /j*i ;    │
└──────────────────────────┘
```

OUTPUT EQUATIONS:

```
V=/(P0*P1)=a*b*c*e+g*d;
 Note that: CPT1 = a*b*c ;
┌──────────────────────────┐
│ V = e*/CPT1 + /g*/d ;    │
└──────────────────────────┘

W = P2 = f + /P3 + /P4

 Note that: b*d*g = CPT2;
┌──────────────────────────┐
│ W = f + a*c + CPT2 ;     │
└──────────────────────────┘

X = P15

┌──────────────────────────────────────────┐
│ X = a + [b + CPT1 + e*(c+a.CPT2)] + b*c*d*e ; │
└──────────────────────────────────────────┘
```

```
Y = /P9
```

```
Y = /(a + CPT1) ;
```

```
S = P12 = /P13 + /P14
```

```
S = f*/c + b*/g ;
```

```
T = P19 = /(P20*P21*P22*P23*P24*P25)
```

```
T = m*k*a + m*k*j + m*a*f + m*j*f + m*k*b + m*b*f ;
```

9.5 PLE IN AMAZE

This release of AMAZE has the capability to design with Signetics PROM's to generate logic functions.

The following key elements of AMAZE help implement logic functions into a PROM device :

1) BLAST
2) SIM
3) DPI

Examples

PROM's of course, can be viewed as big Karnaugh maps. Therefore, there is no way to build an equivalent Program Table similar to that of PLD's. (Note that PTE cannot work with ".STD" files generated for PROMS).

We have chosen the HEX data format to show the multiple outputs of the various Signetics PROMS. One or two hex characters are used to show the four or eight outputs respectively of the various PROM devices. This output data is stored in a big array map which is called a Standard File or ".STD" file.

To display the file, arrange the width to sixteen columns by expanding the rightmost column to its 16 HEX values. The typical PROM .STD file will look similar to Figure 9-15. Where Cx1 or Cx2 is a HEX character in HEX column "X" with the 4 lower significant outputs (Cx1) or the 4 most significant address outputs (Cx2).

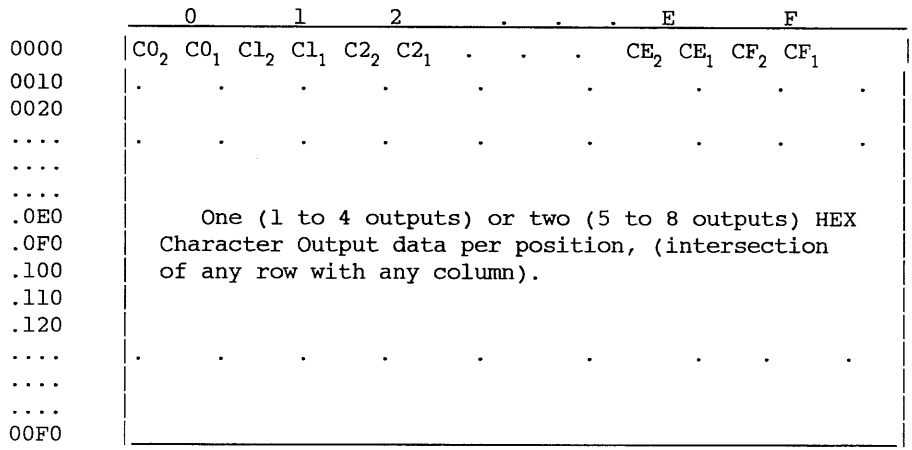


Figure 9-15 PROM Standard File Format

This Standard Table format (Figure 9-15) is useful for verifying design information but it is not recommended that one actually edit this file for modifications. It is recommended that the equations be modified and re-entered into BLAST. This will automatically create a new, updated Standard File (.STD) containing the modified information after assembly with BLAST. Unused inputs will be held at a logic value of zero. Therefore, for each unused variable, half of the device will be used.

Negative asserted devices like the 82HS187 and 82HS189 will have a default value of one and therefore the virgin state.STD HEX Data file for these devices will be all F's, (rather than all 0's for the positive asserted devices). Initiation of PROM's with Registered Outputs (82HS187, 82HS189, etc.), is done in the following way:

In BLAST Menu select "2" (Edit logic, Boolean equations - .BEE File)

```
@ Initial   |                          |
  LABEL:: = | [<Symbol>], <Symbol>]    |";"
            |                          |
```

The above label is a dummy, holding variable.

NOTE: It is recommended to start with the lowest address bit and work towards higher address bits as more input variables are added.

The following example shows a six (6) variable design in three equations, (three outputs) for an 82S129 PROM, (1K bit). The same three equations will be reimplemented in a PLS153 design. The standard file for the PROM design will then be compared to the Program Table (PLD Standard File with labels, heading, etc., put around it for clarity) for the PLS153 device.

Examples

The three equations are:

01 = A*B +C*D + E*F*A + /B*C/D*F ;

03 = E*C*/F ;

04 = F*C ;

The resulting Karnaugh maps for each output are then converted to the PROM Standard file notation. These are then compared individually with the AMAZE Output Standard File.

BLANK TEMPLATE No Output

		E							
	\BA	0				1			
	DC \	0 0	0 1	1 1	1 0	1 0	1 1	0 1	0 1
	00	0	1	3	2	2	3	1	0
0	01	4	5	7	6	6	7	5	4
	11	C	D	F	E	E	F	D	C
F	10	8	9	B	A	A	B	9	8
	10	8	9	B	A	A	B	9	8
	11	C	D	F	E	E	F	D	C
1	01	4	5	7	6	6	7	5	4
	00	0	1	3	2	2	3	1	0

NOTE: SPACES INDICATE THE INACTIVE STATE.

Figure 9–16 Variable Array Template

Finally, all three outputs are strung together as they would appear normally in a design and the resulting AMAZE Standard File is shown with HEX characters (Figure 9–17).

O_1 = A*B + /C*D + E*/F*A + /B*C*/D*F;

E

	\BA \	0				1			
	DC \	0 0	0 1	1 1	1 0	1 0	1 1	0 1	0 1
	00	0	1	3 1	2	2	3 1	1 1	0
0	01	4	5	7 1	6	6	7 1	5 1	4
	11	C	D	F 1	E	E	F 1	D 1	C
F	10	8 1	9 1	B 1	A 1	A 1	B 1	9 1	8 1
	10	8 1	9 1	B 1	A 1	A 1	B 1	9 1	8 1
1	11	C	D	F 1	E	E	F 1	D	C
	01	4 1	5 1	7 1	6	6	7 1	5 1	4 1
	00	0	1	3 1	2	2	3 1	1	0

NOTE: SPACES INDICATE THE INACTIVE STATE.

Figure 9–17 Example Variable Array for O1

This can then be readily transformed to the Standard File (.STD), format by converting each quadrant of the above map into one line of the .STD file. See hand calculation below (Figure 9–18):

PLE .STD file format:

For: O_1 = A*B + /C*D + E*/F*A + /B*C*/D*F;
ADDRESS = "FE dcba

"FE"	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0				1			1	1	1	1	1					1
01	1		1		1		1		1	1	1	1	1		1		1
10	2				1	1	1		1	1	1	1	1				1
11	3				1	1	1		1	1	1	1	1				1

Figure 9–18 Example .STD File for O1

An example was built around this first equation only. See Figure 9–19 and Figure 9–20 below for the pertinent input data on test case XMAS–5

```
File Name: XMAS-5

   ************************PINLIST*********************

   LABEL    **  FNC  **PIN            PIN** FNC **    LABEL
 N/C         **  I    **  1-|        |-16 ** +5V **VCC
 F           **  I    **  2-|   8    |-15 ** I   **N/C
 E           **  I    **  3-|   2    |-14 ** /CE **N/C
 D           **  I    **  4-|   S    |-13 ** /CE **N/C
 A           **  I    **  5-|   1    |-12 ** 0   **01
 B           **  I    **  6-|   2    |-11 ** 0   **N/C
 C           **  I    **  7-|   9    |-10 ** 0   **N/C
 GND         **  OV   **  8-|________|- 9 ** 0   **N/C
```

Figure 9–19 PINLIST (XMAS–5) for O1

```
@DEVICE TYPE
82S129
@DRAWING
@REVISION
@DATE
2/28/87
@SYMBOL
@COMPANY
SIGNETICS
@NAME
JOE USER
@DESCRIPTION
MAPPING VERIFICATION
@COMMON PRODUCT TERM
@LOGIC EQUATION
O1 =A*B /C*D + E*/F*A + /B*C*/D*F ;
```

Figure 9–20 (XMAS–5) O1 BLAST Transaction

Examples

The AMAZE processed .STD file below (Figure 9-21) compares favorably to that calculated above in Figure 1-7.

```
82S129
          0 1 2 3 4 5 6 7 8 9 A B C D E F
-------------------------------------------

A0000     0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 1
A0010     0 1 0 1 0 1 0 1 1 1 1 1 0 1 0 1
A0020     0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 1
A0030     0 0 0 1 1 1 0 1 1 1 1 1 0 0 0 1
```

Figure 9-21 AMAZE Processed .STD FILE FOR O1

```
03 = C* E*F ;
```

\BA \ DC \	E = 0				E = 1			
	0 0	0 1	1 1	1 0	1 0	1 1	0 1	0 1
00	0	1	3	2	2	3	1	0
01	4	5	7	6	6	7	5	4
11	C	D	F	E	E	F	D	C
10	8	9	B	A	A	B	9	8
10	8	9	B	A	A	B	9	8
11	C	D	F	E	E 1	F 1	D 1	C 1
01	4	5	7	6	6 1	7 1	5 1	4 1
00	0	1	3	2	2	3	1	0

0 F 1

Note: Space indicates the inactive state.

Figure 9-22 Example Variable Array for O3

.STD file format:

For: O_3 = C* E*F ;
ADDRESS = "FE dcba

"FE"	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0																
01	1																
10	2																
11	3					4	4	4	4					4	4	4	4

Figure 9-23 Example .STD File for O3

```
     File Name: XMAS-7

**************************PINLIST********************

   LABEL    **  FNC   **PIN  _________   PIN** FNC **    LABEL
N/C         **  I     **  1-|         |-16 ** +5V **VCC
F           **  I     **  2-|    8    |-15 ** I   **N/C
E           **  I     **  3-|    2    |-14 ** /CE **N/C
D           **  I     **  4-|    S    |-13 ** /CE **N/C
A           **  I     **  5-|    1    |-12 ** 0   **N/C
B           **  I     **  6-|    2    |-11 ** 0   **N/C
C           **  I     **  7-|    9    |-10 ** 0   **03
GND         **  OV    **  8-|_________|- 9 ** 0   **N/C
```

Figure 9-24 Example PINLIST for O3

```
FILE NAME:  XMAS-7

@DEVICE TYPE
82S129
@DRAWING
@REVISION
@DATE
2/28/87
@SYMBOL
@COMPANY
SIGNETICS
@NAME
JOE USER
@DESCRIPTION
COMMON PRODUCT TERM
@LOGIC EQUATION
O3 = C* E*F ;
```

Figure 9–25 Example O3 BLAST Transaction

```
CUST/PROJECT - JOE USER

82S129
         0 1 2 3 4 5 6 7 8 9 A B C D E F
-------------------------------------------

A0000    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A0010    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A0020    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A0030    0 0 0 0 4 4 4 4 0 0 0 0 4 4 4 4
```

Figure 9–26 AMAZE Processed .STD File for O3

0_4 = F*D ;

		E							
	\BA \	0				1			
	DC \	0 0	0 1	1 1	1 0	1 0	1 1	0 1	0 1
	00	0	1	3	2	2	3	1	0
0	01	4	5	7	6	6	7	5	4
	11	C	D	F	E	E	F	D	C
F	10	8	9	B	A	A	B	9	8
	10	8 1	9 1	B 1	A 1	A 1	B 1	9 1	8 1
1	11	C 1	D 1	F 1	E 1	E 1	F 1	D 1	C 1
	01	4	5	7	6	6	7	5	4
	00	0	1	3	2	2	3	1	0

Figure 9–27 Example Variable Array for O4

.STD file format:

```
For: 04 = F*D ;
ADDRESS = "FE dcba
```

"FE"	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0																
01	1																
10	2									8	8	8	8	8	8	8	8
11	3									8	8	8	8	8	8	8	8

Figure 9–28 Example .STD File for O4

```
CUST/PROJECT - JOE USER
82S129
         0 1 2 3 4 5 6 7 8 9 A B C D E F
_____________________________________________

A0000    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A0010    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A0020    0 0 0 0 0 0 0 0 8 8 8 8 8 8 8 8
A0030    0 0 0 0 0 0 0 0 8 8 8 8 8 8 8 8
```

Figure 9-29 AMAZE Processed .STD File for O4

```
File Name: XMAS-6

**************************PINLIST************************
    LABEL    **  FNC   **PIN  _________   PIN** FNC **    LABEL
  N/C        **  I     **  1-|         |-16 ** +5V **VCC
  F          **  I     **  2-|    8    |-15 ** I   **N/C
  E          **  I     **  3-|    2    |-14 ** /CE **N/C
  D          **  I     **  4-|    S    |-13 ** /CE **N/C
  A          **  I     **  5-|    1    |-12 ** 0   **N/C
  B          **  I     **  6-|    2    |-11 ** 0   **N/C
  C          **  I     **  7-|    9    |-10 ** 0   **N/C
  GND        **  OV    **  8-|_________|- 9 ** 0   **04
```

Figure 9-30 Example PINLIST for O4

```
FILE NAME:  XMAS-6

@DEVICE TYPE
82S129
@DRAWING
@REVISION
@DATE
2/28/87
@SYMBOL
@COMPANY
SIGNETICS
@NAME
JOE USER
@DESCRIPTION
COMMON PRODUCT TERM
@LOGIC EQUATION
O4 = F*D ;
```

Figure 9-31 O4 BLAST Transaction

Only four lines of the .STD file (from a total of sixteen possible) are shown because only using six variables (out of eight possible) the output maps into one-quarter of the device.

The last few pages have shown how each of the three equations look individually for the Standard File. Below, the three output functions are combined into one test case, XMAS-8. The following page shows the AMAZE derived outputs.

```
File Name: XMAS-8

*************************PINLIST************************

   LABEL    **  FNC   **PIN  _________   PIN** FNC **    LABEL
N/C         **  I     **  1-|         |-16 ** +5V **VCC
F           **  I     **  2-|    8    |-15 ** I   **N/C
E           **  I     **  3-|    2    |-14 ** /CE **N/C
D           **  I     **  4-|    S    |-13 ** /CE **N/C
A           **  I     **  5-|    1    |-12 ** 0   **01
B           **  I     **  6-|    2    |-11 ** 0   **N/C
C           **  I     **  7-|    9    |-10 ** 0   **03
GND         **  OV    **  8-|_________|- 9 ** 0   **04
```

Figure 9-32 Example PINLIST for O1, O3, O4

```
FILE NAME:  XMAS-8

@DEVICE TYPE
82S129
@DRAWING
@REVISION
JOE USER
@DATE
2/28/87
@SYMBOL
@COMPANY
SIGNETICS
@NAME
LES NONE
@DESCRIPTION
COMMON PRODUCT TERM
@LOGIC EQUATION
O1 = A*B + /C*D + E*/F*A + /B*C*/D*F ;

O3 = E*F*C ;

O4 = F*D ;
```

Figure 9–33 Composite BLAST Transaction (82S129)

0_4 = XMAS-8

		E = 0				E = 1			
	\BA / DC \	0 0	0 1	1 1	1 0	1 0	1 1	0 1	0 1
	00	0	1	3 1	2	2	3 1	1 1	0
0	01	4	5	7 1	6	6	7 1	5 1	4
	11	C	D	F 1	E	E	F 1	D 1	C
F	10	8 1	9 1	B 1	A 1	A 1	B 1	9 1	8 1
	10	8 9	9 9	B 9	A 9	A 9	B 9	9 9	8 9
1	11	C 8	D 8	F 9	E 8	E C	F D	D C	C C
	01	4 1	5 1	7 1	6	6 4	7 5	5 5	4 5
	00	0	1	3 1	2	2	3 1	1	0

Figure 9–34 Example Composite Variable Array

.STD file format:

For: XMAS-8
ADDRESS = "FE dcba

"FE"	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0				1				1	1	1	1	1				1
01	1		1		1		1		1	1	1	1	1		1		1
10	2				1	1	1		1	9	9	9	9	8	8	8	9
11	3				1	5	5	4	5	9	9	9	9	C	C	C	D

Figure 9–35 Example Composite .STD File

```
CUST/PROJECT:  JOE USER

FILE NAME:  XMAS-8
82S129
         0 1 2 3 4 5 6 7 8 9 A B C D E F

A0000    0 0 0 1 0 0 0 1 1 1 1 1 0 0 0 1
A0010    0 1 0 1 0 1 1 1 1 1 1 1 0 1 0 1
A0020    0 0 0 1 1 1 0 1 9 9 9 9 8 8 8 9
A0030    0 0 0 1 5 5 4 5 9 9 9 9 C C C D
```

Figure 9-36 Example Composite Processed .STD File

Now to implement these same three equations with a PLD; the PLS153:

```
File Name: XMAS-9

**************************PINLIST************************

   LABEL    **  FNC   **PIN          PIN**  FNC  **    LABEL
A           **  I     **  1-|        |-20 **  +5V  **VCC
B           **  I     **  2-|   P    |-19 **  B    **N/C
C           **  I     **  3-|   L    |-18 **  B    **N/C
D           **  I     **  4-|   S    |-17 **  B    **N/C
E           **  I     **  5-|   1    |-16 **  B    **N/C
F           **  I     **  6-|   5    |-15 **  B    **N/C
N/C         **  I     **  7-|   3    |-14 **  B    **N/C
N/C         **  I     **  8-|        |-13 **  O    **04
01          **  O     **  9-|        |-12 **  O    **03
GND         **  OV    ** 10-|________|-11 **  O    **02
```

Figure 9-37 Composite PINLIST - PLS153

```
FILE NAME:  XMAS-9

@DEVICE TYPE
PLS153
@DRAWING
@REVISION
@DATE
2/28/87
@SYMBOL
@COMPANY
SIGNETICS
@NAME
JOE USER
@DESCRIPTION
COMMON PRODUCT TERM
@I/O DIRECTION
O1 = A*B + /C*D + E*/F*A + /B*/C*/D*F ;

O3 = E*C*/F ;

O4 = F*C ;
```

Figure 9–38 Composite BLAST Transaction for PLS153

Examples

```
  File Name:  XMAS-9
CUST/PROJECT:  - Joe User
PLS153                                    |     POLARITY      |
___
 T |                                      |H:H:H:H:H:H:H:H:H|
 E |-------------------------------------------------------------
 R |       I        |        B(i)        |        B(o)
 M |
___|7_6_5_4_3_2_1_0|9_8_7_6_5_4_3_2_1_0|9_8_7_6_5_4_3_2_1_0|
  0|- - - -.- - H H|- -,- - - -,- - - -|A A,A A A A,. . . A|
  1|- - - -,H L - -|- -,- - - -,- - - -|A A,A A A A,. . . A|
  2|- - H H,- - - H|- -,- - - -,- - - -|A A,A A A A,. . . A|
  3|- - H -,L H L -|- -,- - - -,- - - -|A A,A A A A,. . . A|
  4|- - H H,- H - -|- -,- - - -,- - - -|A A,A A A A,. A . .|
  5|- - H -,- H - -|- -,- - - -,- - - -|A A,A A A A,A . . .|
  6|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
  7|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
  8|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
  9|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 10|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 11|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 12|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 13|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 14|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 15|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 16|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 17|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 18|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 19|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 20|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 21|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 22|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 23|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 24|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 25|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 26|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 27|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 28|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 29|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 30|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 31|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|A A,A A A A,A A A A|
 D9|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|                   .
 D8|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|                   .
 D7|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|                   .
 D6|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|                   .
 D5|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|                   .
```

```
D4|0 0 0 0,0 0 0 0|0 0,0 0 0 0,0 0 0 0|                    .
D3|- - - -,- - - -|- -,- - - -,- - - -|                    .
D2|- - - -,- - - -|- -,- - - -,- - - -|                    .
D1|- - - -,- - - -|- -,- - - -,- - - -|                    .
D0|- - - -,- - - -|- -,- - - -,- - - -|                    .

   N N F E D C B A N N N N N N 0 0 0 0 N N N N N N 0 0 0 0
   / /           / / / / / / 4 3 2 1 / / / / / / 4 3 2 1
   C C           C C C C C C       C C C C C C
```

Figure 9–39 XMAS–9

Output "2" (0_2 or pin label B1) is unused. Hence, the first six terms of that output column are dotted out.

Figure 9–39 for PLD's and Figure 9–36 for PLE's (PROM'S) define the same three equations with the same input variables and the appropriate fusemaps for each.

APPENDIX B

Messages

Appendix B BLAST (Error Messages)

Note: "ERROR" messages are usually fatal and will stop completion of execution cycle.

"WARNING" messages are nonfatal and usually do not affect execution but only alert the user to potentially overlooked situations.

1. ERROR: (On Line)
   Syntax error. Incorrect Format.

2. ERROR: (On Line)
   Syntax error. Only a single product term is permitted here.

3. ERROR: (On Line)
   Unknown label: Labels must not be more than 12 characters in length.

4. ERROR: (On Line)
   Syntax error. Only <1> or <0> is allowed on the right-hand-side of equation.

5. ERROR: (On Line)
   Unknown I/O Direction label or the corresponding output pin label is not defined.

6. ERROR: (On Line)
   Unknown pin label.

7. ERROR: (On Line)
   Syntax error. Not a valid equation.

8. ERROR: (On Line)
   Unknown Output Polarity label.

9. ERROR: (On Line)
   Label must be <FC> : (Flip-Flop Control).

10. ERROR: (On Line)
    Label must be </C> for Complement Array.

11. ERROR: (On Line)
    Label must be <LA> or <LB> :
    (Parallel Load - banks A/B).

12. ERROR: (On Line)
Label must be <PA>, <PB>, <RA> or <RB>
(Preset/Reset - banks A/B).

13. ERROR: (On Line)
Label must match the pin whose function is /OE.

14. ERROR: (On Line)
Syntax error. Comma <,> is not permitted here.

15. ERROR: (On Line)
Duplicated Output Polarity label.

16. ERROR: (On Line)
Duplicated Flip-Flop Mode label.

17. ERROR: (On Line)
Unknown or duplicated Output Enable label.

18. ERROR: (On Line)
Syntax error. Only <1> or <FC> allowed on RHS of equation.

19. ERROR: (On Line)
Unknown Flip-Flop Mode label.

20. ERROR: (On Line)
The device you specified is not supported.

21. ERROR: (On Line)
You have entered an illegal character in this field.

22. ERROR: (On Line)
This label has been used. Please use a new label.

23. ERROR: (On Line)
Syntax error, missing <"> to end a comment.

24. ERROR: (On Line)
You cannot use Internal labels or blanks as a label.

25. ERROR: (On Line)
Product term overfilled. Please review your design.

26. ERROR: (On Line)
Definition for registered output does not match its type. You must use J, K, D, T or R, S labels which match the flip-flop types used. Check for conflicting FLIP-FLOP MODE or FLIP-FLOP CONTROL information if applicable.

27. ERROR: (On Line)
Output pin label cannot be used on the right-hand-side of an equation.

28. ERROR: (On Line)
Duplicated I/O Steering label.

29. ERROR: (On Line)
Input pin label cannot be used on left-hand-side of an equation.

30. ERROR: (On Line)
The Complement Array has not been defined.

31. ERROR: (On Line)
Syntax error. Missing <;> at the end of the equation.

33. ERROR: (On Line)
The function you specify for this pin is not permitted. Refer to the HELP documentation in Appendix B of the manual for the permissible functions.

34. ERROR: (On Line)
Syntax error,the following format is expected:
label = / (expression) ; (Check output polarity definition).

35. ERROR: Syntax error. You must use the identical output label in the Pinlist and the left-hand-side of an equation; only when using the PLS155-9, or 179 register outputs, or "/D" function in pinlist you must use the negation of the label. of the label..

36. ERROR: (On Line)
Output Enable, Preset, Chip Enable and Clock signals cannot be used in any Boolean equations.

37. ERROR: (On Line)
There should not be any common terms between the "K" or "T" inputs and the "D" input of a flip-flop in the Logic Equations.

38. ERROR: (On Line)
Complement Array has been defined in terms of itself.

39. ERROR: (On Line)
You cannot use <0> or <1> as a pin label.

40. ERROR: (On Line)
You can only use <1>, <D0>, <D1>, or <D2> on the right-hand-side of this equation.

41. ERROR: (On Line)
Unknown I/O Steering label.

42. ERROR: (On Line)
Incorrect I/O Direction label has been defined for an I/O Steering label on the left-hand-side of the equation.

43. ERROR: (On Line)
You cannot have blanks or slashes between characters of a pin label.

44. ERROR: (On Line)
This Internal S/R flip-flop has already been defined.

45. ERROR: (On Line)
Nondeterministic use of S/R flip-flop.

46. ERROR: (On Line)
You cannot have blanks or slashes between characters of an internal S/R flip-flop label.

47. ERROR: (On Line)
FC definition conflicts with assumed F/F mode as described by logic equations.

48. ERROR: (On Line)
The pin function is defaulted to input. You must define CK2 in the pinlist.

49. ERROR: (On Line)
This pin is dedicated as a clock pin. It cannot be used on the right-hand-side of equations.

50. ERROR (On Line)
Label must be (C0) or (/C1) for multiple complement array.

51. ERROR (On Line)
Label must be (/C) for single complement array.

52. ERROR (On Line)
Clock and output enable pins must be defined because one or more pin(s) are chosen to be D register output(s).

101: ERROR: (On Line)
Duplicate clock label.

102. ERROR: (On Line)
Multiple definition of an equation in @Logic Equation field.

103. ERROR: (On Line)
Multiple definition of an equation @Common Product Term field.

104. ERROR: (On Line)
Multiple definition of an equation in @Complement Array field.

105. ERROR: (On Line)
Multiple definition of an equation in @Flip-Flop Control field.

106. ERROR: (On Line)
Multiple definition of an equation in @Asynchronous Preset/Reset field.

107. ERROR: (On Line)
Multiple definition of an equation in @Register Load field.

108. ERROR: (On Line)
Multiple definition of an equation in @I/O Direction field.

109. ERROR: (On Line)
I/O Direction information given here conflicts with that implied from pinlist screen.

110. ERROR: (On Line)
I/O Steering information given here conflicts with that implied from pinlist screen.

111. ERROR (On Line)
Internal labels are declared but not defined.

120. ERROR: (On Line)
Output Polarity information given here conflicts with that implied from pinlist screen.

126. ERROR: (On Line)
The first character of the vector label must be alpha.

127. ERROR: (On Line)
You have designated this bank of flip-flops as bidirectional pins, this Output Enable definition must be the label of the output enable pin, not <0> or <1>.

128: ERROR: (On Line)
You have designated this bank of flip-flops as input pins, therefore, this Output Enable definition must be <1>.

129. ERROR: (On Line)
You have designated this bank of flip-flops as bidirectional pins, therefore, this Register Load definition must be an expression other than <0> or <1>.

130. ERROR: (On Line)
You have designated this bank of flip-flops as input

pins, therefore, this Register Load definition must be <1>.

131. ERROR: (On Line)
You have designated this bank of flip-flops as output pins, therefore the output enable definition must be <0>.

132. ERROR: (On Line)
You cannot mix output and bi-directional functions for pins under the same control term of a PLS151 device.

133. ERROR: (On Line)
You cannot use the negation of the COMPLEMENT ARRAY.

134. ERROR: (On Line)
You must define an I/O Direction equation for pins whose function is Bi-directional <B> or Bi-directional Complement </B>.

135. ERROR: You must have more COMMON PRODUCT TERMs than the number of P-terms available in the device.

136. ERROR: Unknown Header in the .SEE data file.

137. ERROR: <Filename> is expected. This file name is the name of the .BEE design under the @Device Selection of the .SEE data file - Format: <filename>/PLXXXXX (Must be different than .SEE filename).

138. ERROR: Device name is expected under the @Device Selection heading of the .SEE data file - Format: <.BEE filename>/PLXXXXX.

139. ERROR: Unexpected label. Pin Label or Vector Label is expected.

140. ERROR: <[> - (square bracket) is expected.

141. ERROR: This label is used in the grouping, so it can not be used again.

142. ERROR: This label is defined as a Vector. Duplicate definition detected.

143. ERROR: This label is not defined as a PinLabel.

144. ERROR: This PinLabel is invalid for usage in this context.

145. ERROR: PinLabel is expected in the grouping.

146. ERROR: <,> or <]> is expected or missing.

147. ERROR: Vector Label is expected or missing.

148. ERROR: <=> is expected or missing.

149. ERROR: Error in number definition.

150. ERROR: Unexpected Label in number.

151. ERROR: Number is bigger than 100 digit number (Hex, Octal or Binary).

152. ERROR: Number used is greater than that can be fit with the defined grouping, or ERROR in number.
(OR)
Number of binary bits used in binary number should be equal to the number of labels used in grouping.

153. ERROR: Unknown PinLabel or number.

154. ERROR: A separator <,> or <*> is expected in the Vector Definition.
<*> is the separator in State or Input Vectors.
<,> is the separator in Output Vectors.

155. ERROR: PinLabel is expected or missing.

156. ERROR: One of the PinLabels defined in the current grouping or vector definition is invalid for synchronous output pin label; (Note: Clocked output vectors/ labels need apostrophe: REG_OUT').

157. ERROR: This label can not be used in this vector because this is not defined as a PinLabel, or functionally it cannot be used in the vector definition.

158. ERROR: Unexpected word or character.

159. ERROR: </> - (slash) is expected or missing.

160. ERROR: DEVICE_Name is expected or missing.

161. ERROR: The Complement Array is not available in this device.

162. ERROR: <IF> or <CASE> statement is expected or missing.

163. ERROR: <THEN> statement is expected or missing.

164. ERROR: <::> is expected or missing.

165. ERROR: <ENDCASE> statement is expected or missing.

166. ERROR: Repeated use of <ELSE> clause in <WHILE> statement.

167. ERROR: Input pin is not in the same device.

168. ERROR: <'> is not allowed.

169. ERROR: The exact label should be used here as declared in Vector definition.

170. ERROR: <'> is required.

171. ERROR: <+> is not allowed in this expression.

172. ERROR: Complement is not allowed in this expression.

173. ERROR: Illegal syntax: Compiler cannot complete reduction.

174. ERROR: </C> is only allowed in input condition fields (i.e. IF argument), and Input Vector definition fields.

175. ERROR: PinLabel definition by number is not valid here. This number does not have a corresponding valid grouping.

176. ERROR: In vector grouping no more than 100 labels are allowed.

177. ERROR: <Label> and </Label> cannot be used together in Pinlists.

178. ERROR: Complementing a Vector label is allowed only in <IF> expression.

179. ERROR: The <ELSE> statement cannot be used for this transition.

180. ERROR: Output only pin is not allowed in state vector.

181. ERROR: Conflicting output or state assignment detected.

182. ERROR: Incorrect usage of apostrophe <'>.

183. ERROR: The equation specified exceeds the maximum number of P-terms for the output pin. Please recheck the equation or consider FPLA.

184. ERROR: Complement Array output = 0 unconditionally. (Both IF and WHILE variables are NIL.)

185. ERROR You have exceeded the maximum limit of 72 Common Product Terms. Please reduce the number of common product terms.

201. WARNING: A product is duplicated in this sum of products equation.

203. WARNING: Since you did not explicitly define I/O DIRECTION in the .BEE file, asynchronous I/O pin direction defaults are controlled by the functions assigned in the Pinlist screen.

204. WARNING: Information provided has used up the maximum available storage space. No additional entry is accepted. Please delete any unnecessary information.

221. WARNING: Incomplete Asynchronous Preset/Reset definition. Defaults assumed: Undefined Presets/Resets are disabled. (P, R-term fuses intact.)

222. WARNING: Incomplete Flip-Flop Mode definition. Defaults assumed: Undefined F/F mode is controlled by FC-term. (Mode fuse intact.)

223. WARNING: Incomplete I/O Direction definition; Defaults assumed: Undefined I/O Direction control <D> terms default to the functions (input or output) assigned^ in the Pinlist screen. (^B-pin assignments require definition of respective D-terms.)

224. WARNING: Incomplete Register Load definition. Defaults assumed: Register Load is disabled until defined. (L-term fuses intact.)

225. WARNING: Incomplete Output Polarity definition. Defaults assumed: Undefined Output Polarity defaults to functions assigned in the Pinlist screen.

231. WARNING: You have designating this bank of flip-flops as output pins, the Output Enable and Register Load cannot be used to externally load the registers.

232. WARNING: (On Line)
You are defining an Asynchronous <B> output in terms of itself. This feedback may result in a latching function or oscillator at the output.

236. WARNING: You should designate the SAME FUNCTION for all pins of the same bank of flip-flops. If you use the flip-flops of a given bank differently, you should be aware that they all behave in the same manner at any given instance of time.

237. WARNING: Unknown section title ignored.

240. WARNING: Conflicting Vector definition and usage; (synchronous/Asynchronous).

241. WARNING: <'> has no meaning in any section of declaration other than Output Vectors and AND clause of TRANSITIONS. So <'> is ignored elsewhere.

242. WARNING: <'> is associated with a bi-directional pin: (<"> must be associated with a clocked output or Vector).

300. WARNING: You used this label in the equation but this pin or function has not been defined.

320. WARNING: You are defining an output in terms of negation of itself.

400. ERROR: Internal error module <, routine <routine).

401. ERROR: Run aborted, module <module>.

402. ERROR: Missing or corrupted <filename>.

403. ERROR: Out of memory module <N module>.

404. ERROR: Empty partition list. No equations written.

406. ERROR: <Filename> too long. Name should not exceed 8 characters.

410: ERROR: Missing "data" selection of <filename>.

411. ERROR: Conflicting default polarities found in AMZ file. name>".

412. ERROR: Expecting "<ch>" in AMAZE.CTL definition of type "<type name>".

413. ERROR: "<pin function>" is an invalid pin function for type "<type name>" in AMAZE.CTL file.

414. ERROR: Logic equation of the type "<type name>" in AMAZE.CTL file has more than 255 characters. Please verify that the formula is syntactically correct. If so, contact Signetics to increase "MAX_PIN_FORMULA_LEN".

415. ERROR: <Signal name> referred to in pin file <filename> does not appear in circuit. It will be ignored.

416. ERROR: Reserved label <label> should not be used as a signal name.

417. ERROR: Format error in netlist routine <routine>.

418. ERROR: No pin attributes given for AMAZE.CTL.

420. ERROR: Unknown pin function found in description of type "<typename>". Check library file <library filename>.

421. ERROR: Unknown format found in description of type "<typename>". Check library file <library filename>.

422. ERROR: Expecting 'string' in description of type. Check library file <library filename>.

424. ERROR: Duplicate pin function found in description of type. Check library file <library>.

425. ERROR: Formula of type 'string' contains more than the allowed characters. Check library file

<library filename>.

426. ERROR: Expecting "=". Statement should read:
CP MFANIN = <integer>; check library file
<library filename>.

427. ERROR: Unrecognized section header in library file
<library filename>. Skipping to next section.

428. ERROR: No <Register or Tri-State> types declared in library
file <library filename>.

430. ERROR: No pin attributes given in library file
<library filename>.

431. ERROR: Type <typename> is not described in library file
<library filename>.

432. ERROR: Incomplete netlist created. Contact Signetics.

434. ERROR: Illegal use of pin <pin name>, signal "<signal name>".

435. ERROR: Internal feedback detected, BEE file will contain a
"?" where this signal should be.

436. ERROR: Unknown pin/signal name, BEE file will contain a "?"
where this signal should be. Verify that pinlist is
complete.

437. ERROR: Implicit wired functions are not allowed. Check
signal <signal name>.

438. ERROR: Unable to find pin <pin name> of element <element
name>.

439. ERROR: Pin <pin name> of type "<type name>" is not defined in
file <filename>. Check pin attributes.

440. ERROR: Net conflict on PLD point <pld number>.

441. ERROR: Polarity conflict on PLD point (pld number>.

442. ERROR: <Register name> cannot be used as an input transfer,
no load signal.

443. ERROR: Signal <signal name> should appear on a register input pin.

444. ERROR: Register <register name> is associated with more than one PLD pin.

445. ERROR: More than one register associated with PLD pin <pin number>.

446. ERROR: Unable to fit register <register name> into PLD device. No registers (left) on device.

447. ERROR: Register <register name> should contain a FF control line.

448. ERROR: Unrecognized register mode mix on register.

449. ERROR: Pin <pin name> does not have polarity control capability.

450. ERROR: I/O pin found without control line.

451. ERROR: Misplaced register type on register. Expecting J/K, S/R, D, T type register.

452. ERROR: Unable to find any input data line (J, K, D, T, R, S) of flip-flop instance "<element name>".

453. ERROR: Instance "<element name>" is configured to be an internal register in the schematic, but the designated PLD does not provide such a capability; no output pin label is assigned to the register.

454. ERROR: Unable to find data line of flip-flop element "<element name>". Contact Signetics.

455. ERROR: Bi-directional pin "string" requires load line to be set.

456. INFO: Assigning signal <signal name> to a complement line.

457. INFO: Complement line <complement label> was not used in circuit <circuit name>.

458. ERROR: Unable to find clock line of register <register name>.

460. ERROR: Flip-flop type on pin <pin number> does not match the targetted PLD device.

461. ERROR: "<element name>" cannot be associated with more than one register pin.

462. ERROR: Pin <pin number> must have an SR flip-flop attached to it.

463. ERROR: Unknown pin category for pin <pin number>. Check .AMZ file.

464. ERROR: Invalid flip-flop mode combination (mixed SR and JK/D/T inputs on the same flip-flop) on instance "<element name>".

465. ERROR: More than one flip-flop control (FC) line found in schematic.

466. ERROR: Flip-flop "<element name>" has programmable (mixed mode) data lines (ie. JK-D) but no FC (control) line.

467. ERROR: Unknown register type used for element "<element name>". Valid types are JK, D, T, JK-D, JK-T, JK-D-T, and SR.

469. ERROR: Flip-flop control (FC) line is required when more than one flip-flop type or mixed mode flip-flops are used.

470. WARNING: Register <register name> contains no data pins.

471. ERROR: Output label of register <register name> defaulted to <label name>. User must assign a name to this signal.

472. ERROR: Function conflict on PLD <pin name>, function <function>. Pin should not be used as <function>.

473. ERROR: Missing input pin for tri-stateable instance "<element name>".

474. ERROR: Pin <pin number> of the designated PLD is not tri-stateable.

481. ERROR: Corrupted data. Call Signetics.

482. ERROR: Signal name too long. Shorten signal "<signal name>".

484. ERROR: No <input line> input to flip-flop "<element name>". Corrupted data. Call Signetics.

516. WARNING: Unrecognized section header "<string>" in AMAZE.CTL file. Skipping to next section.

531. WARNING: String read has been truncated to 12 characters, to read "<string>".

542. WARNING: Internal feedback detected (signal "<signal name>"). BEE file will contain a "?" where this signal should be.

543. WARNING: Unknown pin/signal name "<signal name>". .BEE file will contain a "?" where this signal should be. Verify that pinlist is complete.

559. WARNING: Load line supplied for flip-flop running to pin whose function is "O" or "/O". Load of bank <register bank> initialized to "<signal name>".

583. WARNING: Pin <pin number> is used in place of "/OE".

591. INFORMATION: No <register or tri-state> types declared in AMAZE.CTL.

592. INFORMATION: Signal "<signal name>" referred to in the pinlist cannot be found in the schematic.

593. INFORMATION: <preset or reset> of bank <register bank> will default.

594. INFORMATION: "<signal name>" is used as flip-flop control (FC) signal.

AMAZE GLOSSARY

AMAZE . . . . . . The user software environment developed by Signetics to support PLD design. It consists of several interconnected software modules. The name is an acronym for Automatic Map And Zap Equations.

AMAZE MASTER . . . AMAZE MASTER is a term no longer used. Prior to version 1.65, it referred to the mandatory disk placed in drive A:; used to hold the design files. Design files are now held in a directory specified upon starting AMAZE. See Section 1.1.

BLAST . . . . . . The primary front end software module of AMAZE. Within BLAST, the user may define his design pins, establish logical relationships among internal circuit nodes and generate the basic design definition files. The name is an acronym for Boolean Logic And State Transfer.

Bracket Notation . . . . Unique to PML designs, enclosing a portion of an equation with brackets prevents those literals enclosed from combining with terms outside of the brackets.

Buried Register – (also see Register) . Any flip–flop (or group of flip–flops) which does not connect directly to an input or output pin.

Common Product Term A symbol associated with an expression that is to be used later defined logic equations. A form of macro substitution to simplify equation typing tasks where repeated functions are desired. See Section 3.5.3.

Complement Array . . . A single term (inverted OR) function is a signal routing convenience allowing one or more product terms (depending on the specific part) to be inverted and fedback to the logic array without exiting and re–entering via pins.

Controller – (also see State Machine) .
A state machine, similar to a sequencer, designed for rapid transactions with the input and output pins of a device. The goal of a controller is high speed reaction to an input stimulus to generate correct output response and ultimately exhibit "control" of some process. The fastest responding controllers are Mealy machines.

DPI . . . . . . Device Programmer Interface. This AMAZE module allows uploading and downloading of fuse configuration files to an externally connected programmer. The physical connection is by serial port and the software supports several fuse file standards.

Fusemap . . . . . . The array description of the binary contents of a PROM–type configuration image. This may look like a large truth table or a cross–hatched interconnect grid.

Internal Node . . . . . Unique to PML designs, it is a symbol associate with the output of a gate not connected to a device pin but connected to another gate input. Please see Section 3.5.9.

JEDEC FILE . . . . A computer file, developed by the JEDEC JC–42 committee, which describes the information format for reading and writing fuse configurations. It contains the address and contents of the specific fusemap, it also includes specification of the number of fuses, default fuse state, signature analysis, access time, pin sequence, test vectors, etc.

LOG FILE . . . . . The basic documentation of a simulation session. The Log File generation, which is optional, contains both input stimulus and output response information, and has a .LOG extension. It is used for documentation, debugging and automatic test vector generation.

MACRO . . . . . . A term used most often in computer science describing a function. In PLDs, a MACRO is a repeatable function which can usually be altered by the designer. One example of a MACRO would be generating an exclusive–OR function from four NAND gates. The cluster of gates could be labeled as a MACRO and everytime an exclusive–OR function is needed, the four gate cluster could be used.

Mealy . . . . . A category of finite state synchronous sequential machines, named after its inventor. Mealy machines are described elsewhere within the manual, but the key property is that of gating the input to the machine with the present state of the machine to generate the output. Because its output is gated directly, the Mealy machine's output may change in between clocks, resulting in a pulsed output. A Mealy machine typically responds faster and has fewer states than a corresponding Moore machine.

Moore . . . . . A category of finite state synchronous sequential machines, named after its inventor. Moore machines are described elsewhere within the manual, but the key property is that of generating an output as a function of the present state only. Common Moore machines include counters and shift registers. Because the output can only change in response to an input and a clock, Moore machines are typically slower than Mealy machines.

P–term . . . . . Product terms. The result of the Boolean AND function. The number of available P–terms for a PLD is sometimes taken as bounding its functional capacity. A P–term in the virgin is an input AND gate (n = number of AND array inputs). When programmed each is a result of the Boolean AND function. In some devices (PLA's PLS's) common product terms can be shared between several outputs.

PAL . . . . . Programmable Array Logic. A trademarked architecture of MMI. PALs allow flexibility at the input P–term level only. The OE array which drives the outputs is no programmable.

PLA . . . . . Programmable Logic Arrays. The most flexible of the PLD architectures, allowing configuration of the AND array as well as the OR array for complex 2–level combinational designs.

PLD . . . . . The whole class of semicustom products which are use configurable. Includes PLA, PAL, PLE and EPROM architectures. PLD stands for Programmable Logic Devices.

PLE . . . . . Programmable Logic Element. This is a new term for an old idea – PROMS.

PTP . . . . . An AMAZE module which converts PAL based designs to PLD devices in the Signetics product line. The software automatically accesses the fusemap in a PAL and will generate a functionally corresponding fusemap for Signetics' products. PTP is an acronym for PAL to PLD.

Register . . . . . Flip-flops are called registers (1-bit!) in the PLD world. They can be S-R, T, D or J-K varieties.

Registered PROM. A PROM with an output data latch. If the data width is appropriate, a registered PROM can be configured into a very primitive sequence generator (sequencer).

Ruler File . . . . . A display configuration description file used in the AMAZE simulation module, SIM. It allows the user to input and output variables in a prescribed format on the screen. The Ruler File has a file extension of .RUL.

S-term . . . . . (sum term). The logical output of a Boolean OR function is called an S-term which is short for logical sum term. Two level combinational functions are often realized by using both P-terms and S-terms.

Sequencer . . . . . A state machine which is capable of generating an output sequence (sequence generator = sequencer). Sequencers output a specific pattern of logical ones and zeroes for a given input configuration. They usually have a level output and hence are most often Moore machines. Registered PROMs can be simple sequencers. The PLUS405 is a sophisticated sequencer.

SIM . . . . . The AMAZE module which performs the device simulation function. Along with design files, input vector files and a Ruler file, it generates output vectors and a LOG FILE. Other features are described elsewhere in the manual.

State Machine . . The term which has come to describe the whole collection of sequential machines. It encompasses Mealy, Moore, sequencer, controller, finite state machines, synchronous and asynchronous types. In this manual, the most consistent usage is simply machines containing clocked flip-flops as well as other features.

Test Vectors . . . The input stimulus and output responses of a circuit designed to thoroughly test the circuits operation. Wtih AMAZE they can always be autogenerated.

INDEX

AMAZE BUG REPORT

Date Opened _____
Response Date _____
Date Closed _____

Submitted by: (PLEASE PRINT)

Name _____ Mail Stop _____ Date _____

Company _____

Address _____

City _____ State _____ ZIP _____

AMAZE Module: BLAST ☐ PTE ☐ SIM ☐ DPI ☐ PTP ☐
Pin Editor ☐ FTE ☐
STBC ☐
Assembler ☐

AMAZE Version Number: _____ Version Date: _____

Computer System Manufacturer and Type (IBM PC/XT/AT. . .): _____

Configuration (64K/CGA/EGA. . .): _____

DOS Type (PC/MS): _____ DOS Version: _____

Bug Description: _____

_____

_____

_____

_____

_____

Additional Required Information:

So that we may duplicate the problem, PLEASE attach the following:
— All design files (i.e., .PIN, .BEE (.SEE), and STD
— Any other file which may be related to the problem.
These files may be provided in printed form or preferably on a floppy diskette.

FOLD HERE

Signetics Company
811 E. Arques Avenue
P.O. Box 3409
Sunnyvale, California 94088-3409

Attn: MS25