PALASM 4 VERSION 1.5 RELEASE NOTES

Advanced Micro Devices reserves the right to make changes in specifications at any time and without notice. The information furnished by Advanced Micro Devices is believed to be accurate and reliable. However, no responsibility is assumed by Advanced Micro Devices for its use, nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Advanced Micro Devices.

MS-DOS and Windows are trademarks of Microsoft Corporation.

PAL and PALASM are registered trademarks, and MACH a trademark of Advanced Micro Devices, Inc.

QEMM is a registered trademark of Quarterdeck Office Systems.
DR-DOS is a trademark of Digital Research

TABLE OF CONTENTS

PALASM 4 VERSION 1.5 RELEASE NOTES

PREFACE

PALASM® 4 version 1.5, an upgrade to PALASM 4 version 1.4, supports the new MACH220 and MACH215 devices as well as existing AMD PAL® and MACH™ devices. PALASM 4 users need only read the portion of the release notes that pertain to the device(s) they currently use in their designs.

- If you are <u>not</u> using MACH devices, read chapters 1 and 2.
- If you <u>are</u> using MACH devices, read chapters 1, 2 and 3. Specific directions (☞) in each chapter will point you to those items new to the version 1.5 release. In addition, if you are using
  - MACH230 devices, read chapter 4
  - MACH215 devices, read chapter 5
  - MACH220 devices, read chapter 6

The appendices explain error messages and describe known problems.

Related documents -
- MACH Family Data Book (PID #14051 revision F or later)
- PALASM 4 User's Manual
- MACH Technical Briefs (PID #15972)
- 1992 PAL Device Data Book (PID #10173)

To obtain a copy of the latest databook or MACH Technical Briefs, contact your local AMD sales representative or call AMD Literature at (800) 222-9323.

CHAPTER 1. PALASM 4 VERSION 1.5 GENERAL NOTES

☞ **What to read in Chapter 1**

If you are a ...

Previous PALASM 4 v1.4 user	*read sections 1.1.2 - 1.5*
New user	*read all of chapter 1*

1.1 Operating System Compatibility

1.1.1 MS-DOS™ 5.0

PALASM 4 is compatible with MS-DOS™ version 5.0.

☞ 1.1.2 Memory Managers and the Extended Memory Version PALASM 4

The extended memory version of PALASM 4 requires a memory manager such as Microsoft's EMM386 or Quarterdeck's QEMM. Two tested configurations appear below.

```
          CONFIG.SYS file for Microsoft EMM386 memory manager:

DEVICE=C:\SYS\SETVER.EXE
DEVICE=C:\SYS\HIMEM.SYS
DOS=high,umb
device=C:\sys\emm386.exe 2048 m9
break = on
buffers = 10
files = 35
device=C:\sys\smartdrv.sys 2048 1024
devicehigh=C:\sys\ansi.sys
shell c:\command.com /e:1512 /p
```

Table 1. CONFIG.SYS for Microsoft EMM386

Note that the suggested CONFIG.SYS for QEMM (below) has changed; the information published in the PALASM 4 version 1.4 Release Notes was incorrect:

```
          CONFIG.SYS file for Quarterdeck's QEMM memory manager:

DEVICE=C:\QEMM\QEMM386.SYS EXTMEM=4096 RAM
rem  DEVICE=C:\QEMM\QEMM386.SYS RAM MAPS=12 HANDLES=96 NS -incorrect!!!
SHELL =c:\COMMAND.COM /E:880 /P
DOS=HIGH,UMB
break=on
FILES=20
BUFFERS=20
rem
install= c:\qemm\loadhi.com /tsr /r:3 c:\dos\fastopen.exe c:=80 /x
```

Table 2. CONFIG.SYS File compatible with MS-DOS 5.0 and EMM386

These configurations have been tested on a system configured with most utilities and device drivers in a \sys directory. Your system may be different.

☞ 1.1.3 DR-DOS

PALASM 4 version 1.5 is not supported under Digital Research's DR-DOS.

However, some users have run PALASM 4 successfully under DR-DOS 6.0 with QEMM 6.0, using the configuration below. AMD has not completely tested this configuration and therefore cannot assume responsibility for supporting customers in this environment.

```
                 DR-DOS Configuration - AUTOEXEC.BAT

@echo off
:DRDOSBEG
PATH C:\DRDOS;c:\QEMM;C:\pc;C:\PALASM\EXE;
@
LOADHI BUFFERS=40
LOADHI FILES=40
LOADHI=FCBS 4,4
VERIFY OFF
PROMPT $p$g
SET PALASM=C:\PALASM\
SET ORCAD=c:\ORCAD\
@
rem LOADHI C:DRDOS\SHARE.EXE /L:40   ---- MUST BE REMOVED!!!
cls
ECHO PALASM 4 Software    - (c)Copyright AMD 1992  All Rights Reserved
rem
:DRDOSEND
```

```
                 DR-DOS Configuration - CONFIG.SYS

DEVICE=C:\QEMM\QEMM386.SYS RAM NOSH
SHELL =c:\COMMAND.COM C:\ /P /E:512
BREAK=ON
BUFFERS=1
FILES=10
FCBS=4,4
FASTOPEN=512
LASTDRIVE=E
HISTORY=ON, 256, ON, OFF, OFF
COUNTRY=001,,C:\DRDOS\COUNTRY.SYS
HIDOS=ON
DEVICE=C:QEMM\LOADHI.SYS C:\DRDOS\ANSI.SYS
```

Table 3. AUTOEXEC.BAT and CONFIG.SYS files for DR-DOS 6.0 with QEMM 6.0

Again, AMD has not completely tested this configuration and cannot assume responsibility for supporting customers in this environment.

☞ 1.1.4 Microsoft Windows®

PALASM 4 version 1.5 is not a WINDOWS program and must be run as a stand-alone program.

☞ 1.1.5 IBM OS/2™

PALASM 4 version 1.5 is not supported under and is not compatible with OS/2.

1.1.6 Networks

PALASM is not supported for use in networked environments.

☞ 1.2 Changes to the Installation Program

PALASM 4 version 1.5 software takes 5-7 MB hard disk space, depending on whether the regular or extended memory version is installed. Beginning with this release, the INSTALL program now asks whether to load the Examples and On-line Help files. Not loading these files will save disk space.

```
Install Examples      350KB (Y/N) ? Y
Install On Line Help 750KB (Y/N) ? Y
```

Even if the Examples are not installed, the \PALASM\EXAMPLE subdirectory and its associated subdirectories are still created by the INSTALL program. These empty subdirectories may be deleted.

The same is true of the On-Line Help. The INSTALL program creates subdirectories within \PALASM\DOC directory, even if the On-line Help is not installed. With the exception of the LOOKUP.HLP file (used the PALASM 4 menu system), the empty subdirectories under \PALASM\DOC may be deleted.

☞ 1.3 Installing From an Alternate Drive

To install PALASM 4 or the MACH Libraries from a drive other than "A", use the letter of the drive you are installing from both before and after the INSTALL command. For example:

```
B:INSTALL B:
```

This command will install the software from the B drive onto whatever hard drive is chosen later in the installation program.

☞ 1.4 Extended Memory Problems Reported With IBM/AT®s and Compatibles

1.4.1 Designs Not Compiling

Problems have been reported running the extended memory version of PALASM 4 on IBM AT (80286) compatibles. These same PCs are able to run the standard memory version successfully. When the problem occurs, compilation of a design would begin under the extended memory version, only to have the menu reappear in a few seconds with no error message or results.

The problem occurs because the INSTALL program, which calls a "tuning" program, did not successfully adapt the

extended memory version of PALASM 4 to your PC. The tuning program is not called if you are using extended memory memory manager or a machine-specific memory access program.

Work Around: Try "tuning" PALASM 4 manually. After installing this release, type:

```
C:> TUNE_MAN 1
```

The numerical arguments 1, 2 or 3 implement different tuning strategies effective on many PCs. If the same problem in running the extended memory version of PALASM 4 continues, try running the TUNE_MAN program again using the numerical argument 2 or 3 instead. PALASM 4 must be already installed in order to use TUNE_MAN. It does not have to be installed again from floppy disks each time, or re-run each time you reboot or reprocess a design.

1.4.2 BIOS ROMs and Early IBM/AT Compatibles

Some customers with early IBM/AT compatibles have reported problems with the extended memory version of PALASM 4. The problem has been traced to problems in early versions of the ROM BIOS shipped with IBM/AT compatibles manufactured before 1989. If you have one of these PCs, consider upgrading the BIOS ROMs. This inexpensive investment will solve problems with many software packages, not just PALASM 4.

☞ 1.5 Documentation Errors

The following error appears in all copies of the PALASM 4 User's Manual.

page 11-167	Defining Clock Pins: Pin 16 should be listed as CLK1 for the MACH120 and 220 devices, not pin 17.

The errors listed below (previously described in PALASM 4 version 1.4) appear in older, three-hole-punched PALASM 4 User's Manuals acquired prior to August 1992.

page 11-169	Pin Declaration Segment: Pin 4 should be REGISTERED, not COMBINATORIAL. Pin 4 IO2 ~~COMBINATORIAL~~ **REGISTERED**
page 11-170	Top of page: Change the T-registered signal inside the Note as follows **A6.T: = {IO2.T}**
page 11-171	Top of page: Change the Pin Declarations as follows Pin ? ~~IO2~~ **IO1** REG Node ? A4 PAIR IO2 IO1 REG
page 11-171	Middle of page: Change the Pin Declaration to add PAIR A3 to Pin 33 Pin 33 I4 **PAIR A3**
page 11-172	Top of page: Change the Pin Declaration to add PAIR A3 to Pin 33 Pin 33 I4 **PAIR A3**

☞ **What to read in Chapter 2**

If you are a ....
Previous PALASM 4 v1.4 user   *read just section 2.2*
New user   *read all of chapter 2*

2.1 Menu Changes - Minimizer

Beginning with PALASM 4 version 1.4, a new Minimizer was added which has a new Logic Synthesis menu option and two improvements. The two improvements are -

- It reduces the memory utilization of most designs by 50% or more.

- The old Minimizer "hung" if more than 14 variables were used in an equation, while the new one is guaranteed to reduce 32 variable equations.

The menu option shown in Figure 1, *Use fast minimization*, allows the user to run an abbreviated version of the new Minimizer. If it is OFF, a more exhaustive minimization is performed. It should be turned ON only if an *Out of memory* error is generated or Minimizer compilation times are unusually long, since the *fast minimization option* may produce equations with more product terms than the standard Minimizer does.

```
┌──────────────────────────────────────────────────────────────────────────────────┐
│ ══════════════════════════ PALASM4 version 1.5 ══════════════════════════          │
│  FILE   EDIT   RUN   VIEW   DOWNLOAD   DOCUMENTATION   <F1> for Help                │
│  ┌─────────────────────────────────┐                                               │
│  │ Begin new design                │                                               │
│  │ Retrieve existing design        │                                               │
│  │ Merge design files              │                                               │
│  │ Change directory                │                                               │
│  │ Delete specified files          │                                               │
│  │ Set up . . .                    │                                               │
│  │   ┌──────────────────────────────────┐                                          │
│  └───│ Working environment              │                                          │
│      │ Compilation options              │                                          │
│      │ Simulation options               │                                          │
│      │ Logic synthesis options          │                                          │
│      │ ══════════════ LOGIC SYNTHESIS OPTIONS ══════════════                        │
│      │  Use automatic pin/node pairing?         Y                                   │
│      │  Use automatic gate splitting?           N   . . . if 'Y', Max = 4           │
│      │  Optimize registers for D/T-type         As specified in design file         │
│      │  Ensure polarity after minimization is   As specified in design file         │
│      │  Use 'IF-THEN-ELSE','CASE' default as     Don't care                         │
│      │  Use fast minimization?                  N                                   │
│                                                                                    │
│  >  move cursor,<F10> form ok, <Esc>                                               │
└──────────────────────────────────────────────────────────────────────────────────┘
```

Figure 1. Logic Synthesis Options Menu.

☞ 2.2 Simulator Usage Notes

The Simulator has been revised for PALASM 4 version 1.5. Most devices, including MACH devices are affected.

2.2.1 Devices Still Supported by the Old Simulator (SIM)

The following devices are supported by the old Simulator and are not affected by the changes discussed in sections 2.2.3 through 2.2.11:

- PAL16RA8
- PAL20RA10
- PAL22IP6
- AMPAL23S8
- PAL32R16
- PAL64R32

2.2.2 Assumptions Made by Previous Versions of the Current Simulator (PLDSIM)

The revised PALASM 4 version 1.5 Simulator is less tolerant of incompletely specified simulation files. Some previously clean simulation files may now generate error messages and simulation miscompares, or fail programmer JEDEC verification if the simulation file is incompletely specified.

Previous versions of this Simulator (PLDSIM) made several assumptions based on the evaluation of the programmer at power-up that sometimes resulted in devices being modeled incorrectly:

- Registers and latches were incorrectly modeled:
    - Registers powered up to a known state regardless of the state of the input pins.
    - The effect of the control signals on the state of the registers was not evaluated at power-up.
    - Clocks were triggered if clock fan-in went from a low or an unknown to a high.
- MACH devices were incompletely modeled.
- After preload, there was no evaluation of the effect of the control signals on the state of the registers. ·
- An unknown SET or RESET or CLOCK had no effect on the state of the registers.
- Pins that where unassigned were not evaluated.

2.2.3 Correct Modeling of Registers and Latches

The PALASM 4 version 1.5 Simulator corrects modeling problems with registers and latches:

- Register and latches with unknown SET, PRE-SET or CLK/LE signals now generate an unknown state at the output Q.
- The clock will now trigger only if it rises from a low to a high in the case of an active high clock, or from high to a low in the case of an active-low clock.
- The Simulator now reflects the specifications in the MACH Family Data Book. For example, it is legal to have SET and RESET signals high at the same time.
- For devices with preload pins, asserting PRELOAD with SET or RESET results in an unknown fanout at the register output Q.

2.2.4 Programer Emulation at Power-Up

PLD programmers and testers forced a default test condition on pins set to unknown logic states. On some programmers the default test condition value is programmable. The JEDEC format for the default test condition is "X0*" for a LOW state, and "X1*" for a HIGH state. This field must be placed before the first test vector and after the number of pins (QP) and the number of fuses (QF) fields.

Nearly all AMD approved programmers support the default test condition X0*. The new Simulator assumes that all pins are

forced to a SOFT-LOW before power-up, and places a "X0*" before the first test vector.

The unknown state cannot be realized in a physical sense in hardware. Individual programmers can set pins HIGH or LOW, and some programmers are even able to set pins to float (normally not done because the effect of a floating pin cannot be determined for all devices and test cases).

Uninitialized pins are generally set to the default test condition before the power is turned on to the device under test. Programmers cannot determine which pins are inputs and which are outputs, and therefore must use "soft conditions". Under soft conditions, pins are driven high or low by a resistance low enough to drive an input pin but not low enough to override or destroy an output pin.

The Simulator shows the default test condition for uninitialized pins in the history file. However, uninitialized pins remain as "X"s or unknown in the JEDEC test vectors because some testers have a limit on the amount of pins that can be toggled on each single vector.

2.2.5 Power-Up Sequence

A two stage power-up routine was added to the Simulator to simulate the behavior of registered devices better. The routine evaluates the device state before the first user vector is applied, and takes in account all control signals connected to registers.

Stage 1

- Load all inputs with the default condition (currently 0) and enter affected signals into event queue.
- Load all registers with power-up preload value and enter affected signals into event queue.
- Fix registers so they will not change in response to control signals CLK/LE, SET, RESET, or PRELOAD.
- Evaluate until steady state.

Stage 2

- Load all inputs with the default condition (currently 0) and enter affected signals into event queue.
- Allow registers to be affected by control signals.
- Evaluate until steady state.

2.2.6 Software Preload Sequence

A two stage preload routine was added to the Simulator for the same reasons as at power-up. Control signals such as

SET, RESET, CLK/LE and output enables can affect the register states after they have been preloaded.

Stage 1
- Load all registers with the preload value and enter affected signals into the event queue.
- Fix registers so they will not change in response to the control signals CLK/LE, SET, RESET, and PRELOAD.
- Evaluate until steady state.

Stage 2
- Allow registers to be affected by control signals.
- Evaluate until steady state.

When the software encounters the PRELOAD statement, test vectors are no longer appended to the JEDEC file.

2.2.7 Full Evaluation of Input Pins

- All input pins are assumed to be initialized to the default test conditions at power-up.
- The effect of all input pins is now evaluated upon power-up.

2.2.8 Driving Active-Low and Active-High Clocks for the MACH215

The MACH215 and future MACH devices with active-low clocks can be driven with an active-low clock signal at the pin. An active-low clock (a JEDEC "K" clock) is a high-to-low-to-high pulse.

Polarity conventions are consistent with the polarity convention for the SETF command:
- To generate a JEDEC "C" clock force for the active-high pin, CLK, use the simulator command "CLOCKF CLK". If the pin CLK is active-low, use the simulator command "CLOCKF /CLK".
- To generate a JEDEC "K" clock force for the active-high pin, CLK, use the simulator command "CLOCKF /CLK". If the pin CLK is active-low, use the simulator command "CLOCKF CLK".

An example is shown on the following page.

```
          PIN 1 CLK1                  ; Active-high pin
          PIN 2 /CLK2                 ; Active-low pin
SIMULATION
; For active-high pins
          SETF CLK1                   ; Generates JEDEC "1" force
          SETF / CLK1                 ; Generates JEDEC "0" force
; For active-low pins
          SETF CLK2                   ; Generates JEDEC "0" force
          SETF / CLK2                 ; Generates JEDEC "1" force
; Global clock
          CLOCKF                      ; Generates JEDEC "C" clock
; For active-high pins
          CLOCKF CLK1                 ; Generates JEDEC "C" clock
          CLOCKF / CLK1               ; Generates JEDEC "K" clock
; For active-low pins
          CLOCKF CLK2                 ; Generates JEDEC "K" clock
          CLOCKF / CLK2               ; Generates JEDEC "C" clock
```

Table 4. Active-Low and Active-High Clocks for the MACH 215
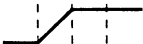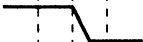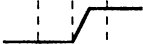
Waveform	Test Condition	Description
	0	Drive input low
	1	Drive input high
	C	Drive input low, high, low
	D	Drive input low, fast transistion
	K	Drive input high, low, high
	U	Drive input high, fast transistion

Table 5. Driving Test Conditions.

Errors are generated if a "C" clock is asserted on a pin whose state is initially high, or if a "K" clock is asserted on a pin whose state is initially low.

2.2.9 Product-Term-Driven Clocks

The PALASM 4 version 1.5 Simulator supports JEDEC "U" and "D" transitions for dedicated clock pins. If a transition occurs on a dedicated clock pin it is translated to a "U" or "D" transition.

The purpose of "U" and "D" clocks is to allow data from all other inputs to be stable before a latch enable or clock transition occurs. Some dedicated clock pins can be used both as clock and as data pins. The user should be aware that on a JEDEC tester this can cause some data lines to be driven at the same time or later than clock signals.

To avoid potential test problems with the simulator command "SETF" the data and CLOCK/LE transitions should occur in separate test vectors. "C" and "K" clock transitions should be used to drive pins that affect register clocks.

The version 1.5 Simulator supports both fast rise and fall transitions for dedicated clock pins on all devices controlled by the SETF syntax. Data from all other inputs must be stable before a latch enable or a clock transition occurs.

Because some dedicated clock pins can be used as both clock and data pins, customers should be aware that some data lines could be driven at the same time or later than clock signals on a JEDEC tester, leading to differences between simulated and observed programmer behavior.

To avoid this problem, Simulator SETF commands and test patterns should be written so that data and CLOCK/LE transitions occur in separate vectors. AMD recommends that CLOCKF commands drive pins that affect register clocks.

Fast rise and fall transitions are not supported for devices with non-dedicated clock pins driving product term clock resources. Unless the programmer is set up to create fast edges on all input transitions, input changes are too slow to properly trigger CMOS storage elements on devices such as the MACH215. Any JEDEC vector verification errors must be individually analyzed to determine the cause of the miscompare.

2.2.10 Simultaneous Events

While the MACH devices allow the application of SET and RESET signals at the same time, removing both signals at the same time results in an unknown state. Because the PALASM Simulator is a functional simulator, not a timing simulator, it cannot detect simultaneous events. Take care to remove SET and RESET signals in separate vectors.

2.2.11 Power-up Preload on Floating Pins

The PALASM 4 version 1.5 Simulator requires a physical location to preload pins with a power-up state. If there are floating pins, the register value will be set to "X" (the unknown value) at start-up. As a result, some test cases will generate different results if they are executed with floating pins. The work around is to manually place registered pins and nodes.

2.2.12 Simulation of Three-State Pins for Input and Output

The PALASM 4 version 1.5 Simulator may not always properly chose the input symbol set over the output one when a common clock is used to both load an input register and control the output enable.

When the effect of using a single vector for both input and output operations is considered, the problem becomes apparent. When a pin changes from an input logic state (represented by "0" or "1" in the JEDEC signal vector) to an output (represented by "H" or "L") a conflicts arises between which symbol should be used to denote the signal level during that window. Erroneous results may occur:
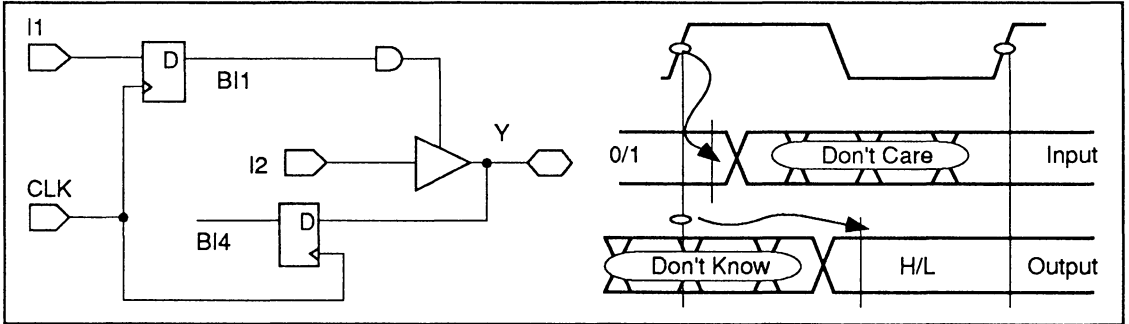
Figure 2. Multi-functional Pin Usage

To avoid this problem, separate the three-state control from the clock event on the input register by adding an extra input to the three-state control PT.

2.2.13 Input Signal Ordering

Programmers apply inputs to a device in different sequences. Some apply inputs in sequential pin order, some apply them in groups of eight pins at a time, and others follow still other patterns. With so many possibilities, no simulator can handle all situations.

Therefore, users should define vectors with device logic in mind, avoiding potential races in vector definition so that any variation in the input sequences will produce the same result. In conventional synchronous logic, this is not a problem - all data input transitions are applied before the device is clocked. However, simultaneous clock events should be avoided.

The problem is more difficult for asynchronous logic designs. Control functions like set and reset should be applied and removed in separate vectors with an "idle" state in between, so that even if input changes are skewed, both will not be applied simultaneously. Likewise, data changes should be separated from storage-enabling or clocking events, so that the ordering of input changes is less likely to have an effect on the output.

CHAPTER 3. AMD MACH FITTER VERSION 1.5

☞ **What to read in Chapter 3**

If you are using <u>MACH devices</u> and are a ....

Previous PALASM 4 v1.4 user *read sections 3.1.3, 3.2, 3.2.1, 3.2.2, 3.2.3, 3.3, 3.3.1, 3.3.2, 3.3, 3.4, 3.4.1 and 3.4.2*

New user *read all of chapter 3*

3.1 General Operation

This section describes the MACH fitting process section (3.1.1) and two traits of the AMD Fitter. These are -

- If more than two tristates are used in a MACH1XX block, the logic may not fit. See section 3.1.2.
- Some designs which fit when compiled under previous versions of PALASM 4 may not fit when compiled with subsequent versions. See section 3.1.3.

3.1.1 Fitting Process

Pin and node locations should not be specified before the design is compiled. Users should follow the following steps-

- First, float pins in the design and compile.
- Next, manually partition the design and re-compile (if necessary).
- After the design is fitted, back-annotate pin and node locations and re-compile.

Figure 3 shows the processes for fitting MACH230 and other MACH designs. Note that optimization of fitted designs is done so that a design meets speed requirements and is more likely to accept future design modifications without changing the pinout. (Refer to the *MACH Technical Brief* titled, "Designing for Change with MACH Devices".)
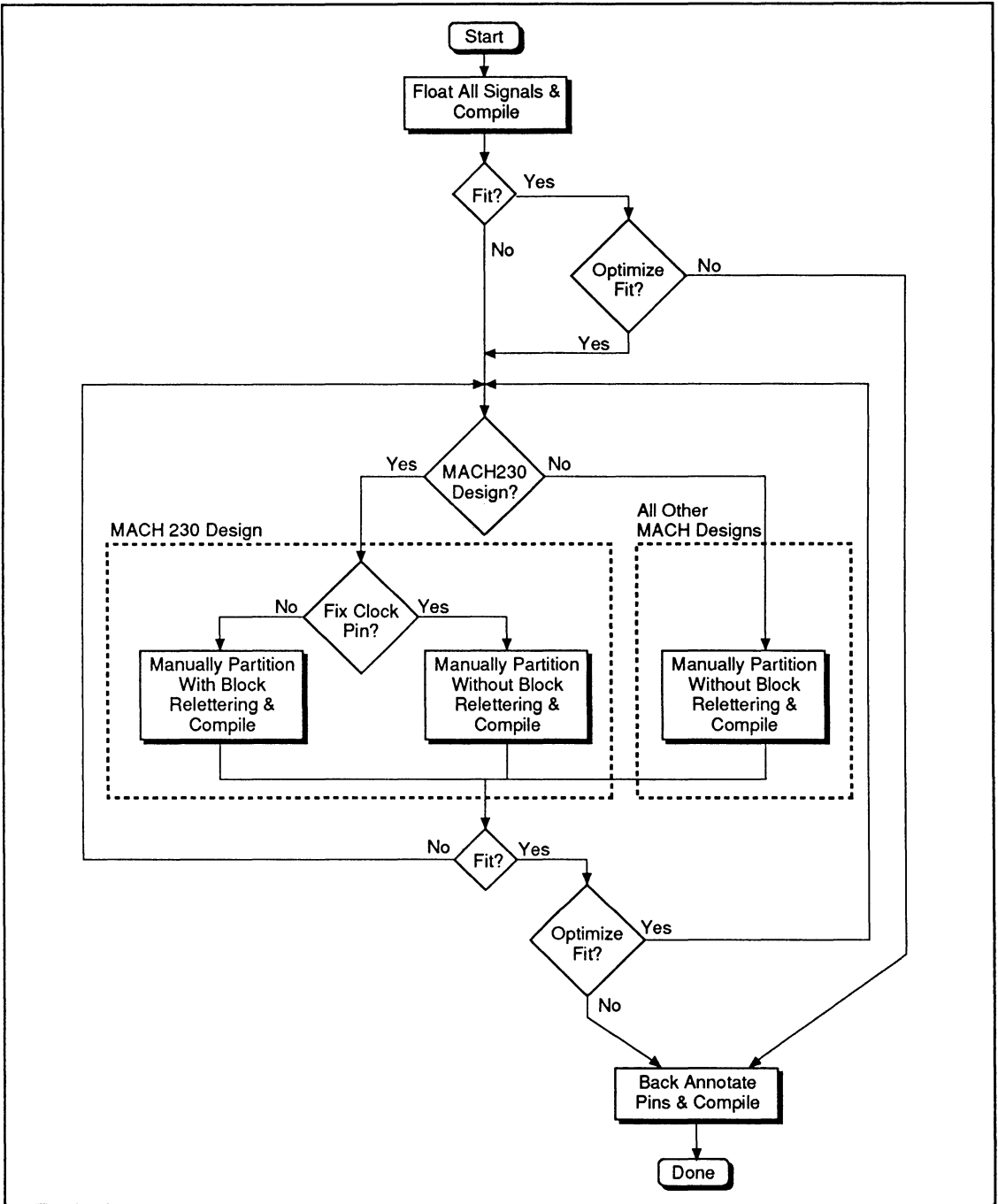
Figure 3. MACH Fitting Process.

The fitting process differs if the pinout is fixed. When changes must be implemented to the design after the pinout for the design has been fixed, the process illustrated in Figure 3 is not applicable. In order to maintain the existing pinout, the design should be compiled with the pins corresponding to the original design fixed. New signals may be floated to aid in fitting. Once the design is fitted, all pins should be back-annotated and the design re-fitted.

3.1.2 Tristate Resources in MACH 1 Devices

The tristate resources in a MACH 1 device blocks are segmented into two banks of macrocells, one bank associated with macrocells 0-7 and the other associated with macrocells 8-15. Two tristate resources are provided for each bank, as shown in Figure 4.
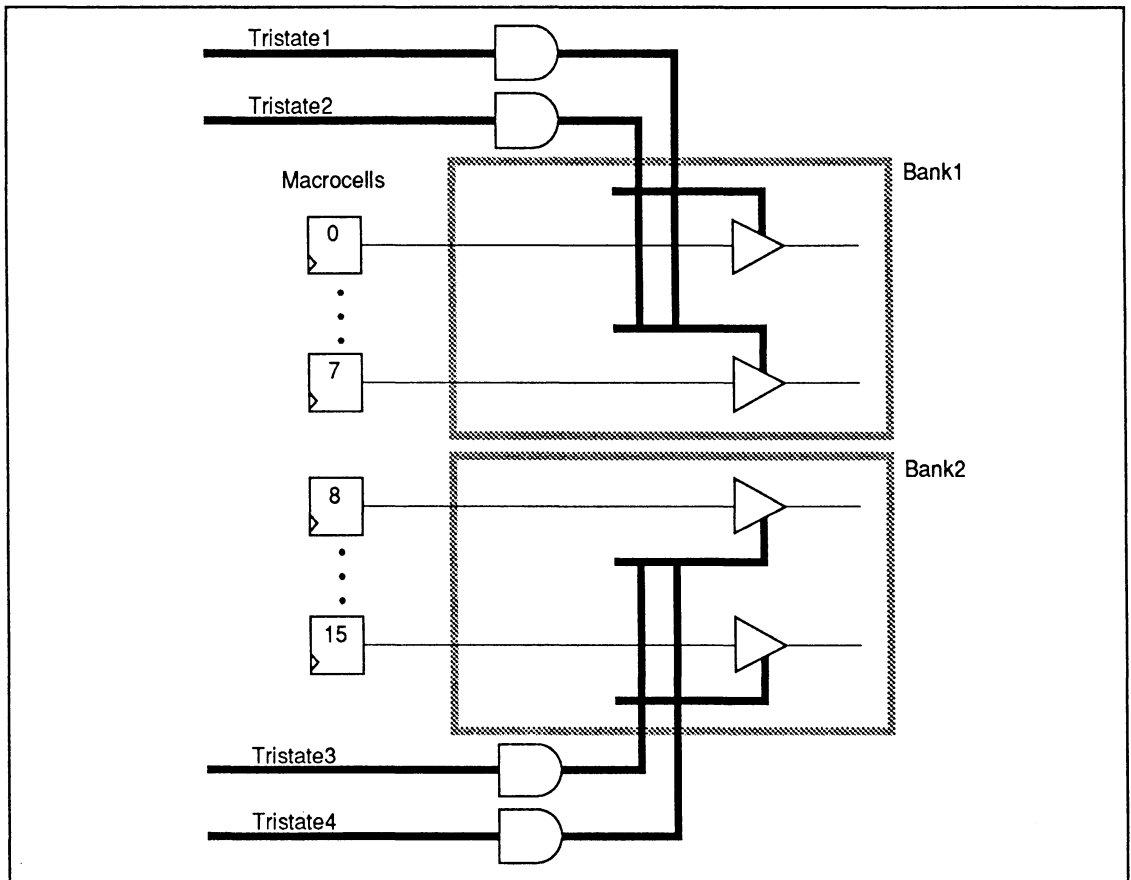
Figure 4. Tristates in a MACH 1 Device Block.

If two or fewer independent tristate resources are used in a block, tristate placement will not affect fitting. However, if more than two tristates are used in a block, macrocells using common tristates must be placed in the same bank. Since the Fitter doesn't take banking into account, it may place equations into a block so that only two tristate signals are available per block.

An example will clarify this. Consider the case where two equations (X & Z) use tristate signal A, and two (W & Y) use tristate signal B. By placing equations using both tristate signals into each bank as illustrated in Figure 5, the Fitter uses all tristate resources in that block.
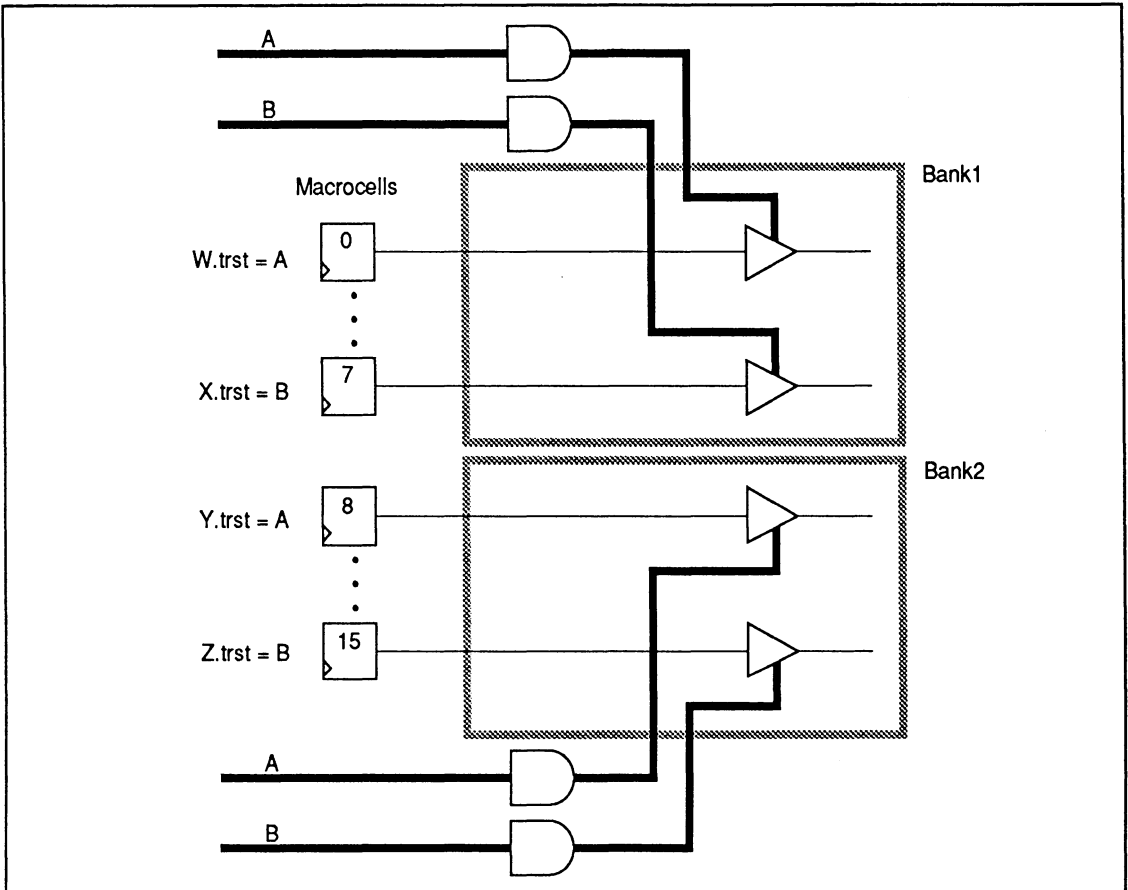
Figure 5. Non-optimal placement of equations W, X, Y and Z uses all tristates.

If, however, the two equations using tristate A are placed in one bank and the two equations using tristate B are placed in the other, two "free" tristate resources are still available for use. See Figure 6.
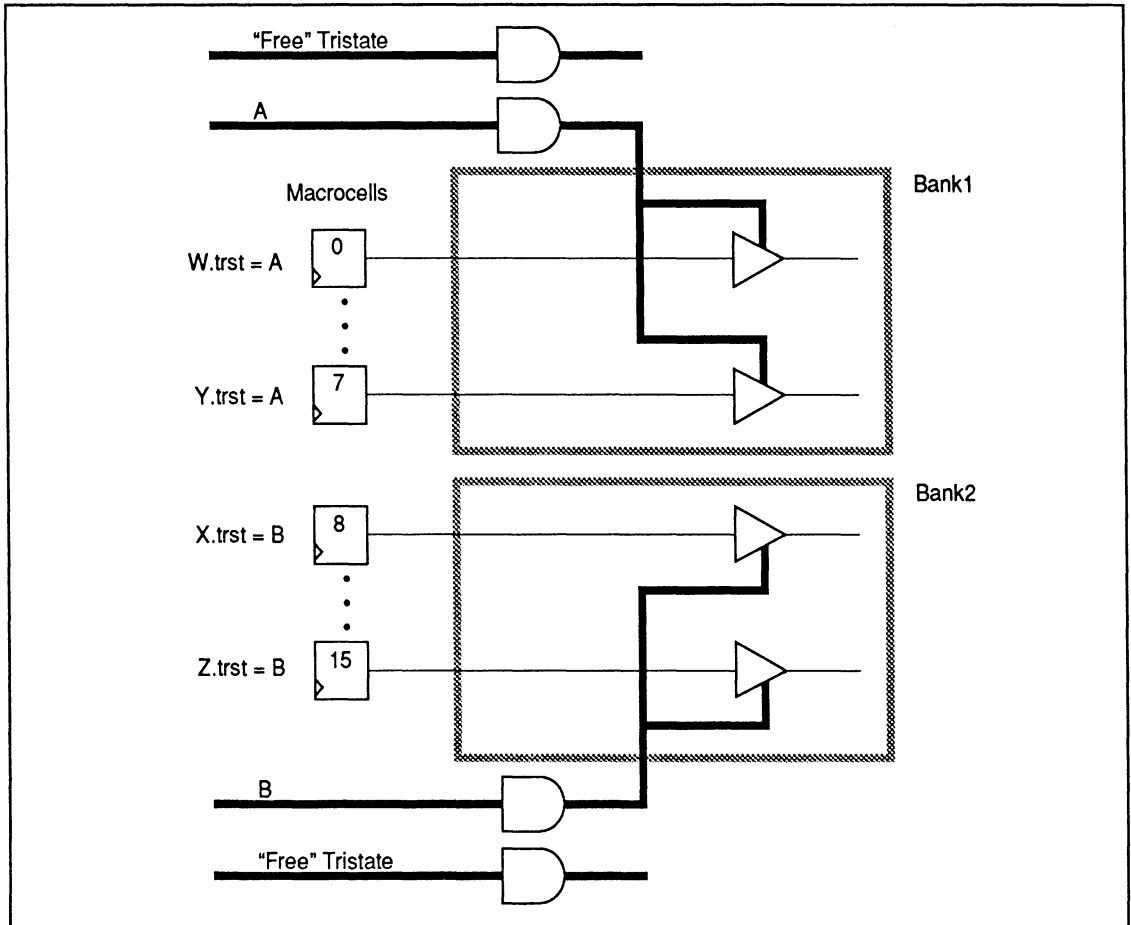
Figure 6. Optimal placement of equations W, X, Y and Z leaves 2 "Free" Tristates.

The following steps should be taken to guide the Fitter in placing logic when three or four tristate signals are used in a block and an error message indicates that too many tristates are used -

- Refit the design with the tristate signals removed from the design file, then fit the design with pin placements from this design back-annotated and the tristate signals added.

- If this doesn't result in a successful fit, the equations should be placed in appropriate banks

by specifying pin numbers for them in the declaration section of the design file.

- If the design still will not fit, partition it into blocks manually to move tristate resources into other blocks. (Refer to the "MACH Devices - Manual Logic Partitioning" *MACH Technical Brief.*)

☞ 3.1.3 How to Use Latches on MACH Devices

MACH1XX and 2XX devices, excluding the MACH215, only support active low latches. The correct syntax for the latch enable for these MACH devices is listed below:

```
                /signal.clkf =
```

The MACH215 device, however, has both active low and active high latches. The correct syntax for enabling active-low and active-high latches is listed below:

```
/signal.clkf = ...       ;active low

signal.clkf = ...        ;active high
```

The following example shows how to use latch enables in design files for MACH devices other than the MACH215:

```
Pin ? A0 LATCHED         ; output
Node ANO LATCHED         ; output
Pin ? LATCH
Pin ? B0                 ; input
Pin ? B1                 ; input
Pin ? LE                 ; input

A0 *=  B0 + B1
ANO *= {A0}
/LATCH.CLKF = LE
```

Because the MACH110, 120, 130, 210, 220 and 230 devices do not support active high latches, if an active high latched equation such as:

```
            LATCH.CLKF = LE
```

is used in this example, the Fitter will generate the following error message:

```
>ERROR F570 - Invalid Signal for MACH Arch.  (Logic on Clk Eq) - ANO
-----------------------------------------------------------------
An Auxiliary equation's (Trst/Clk/Setf/Rstf) logic configuration was
detected that is not compatible with the MACH device architecture.
Please correct the design & rerun all processing steps.
```

3.1.4 Save Old Versions of the AMD Fitter and PALASM Software

New versions of the AMD Fitter software are not guaranteed to fit designs which old versions did. Old versions of the software used to fit existing designs should be saved in case changes must be made to the design.

As a precaution, save the original PALASM 4 software disks from other releases.

☞ 3.2 New Behavioral Features of the AMD MACH Fitter

Several new features, applicable to all MACH devices, have been added to the Fitter with PALASM 4 version 1.5, in addition to those previously added with version 1.4:

- New Fitter features in PALASM 4 version 1.5
  - Product Term Reservation for Logic Functions (section 3.2.1)
  - Unused Input Registers are now ignored by the Fitter (section 3.2.2)
  - Reduced block limits. This feature was first implemented in PALASM 4 version 1.4. Turning off "Maximize packing of logic blocks" limits the amount of logic the Fitter places in a block. The resource limits table has been updated to include the new MACH215 device (section 3.2.3).
- New Fitter Features in PALASM 4 version 1.4
  - Product Term Redistibution. Beginning with PALASM 4 version 1.4, the Fitter redistributes product terms within a block until a fit is found (section 3.2.4).
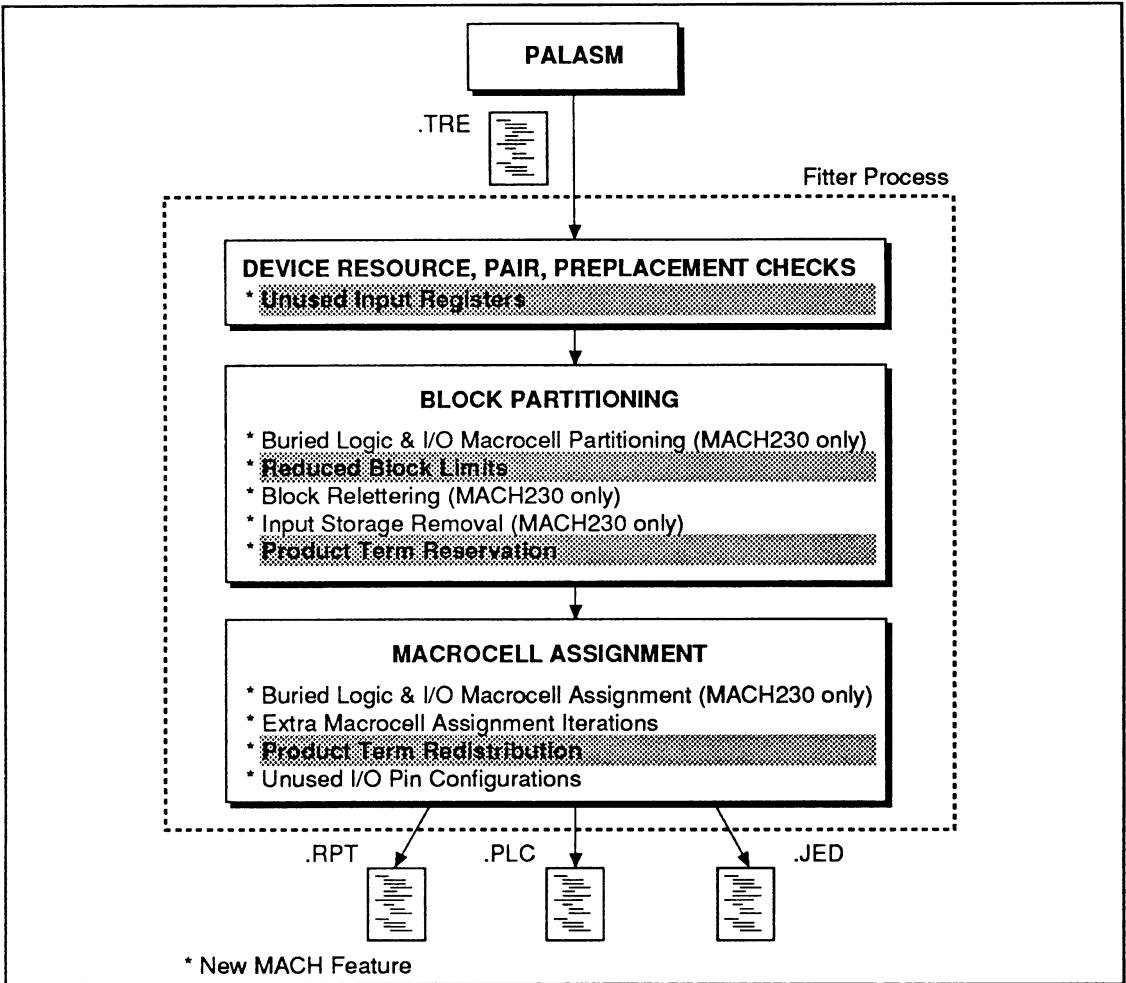
Figure 7. Flowchart of the Fitter process with new Fitter features highlighted.

The MACH230 specific features are discussed in Chapter 4.

If you are already familiar with PALASM 4 version 1.4, read sections 3.2.1 through 3.2.3 before moving to section 3.3.

☞ 3.2.1 Product Term Reservation for Logic Functions

In the past, all logic functions placed by the Fitter were allocated only the exact number of product term clusters needed to realize the desired logic. The new "Product Term Reservation" feature allows you to choose a placement of free product terms directly for a particular logic function.

The syntax for product term reservation is similar to definition of logical groups:

```
GROUP MACH_PTS_8 Sig_List
```

The number used with the GROUP MACH_PTS statement (i.e. 8) sets the <u>minimum</u> space allocated to each of the signals listed.

Allocating additional terms to logic functions can use up available resources within a block and can reduce the chances of a successful fit. As a reminder, PALASM 4 will generate the informational message below whenever Product Term Reservation is used and a design fails to fit:

```
|> INFORMATION F062 - PT Expansion Enabled
------------------------------------------------------------------
Explanation:
 Design for Change PT expansion was enabled for this design, and the
design failed fitting. You may want to reevaluate the PT expansion used
and try refitting the design.
```

See also the MACH Technical Brief titled "Designing for Change with MACH Devices".

☞ 3.2.2 Unused Input Registers Now Ignored by the Fitter

Unused pins, buried nodes and input registers are now ignored in all fitting operations.

In the past these items were connected using conventional array logic transfers and consumed PT resources. Unused registered inputs are now highlighted with the following warning message:

```
|> WARNING F179 - Pair Declaration Ignored: (Unused RI) - Sig_Name
```

3.2.3 Reduced Block Limits

Early versions (pre-PALASM 4 version 1.4) of the Fitter sometimes partitioned too much logic into one block even when other blocks were unused, and then could not fit the design. By reducing block limits, the software attempts to overcome this problem through limiting how much logic the Fitter places in a block.

If the *Maximize packing of logic blocks* Fitter flag is OFF, this feature is enabled and the Fitter uses reduced resource limits to determine how much logic to place in a block, as illustrated in Table 6. If, in contrast, the *Maximize packing of logic blocks* flag is ON, the Fitter uses the limits listed in the ">70%" column.

```
Resource                                 D.U.<50%   50%>D.U.<70%   D.U.>70%

☞ Array Inputs (MACH120,130,220,230)      23           25            26
☞ Array Inputs (MACH110,210,215)          19           21            22
Product Terms (MACH110,130,210,230)       48           56            64
Product Terms (MACH120,220)               32           40            48
☞ Product Terms (MACH215)                 16           24            32
I/O Macros (MACH110,130)                  14           15            16
I/O Macros (MACH210,230)                   6            7             8
I/O Macros (MACH120)                      10           11            12
☞ I/O Macros (MACH220)                     4            5             6
☞ I/O Macros (MACH215)                     6            7             8

D.U. = Device Utilization
```

Table 6. Reduced Block Limits (*Maximize packing of logic blocks* flag OFF).

The benefit of this approach is that it spaces logic evenly among blocks in a MACH device, allowing designs to fit that otherwise would not.

The drawback is that this Fitter may not successfully partition some high utilization designs which previous versions did.

To prevent the software from reducing block limits, run the Fitter with the *Maximize packing of logic blocks* flag ON using the *Select one combination* Fitter option or manually partition the design.

3.2.4 Product Term Redistribution

Previous versions of the Fitter allocated product terms to macrocells within a block using a fixed algorithm that did not redistribute product terms if a particular allocation would not fit. In contrast, the PALASM 4 version 1.4 Fitter redistributes product terms within a block until a fit is found.

The benefit of this feature is that the PALASM 4 version 1.4 software fits some designs that earlier versions could not.

☞ 3.3 MACH Fitting Options Menu Changes

Changes to the MACH Fitting Options menu in both PALASM 4 version 1.4 and version 1.5 have added extra functionality to the Fitter, as shown in Figure 8.

```
╔══════════════════ MACH FITTING OPTIONS ══════════════════╗
║  OUTPUT:                                                   ║
║     Report level                        Detailed          ║
║                                                            ║
║  SIGNAL PLACEMENT:                                         ║
║     Float all signals and ignore grouping?    N           ║
║     Use placement data from             Design file       ║
║     Save last successful placement      <F3>              ║
║     Press <F9> to edit file containing  Last successful placement ║
║                                                            ║
║  PARTITION and Re-Fit?                        Y            ║
║                                                            ║
║  DEFAULT FUSE OPTIONS:                                     ║
║     Unused I/O pins configured as outputs?  Y ... If 'Y', Level = H ║
║                                                            ║
║  FITTING OPTIONS:                                          ║
║     When compiling                      Run until 1st success: EXTRA ║
╚════════════════════════════════════════════════════════════╝
```

Figure 8. MACH Fitting Options Menu.

Changes made to the MACH Fitting Options menu in PALASM 4 version 1.5

- Partition and Refit improves the likelihood of finding logic partitions after an unsuccessful fit for MACH110, MACH210, MACH130, MACH120, and MACH220 devices (section 3.3.1).
- Default Fuse Options allow you to control the logic state unused pins are driven to (section 3.3.2).

Changes made to the MACH Fitting Options menu in PALASM 4 version 1.4

- Improve the chance of fitting by running extra iterations for macrocell assignment (section 3.3.3).

If you are already familiar with PALASM 4 version 1.4, read sections 3.3.1 and 3.3.2 before moving to section 3.4.

☞ 3.3.1 Partition and Re-Fit

After obtaining an unsuccessful fit with the PALASM 4 version 1.5 Fitter, you now have the option of calling an improved partitioning program for solving such problems. This new fitting option - "Partition and Refit" - can be used with all MACH devices except the MACH215 and MACH230 devices.

```
┌─────────────────────────────────────────────────────────────────────┐
│ ┌─────────────────────── MACH FITTING OPTIONS ═══════════════════┐   │
│ │ OUTPUT:                                                         │   │
│ │     Report level                         Detailed              │   │
│ │                                                                 │   │
│ │ SIGNAL PLACEMENT:                                               │   │
│ │     Float all signals and ignore grouping?   N                 │   │
│ │     Use placement data from              Design file           │   │
│ │     Save last successful placement       <F3>                  │   │
│ │     Press <F9> to edit file containing   Last successful placement │
│ │                                                                 │   │
│ │ ▓▓PARTITION and Re-Fit?▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓Y▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓ │   │
│ │                                                                 │   │
│ │ DEFAULT FUSE OPTIONS:                                           │   │
│ │     Unused I/O pins configured as outputs?   Y ... if 'Y', Level = H │
│ │                                                                 │   │
│ │ FITTING OPTIONS:                                                │   │
│ │     When compiling                       Run until 1st success: EXTRA │
│ └─────────────────────────────────────────────────────────────────┘   │
└─────────────────────────────────────────────────────────────────────┘
```

Figure 9. MACH Fitting Options - Partition and Re-Fit.

By using the "Partition and Refit" option, you will have a somewhat higher success rate in finding logic partitions. The improved partitioning program optimizes the block placement of logic functions to minimize the required interconnect. It continues to swap functions while improving the chance of a successful fit.

The "Partition and Re-fit" option can only be used <u>after</u> running the Fitter and getting an <u>unsuccessful</u> fit.

When enabled, the Partition and Re-Fit option asks if the block partitions (GROUP MACH_SEG commands) used in the design file should be kept:

```
Partition option: KEEP GROUP MACH_SEGs (Y/N)? N
```

Choosing "N" allows the Fitter to freely move all signals as needed. Choosing "Y" allows the Fitter to move only those signals not declared in GROUP MACH_SEG statements in the design file, and possibly keep much of the old pinout.

☞ 3.3.2 "Default Fuse Options" Controlling the Logic State Unused Pins Are Driven To

Unused device pins can potentially conduct switching noise into the logic arrays and float between logic thresholds, dissipating additional power. Proper signal termination with a pull-up resistor (>20kΩ) can prevent this from happening, but at the cost of additional components. New MACH devices such as the MACH220 and MACH215 have

internal pull-ups, but older MACH devices, such as the MACH110, 210, 130, 230 and 120 do not.

PALASM 4 version 1.5 allows the user to choose the logic state (high or low) that unused pins are driven to from the MACH Fitting Options menu:

```
╔══════════════════ MACH FITTING OPTIONS ══════════════════╗
║ OUTPUT:                                                    ║
║    Report level                       Detailed            ║
║                                                            ║
║ SIGNAL PLACEMENT:                                          ║
║    Float all signals and ignore grouping?   N             ║
║    Use placement data from            Design file         ║
║    Save last successful placement     <F3>                ║
║    Press <F9> to edit file containing Last successful placement ║
║                                                            ║
║ PARTITION and Re-Fit?                 Y                    ║
║                                                            ║
║ ░DEFAULT FUSE OPTIONS:░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░░ ║
║ ░░Unused I/O pins configured as outputs? Y ... if "Y", Level = H ║
║                                                            ║
║ FITTING OPTIONS:                                           ║
║    When compiling              Run until 1st success: EXTRA║
╚════════════════════════════════════════════════════════════╝
```

Figure 10. MACH Fitting Options - Default Fuse Options.

Note: PALASM 4 version 1.4 allowed the user to configure unused I/O pins as inputs, but did not allow the user to control what state they were driven to.

An informational message is produced when DEFAULT FUSE OPTIONS is set to "Y":

```
INFORMATION F065 — Unused pins default to outputs — High
---------------------------------------------------------------------
Explanation:
. Unused pins are being defaulted to be outputs and enabled.
. This diminishes the coupling of external noise into the MACH device
. and removes the need for external resistor pull-up terminations.
```

Note that unused pins associated with buried logic are still not enabled. Any internal logic changes would cause the pin to change, dissipating additional power and "broadcasting" the buried signal for all to see.

3.3.3 Extra Macrocell Assignment Iterations

EXTRA Option

This Fitter feature is enabled from the PALASM 4 compilation menu.

The new *Run until 1st Success: EXTRA* option executes extra macrocell assignment iterations. When the *EXTRA* Fitter option is selected, the Fitter process is altered. The state of the *Maximize packing of logic blocks* flag, which controls block partitioning, is determined before macrocell assignment occurs.

```
══════════════════ PALASM4 version 1.5 ══════════════════

 FILE    EDIT    RUN    VIEW    DOWNLOAD    DOCUMENTATION    <F1> for Help
┌──────────────────────────────────┐
│ Begin new design                 │
│ Retrieve existing design         │════ MACH FITTING OPTIONS ════════════
│ Merg┌───────────────────────────────────────────────────────────────┐
│ Chan│  OUTPUT:                                                        │
│ Delet│     Report level                       Brief                   │
│ Set u│  SIGNAL PLACEMENT:                                             │
│      │     Float all signals and ignore grouping?    N               │
│  Wd  │     Use placement data from            Design file            │
│  Co  │     Save last successful containing ┌──────────────────────────┤
│  Sir │     Press <F9> to edit file containing│ Use all fitting options │
│  Lo  │                                      │ Run until 1st success: STD │
│      │  DEFAULT FUSE OPTIONS:                │ Run until 1st success: EXTRA │
│      │     Unused I/O pins configured as ou  │ Select one combination . . . │
│      │  FITTING OPTIONS:                     └──────────────────────────┤
│      │     When compiling            Run until 1st success:   STD      │
│      └───────────────────────────────────────────────────────────────┘
└──────────────────────────────────────────────────────────────────────┐
 >  move,<F2> choice,<F3> save,<F9> edit,<F10> form ok, <Esc> abort
```
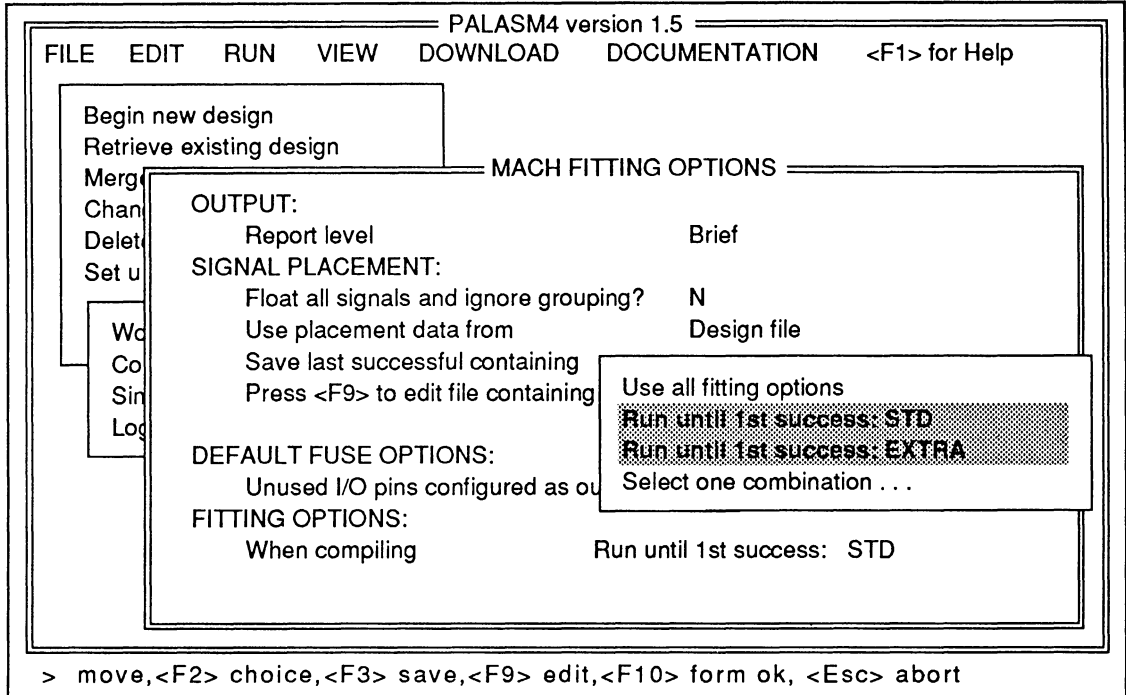
Figure 11. Fitting Options Menu.

If during the fitting process the design is fitted or an error occurs which causes the Fitter to abort, the process is terminated. A message is included in the final error count of the .RPT file which indicates whether the Fitter has aborted:

"%% FITR %% Program Aborted - Errors: 1, Warnings: 0"

If the Fitter does not abort, the process works as follows:

Initially the *Maximize packing of logic blocks* flag is OFF. If the design is successfully partitioned, the macrocell assignment iterations in Table 7 are run. If it fails partitioning, the *Maximize packing of logic blocks*

flag is turned ON and partitioning is re-run. If the design is then successfully partitioned, the macrocell assignment iterations in Table 7 are run. If not, the design has failed to fit.

If the design has failed to fit with the *EXTRA* iterations, the *Run until 1st success: STD* Fitter iterations (see *STD* option) are then executed.

The macrocell assignment iterations in Table 7 include four *STD* Fitter flag combinations and four *EXTRA* iterations. The extra iterations attempt placements which cannot be duplicated using the *Run until first success: STD* or *Select one combination* menu options. Therefore, it may be necessary to back-annotate pins in order to duplicate *Run until 1st success: EXTRA* Fitter runs using these menu options.

Expand small PT spacing	Expand all PT spacing	Comments
N	N	same as *STD*
Y	N	same as *STD*
N	Y	same as *STD*
Y	Y	same as *STD*
N	N	1st *EXTRA* Iteration
N	N	2nd *EXTRA* Iteration
N	N	3rd *EXTRA* Iteration
N	N	4th *EXTRA* Iteration

Table 7. *EXTRA* option fitting flags.

The benefit of the *EXTRA* Fitter option is that additional macrocell assignments are attempted for a given block partition, allowing designs to fit that otherwise would not.

The drawbacks of the option are that compilation times are longer and a larger .RPT file is generated. It is larger because it includes all Fitter iterations (see section 3.4.2).

STD Option

The *Run until 1st success: STD* Fitter option operates the same as the PALASM 4 Version 1.2 *Run until first success* option. It does not execute extra macrocell assignment iterations. When the *STD* option is selected, the Fitter executes up to eight iterations, each using a different combination of the three Fitter flags: *Expand small PT spacing*, *Expand all PT spacing* and *Maximize packing of logic blocks*. Fewer than eight iterations will be run if the design is fitted or an error occurs which causes the Fitter to abort.

The *STD* option creates a .RPT file reflecting the last Fitter iteration.

☞ 3.4 Changes to the Fitter Report File

Several changes were made to the Fitter report (.RPT) file for PALASM 4 version 1.4 and version 1.5:

- Report file changes for PALASM 4 version 1.5
  - "Run Until 1st Success - EXTRA" Intermediate Error Counts (section 3.4.1)
  - "Total Block Fanout" (section 3.4.2)
- Report file changes for PALASM 4 version 1.4
  - The Run Until 1st Success: EXTRA fitting option lists multiple .RPT files (section 3.4.3)
  - Block Partitioning Results table (section 3.4.4)
  - Block Signal List Table (section 3.4.5)
  - Block Input List (section 3.4.6)
  - Using the Block Input List and the Block Signal List Table to Manually Partition a Design (section 3.4.7)

If you are already familiar with PALASM 4 version 1.4, just read sections 3.4.1 and 3.4.2 before moving on to Chapter 4.

☞ 3.4.1 Report File Changes - "Run Until 1st Success" Intermediate Error Counts

The wording of the informational message about "partial" error counts given by the Run Until 1st Success: EXTRA fitting option has been changed to "intermediate" error counts:

```
|> INFORMATION F085 - Intermediate Error Count: %1, Warning Count: %2
-------------------------------------------------------------------
Explanation:
. The Fitter has calculated an intermediate total of all error(s) and
. warning(s) detected to this point in the processing sequence. It will
. try different placement and assignment options automatically for the
. user. The error count is reset to zero before beginning this additional
. processing.
```

☞ 3.4.2 Calculation of Total Block Fanout

A complete description on how the total incremental current due to switching inputs (i_{IT}) is calculated is presented in the General Information section of the MACH Family Data Book (revision F or later), under "Approximiating Actual Application Supply Current".

The AMD MACH Fitter reports the fanout for each individual input in the "Blocks" column on the far right side of the "Signals - Tabular Information" portion of the MACH

report file. The total block fanout - the sum of all the fanouts of the individual inputs - is listed just above that table as part of an informational message from the Fitter. For example:

```
INFORMATION FO58 - Total Block Fanout (all Signals) *: 40
```

3.4.3 Run Until 1st Success: EXTRA Fitting Option Lists Multiple .RPT Files

A single .RPT file is created during compilation if the Run Until 1st Success: STD fitting option is chosen. However, when the Run Until 1st Success: EXTRA option is chosen, the Fitter combines the .RPT files from all fitting attempts in one large .RPT file. This provides the user with more fitting information, but uses additional disk space.

Two techniques for minimizing the size of the .RPT file are -

- Select the Brief report level when using the Run Until 1st Success: EXTRA fitting option.
- Use the Run Until 1st Success: STD fitting option.

3.4.4 Block Partitioning Results Table

Two changes have been made to the Block Partitioning Results table. One change is that "Product terms" are listed in the second column and "Macros Remain" are listed in the last column of the table (see Table 8).

The other change is that asterisks are placed next to numbers reflecting high resource utilization. If the *Maximize packing of logic blocks* Fitter option is ON, asterisks are assigned to resource numbers reflecting 100% utilization. If the *Maximize packing of logic blocks* option is OFF, however, asterisks may appear when utilizations are lower. See Table 7 in Reduced Block Limits, section 3.3.3, for block limits when the *Maximize packing of logic blocks* option is OFF.

```
*** Block Partitioning Results

               Array     Product    # I/O    Buried    Signal    Macros
               Inputs    Terms      Macro    Logic     Fanout    Remain
     Block-> A   26*       48         8*        4         9         4
....
     Block-> H    7        64*        7         0         9         9
```

Table 8. Block Partitioning Results.

3.4.5 Block Signals List Table

There are two additions to the Block Signal List (shown in Table 9): 1) a number following each equation which represents the number of array inputs driving it, and 2) an asterisk, appearing only in MACH230 designs, which flags equations that must be placed in I/O macrocells.

If an equation drives only one array input, it is likely to be a good choice for moving to another block during the manual partitioning process.

The asterisks are used to determine which buried MACH230 equations have been placed in I/O macrocells. When manually partitioning a MACH230 design, users should scan this list for buried equations with asterisks, and re-partition logic so that these equations drive sibling blocks only. If the buried equations drive only sibling blocks, on subsequent attempts to fit the design the Fitter will put them in buried macrocells.

```
*** Block Signal List

Blk-> A        ONESHOT  3       WRITE  1        DS  1       S2*  4
                    S1* 3       DATHI* 4      L_DS* 3   VLDATA*  5
                DSACKO* 7       SIZ1*  2    DSACK1*  7     SIZO*  2
```

Table 9. Block Signal List.

3.4.6 Block Input List

A new section, Block Input List, has been added to the .RPT file (see Table 10). The "Blk" segment lists the array inputs and number of equations using each array input in a block.

Below the *Blk* segment is the *Singular* segment, which lists each array input in the *Blk* segment that drives only one equation and, in brackets to the right of each array input, the equation it drives. Equations listed in the singular list are likely to be good choices when moving logic between blocks during manual partitioning.

```
*** Block Input List

Blk-> A         SIZO  2       AWRITE  1       VLDATA  1        XRDY  2
              LBGACK  7         SIZ1  2         L_DS  1      LWRITE  1
             SCSISEL  2          LDS  2          UDS  2       DTACK  2
                 SA1  1          AAS  3       EXTERN  1    AMIGAMEM  3
                  SO  2           S1  1           S2  0       DSACK  2
             ONESHOT  1

Singular:       AWRITE{         VLDATA}       VLDATA{       ONESHOT}
                  L_DS{             DS}       LWRITE{         WRITE}
                   SA1{          DATHI}       EXTERN{         DATHI}
                    S1{             S2}      ONESHOT{        VLDATA}
```

Table 10. Block Input List.

3.4.7 Using the Block Input and the Block Signal Lists to Manually Partition a Design

The Singular list in Table 10 identifies WRITE and DS as the only equations driven by the signals LWRITE and L_DS. From the Block Signal List we see that these equations use only one array input each (LWRITE and L_DS); therefore they are good choices for moving to another block if we wish to reduce the array input or product term cluster utilization in block A.

CHAPTER 4. MACH230 DEVICE

4.1 MACH230 General Notes

The MACH230 switch matrix has a buried macrocell interconnect limitation not present in other MACH devices. This limitation has made necessary the addition of new Fitter features discussed in Section 4.2. Specifically, the user will find that -

- MACH230 designs are more likely to require manual partitioning than other MACH designs.
- The manual partitioning process is more complex.
- The changed Fitter algorithm causes partitioning commands in MACH230 designs to operate differently than partitioning commands in other MACH designs.

In this section -

- Section 4.1.1 discusses the buried macrocell interconnect scheme.
- Section 4.1.2 describes the differences between manually partitioning a MACH230 design and manually partitioning other MACH designs.
- Section 4.1.3 covers issues concerning manually partitioning MACH230 input registers and latches.

4.1.1 MACH230 Buried Macrocells Drive Only Sibling Blocks

I/O macrocells in the MACH230 device, and all macrocells in every other MACH device, can drive any block in the device. In contrast, buried macrocells in the MACH230 device (macrocells with odd numbers) can only drive macrocells in their "sibling" blocks. Each MACH230 buried macrocell has two sibling blocks, its own and one other. Sibling blocks are listed in Table 11.

```
Buried macrocells                Can only drive macrocells
in MACH230 blocks                in "sibling" blocks

            A                                A   H
            B                                B   G
            C                                C   F
            D                                D   E
            E                                E   D
            F                                F   C
            G                                G   B
            H                                H   A
```

Table 11. Sibling Blocks of MACH230 Buried Macrocells.

4.1.2 Manually Partitioning a MACH230 Design

The two differences between manually partitioning a MACH230 design and partitioning other MACH designs are that the process is more complex and the manual partitioning controls operate differently.

More Complex Manual Partitioning Process

While other MACH designs may be partitioned as described in the "Manual Logic Partitioning" MACH Technical Brief, an additional factor must be considered for MACH230 designs. Since MACH230 I/O macrocells can drive all blocks in the device, but buried macrocells can drive only sibling blocks, I/O macrocells are a more valuable resource than buried macrocells. Therefore, logic should be partitioned so that as many buried signals as possible drive only sibling blocks. If buried signals are partitioned so that they drive only sibling blocks, they will be placed in buried macrocells. Otherwise they will be placed in I/O macrocells.

MACH230 Grouping Controls

The MACH230 manual partitioning command GROUP_MACH_SEG_X is interpreted differently from GROUP_MACH_SEG_X commands for other MACH devices because the Fitter contains a MACH230 Block Relettering feature which "swaps" block letters within the device. (Refer to section 4.2.3 for an explanation of Block Relettering.)

MACH230 designs may be manually partitioned with or without the Block Relettering feature enabled (see Figure 3). It is recommended that Block Relettering be disabled so that the user retains maximum control. To disable relettering, specify the pin or node location of at least one signal in the declaration section of the .PDS file. If possible, the user should place a clock signal because it will deactivate Block Relettering without affecting fitting. Note in Figure 12 that a clock signal (CLK) is pre-placed in the .PDS file; consequently, the equations in the .RPT file are placed in block E as specified by the GROUP_MACH_SEG_E command in the .PDS file.
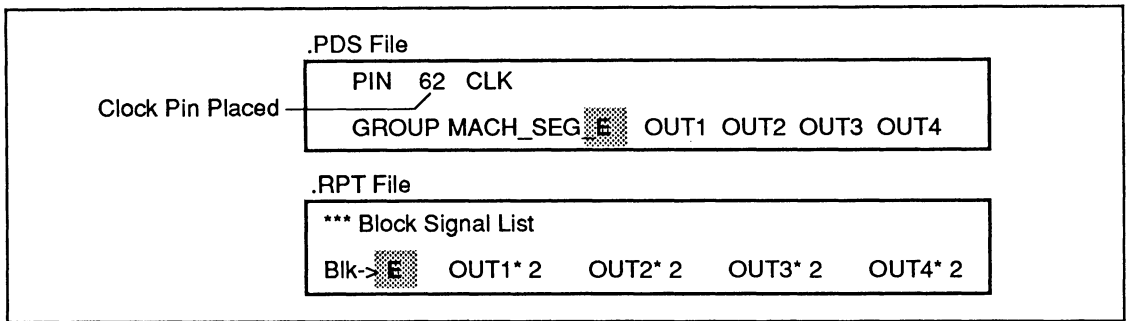
```
                    .PDS File
                    ┌─────────────────────────────────────────────────┐
                    │   PIN   62  CLK                                  │
Clock Pin Placed ───┤          /                                      │
                    │   GROUP MACH_SEG ▓E▓  OUT1 OUT2 OUT3 OUT4        │
                    └─────────────────────────────────────────────────┘
                    .RPT File
                    ┌─────────────────────────────────────────────────┐
                    │  *** Block Signal List                          │
                    │                                                 │
                    │  Blk->▓E▓   OUT1* 2    OUT2* 2    OUT3* 2   OUT4* 2 │
                    └─────────────────────────────────────────────────┘
```

Figure 12. MACH230 Block Partitioning Results - Location Specified.

If no signal locations are declared, the relettering capability is enabled, and the Fitter is free to swap block letters. Note in Figure 13 that no signals are pre-placed in the .PDS file. Subsequently, the equations associated with the GROUP_MACH_SEG_E command are placed in block F, not E, as shown in the .RPT file.
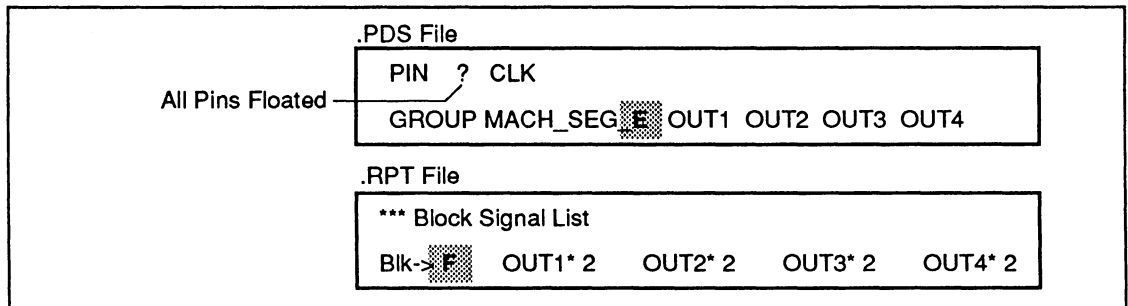
```
                    .PDS File
                    ┌─────────────────────────────────────────────────┐
                    │   PIN   ?  CLK                                   │
All Pins Floated ───┤          /                                      │
                    │   GROUP MACH_SEG ▓E▓ OUT1 OUT2 OUT3 OUT4         │
                    └─────────────────────────────────────────────────┘
                    .RPT File
                    ┌─────────────────────────────────────────────────┐
                    │  *** Block Signal List                          │
                    │                                                 │
                    │  Blk->▓F▓   OUT1* 2    OUT2* 2    OUT3* 2   OUT4* 2 │
                    └─────────────────────────────────────────────────┘
```

Figure 13. MACH230 Block Partitioning Results - No Locations Specified.

4.1.3 MACH230 Input Registers and Latches

Since input registers and latches use buried macrocells, they are capable of driving only sibling blocks. Therefore, it may not be feasible to use an input register or latch for an equation; the feasibility depends on how much logic it drives and where the logic is placed.

Input registers and latches that drive macrocells in non-sibling blocks must be placed in I/O macrocells. The Fitter does this automatically for signals driving non-sibling blocks and issues the following warning:

┌───┐
│   "Warning F177 - Pair Declaration Ignored: ..." │
└───┘

Users should realize that placing an input register or latch in an I/O macrocell increases the setup time of the register/latch and consumes an extra array input and product term cluster.

To force the Fitter to place the input register/latch in a buried macrocell, the design must be manually partitioned as described above in section 4.1.2 so that the input register/latch drives only sibling blocks.

4.2 MACH230 Fitter Features

Several new Fitter features were added with PALASM 4 version 1.4. Four of these features, which apply only to MACH230 designs, are described here (see Figure 14). The other four are described in Section 3.2 of this document.
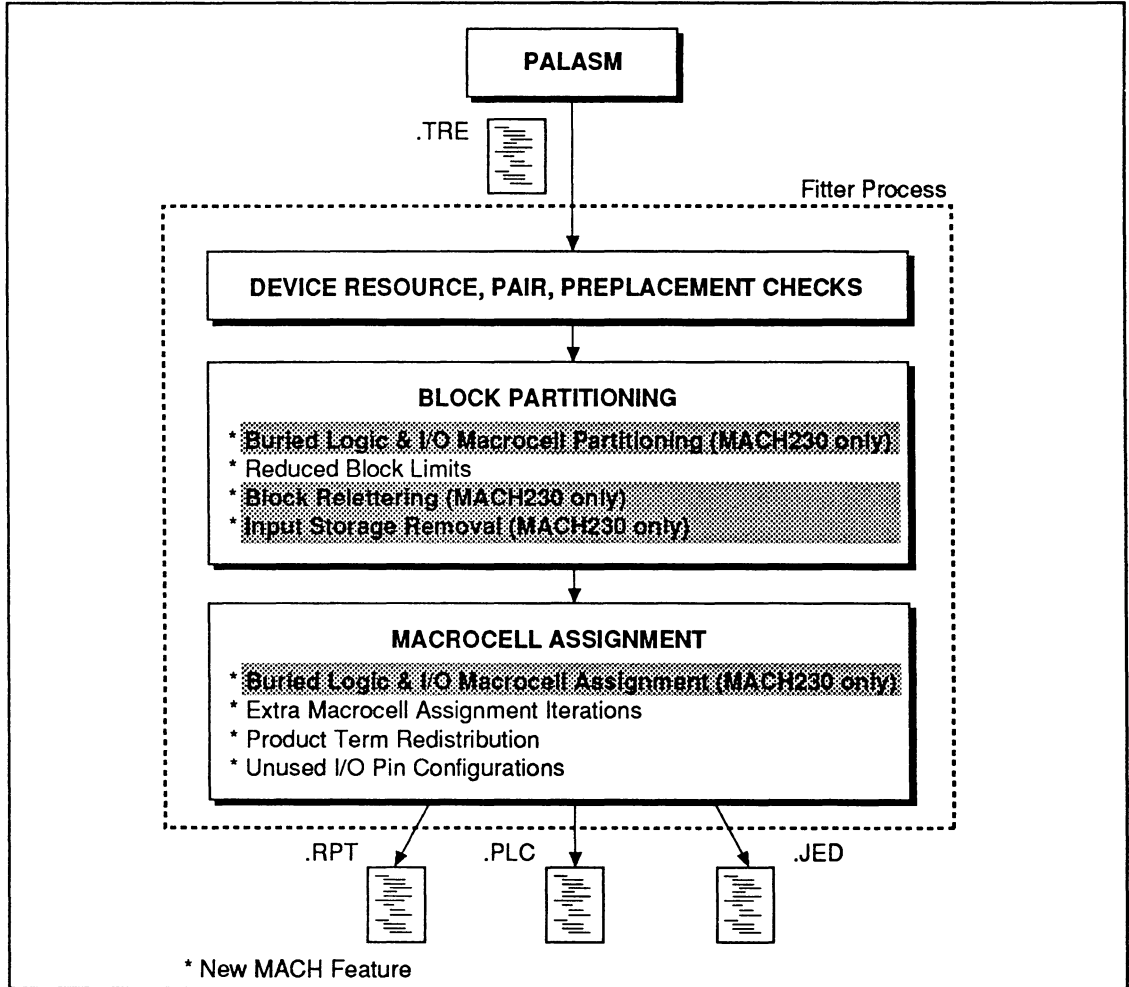

Figure 14. Flowchart of the Fitter process with MACH230 features highlighted.

4.2.1 Buried Logic and I/O Macrocell Partitioning

Buried Logic and I/O Macrocell Partitioning has been implemented so that partitioning is consistent with the placement changes that will be made later in the fitting process during Buried Logic and I/O Macrocell Assignment (see section 4.2.2).

When partitioning logic into blocks, the Fitter counts the equations each buried signal drives. If a buried signal drives many equations, the Fitter assumes the buried signal will drive non-sibling blocks and will be placed in an I/O macrocell. Thus, during block partitioning, the Fitter counts the buried signal as an I/O signal.

The benefit of this feature is that if the resulting partition does not cause the design to exceed block-resource limits, it increases the likelihood that logic can be successfully assigned to macrocells in a block.

The drawback to this feature is that, in some instances when a feasible partition exists for a design, it will cause fitting failures during the block partitioning phase.

To disable Buried Logic and I/O Macrocell Partitioning, manually partition the design.

4.2.2 Buried Logic and I/O Macrocell Assignment

Buried Logic and I/O Macrocell Assignment occurs after a design is partitioned. If a buried equation drives non-sibling blocks, the Fitter places the equation in an I/O macrocell.

The benefit of Buried Logic and I/O Macrocell Assignment is that it causes designs which contain buried equations driving non-sibling blocks to fit.

The drawback is that I/O macrocells are a more critical MACH230 device resource than buried macrocells, and the Fitter may use additional I/O macrocells for buried equations when simply re-partitioning logic into sibling blocks will keep them in buried macrocells.

To keep buried logic from being placed in I/O macrocells, manually partition the logic so that the buried signal drives only sibling blocks.

4.2.3 Block Relettering

Block relettering occurs after block partitioning. If no signal locations are specified in the declaration section of the .PDS design file, the Fitter automatically reletters blocks so that buried logic drives sibling blocks.

Block relettering helps fit designs by moving buried signals and the equations they drive into sibling blocks.

There are two drawbacks to this feature: 1) It causes MACH230 grouping commands to operate differently than

grouping commands for other MACH devices, as described in section 4.1.2, and 2) .RPT file messages listed above the relettering warning message "Warning F176 - MACH 230 Block Reletter" may refer to the wrong blocks.

To disable Block Relettering, pre-place one or more signals in the design. To avoid mixing fixed and floating signals in a design, pre-place a clock signal.

4.2.4 Input Storage Removal

Input Storage Removal occurs after the logic is partitioned into blocks. If an input register/latch drives non-sibling blocks, the input register/latch is configured as a standard register/latch and placed in an I/O macrocell.

The benefit of Input Storage Removal is that it causes designs to fit that otherwise would not.

The drawback of Input Storage Removal is that moving logic from input registers/latches to standard registers/ latches causes additional resources to be used and increases the storage element's setup times. See section 4.1.3 for a detailed explanation of the ramifications of Input Storage Removal.

To keep logic in input registers/latches, manually partition the logic so that it drives only sibling blocks.

☞ **What to read In Chapter 5**

☞ **5.1 MACH215 Silicon Information**

If you are using <u>MACH215</u> devices and are a ....

Previous PALASM 4 v1.4 user	*read all of chapter 5*
New user	*read all of chapter 5*

The MACH215 is the first asynchronous MACH device. A complete description of the device can be found in the new MACH Device Databook, (PID #14051 revision F or later) available from your local AMD sales representative or AMD Literature (800) 222-9323.

The MACH215 Node Numbers and Cell Names appear on the following page. Pin 13 is the default clock.
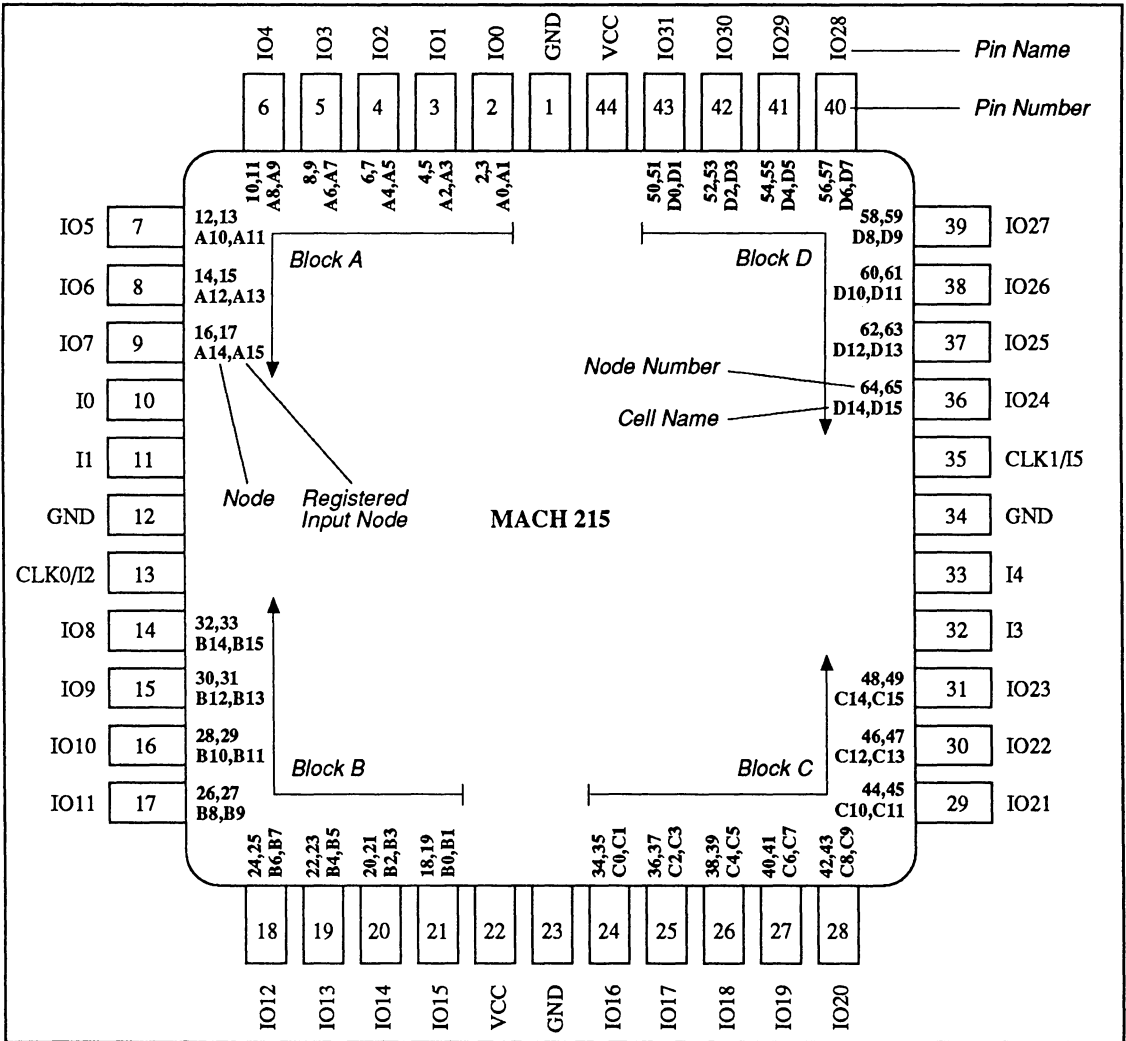
Figure 15. Pin and Node Numbering for the MACH215.

☞ 5.2 Control of Special MACH215 Features

☞ 5.3 MACH215 Input Macrocell Behavior

The individual output enable and flip-flop controls for clocking, preset and reset within the MACH215 are controlled by standard PALASM 4 software syntax.

PALASM 4 version 1.5 attempts to map the user's design to the physical resources of the device. Two restrictions in the input macrocells of the MACH215 affect software operation, particularly the Fitter:

- The input storage element, present for each I/O pin, has independent control of the clocking functions. However, they can only be gener-

ated by the two dedicated clock pins (designated as "Clk/Inputs"), not the internal clock product term functions.

- The input storage element is only initialized by power-on reset. It does not have preset or reset.

☞ 5.4 MACH215 Fitter Issues

The MACH215 device has special features which affect the operation of the MACH Fitter. Specifically:

- PAIRing in the MACH215 can restrict the Fitter's ability to place and route a design (section 5.4.1).
- Input setup times can be adversely affected by clock functions implemented through clock product term resources (section 5.4.2).
- SET, RESET and CLK product term resources are not supported within input PAIR statements (section 5.4.3).
- The fitting algorithm for assigning clocks has been changed (section 5.4.4).

5.4.1 PAIRing In the MACH215 Can Restrict the Fitter

PAIRing directs the Fitter to treat paired signals as one structure for placement purposes. Output pairs are implemented in the same macrocell, while input pairs are implemented in adjacent macrocells.

The presence of input pairs can restrict the Fitter's ability to place and route the design. Removing input PAIRs and using clock product term resources instead for "extra" clock inputs can improve the Fitter's ability to realize the design.

5.4.2 Input Setup Times Affected by Clock Functions Implemented Through Clock Product Term Resources

MACH215 input setup times can be adversely affected by clock functions implemented through a clock product term resource. For example:

```
                 XYZ.CLKF = /ABC
```

If pin ABC is placed on a dedicated clock/input pin, it becomes a high-speed direct connection, and both polarities of the connection are available. However, if more than one clock source is involved, or if other signals are already placed on the dedicated clock/input pins, this CLKF function must be implemented as a clock product term resource, at a slight performance penalty. If this occurs, input setup times may be adversely affected. A warning message will appear:

```
|> WARNING F138 - Clock signal(s) connected by PT logic: Sig_Name
-----------------------------------------------------------------
Explanation:
  The Fitter software has used the PT clock logic to connect the CLKF to
this storage element. This was done because the direct clock pin
connection was already used for another clock source. PT clocks are
slower than dedicated pin connections. If this is acceptable, no action
is required.
```

As long as this is acceptable, no action is required. A correct JEDEC file is produced.

5.4.3 SET, RESET and CLK Product Term Resources Not Supported Within Input PAIR Statements

The MACH215 does not support SET, RESET or CLK product term resources within an input PAIR statements. If it encounters an illegal PAIR declaration, the Fitter places the logic in separate macrocells and displays the warning message below:

```
|> WARNING F179 -Pair Declaration Ignored: 215 PAIR S/R found- Sig_Name
------------------------------------------------------------------
Explanation:
 The PAIR declarations for the indicated signals are slightly contrary
 to the supported device features. The design will be completed
 ignoring the user provided PAIR declaration in independent logical
 resources chosen by the Fitter. (Both Pin and Node are floated.)
 (Input registers, realized in the array will have longer setup times.)
 If fitting subsequently fails, check the construction of this logical
 PAIRing for accuracy. Correction of the PAIRing problem may allow
 more efficient realization as a single resource.

 Presence of S/R PT's on 215 input PAIRs is incompatible w/ macrocell.
 Presence of Clk PT's on 215 input PAIRs is incompatible w/ macrocell.
 An excess of single literal clocks (>2) must be realized as PT clocks.
*  .
 Note: PT clocks are also slower than dedicated pin connections.
```

The same warning message is displayed when the total number of declared clock sources exceeds the clock resources available.

5.4.4 Changes to the Fitting Algorithm for Assigning Clocks

The fitting algorithm for assigning clocks has been changed to accommodate the MACH215:

- First, the Fitter analyzes any pre-placed pins and their usage. If any of the pre-placed pins are for clock sources, or involve dedicated clock/input pins, the appropriate resources are set aside during automatic assignment.
- Next, the Fitter attempts to make the most use of the remaining resources and clock requirements for realizing single pin name connections from the right-hand side of the CLKF equation.
- All floating clocks driving logic macrocells (0, 2, 4, ...14) are counted. The most frequently used source is assigned the default clock, pin 13, if available. The other remaining clocks are implemented using clock product term resources. These clocks are noted with

warning message F138 "Clock signal(s) con-
nected by PT logic".

- Finally, floating clocks driving input macrocells (1,
3, 5, ..15) are counted. Up to two of the most
frequently used signals are assigned to dedicated
clock/input pins. Signal connections for other
cells are implemented by ignoring PAIR declara-
tions, resulting in these signals being placed in
separate macrocells. These signals are noted
with warning message F179 "Pair Declaration
Ignored: 215 PAIR Clk excess".

5.5 MACH215 Design Example - Flag Register

This simple flag register example that can be loaded and read
from a single three-state bus performs two logic fuctions:
monitoring sequential differences; and merging several stored
conditions into a single device.

The description that follows focuses on the the description of
the logic functions and the resulting report files produced by
the MACH Fitter. The full text of the .PDS file is presented in
Appendix 3.

Part 1) Sequential Differences

A stored status bit reflects the presence of a difference
between the current value of a three-state bus (pin P1 in figure
16) and the value previously stored in latch B2 at the time it
was clocked. The status flag is computed by XORing the
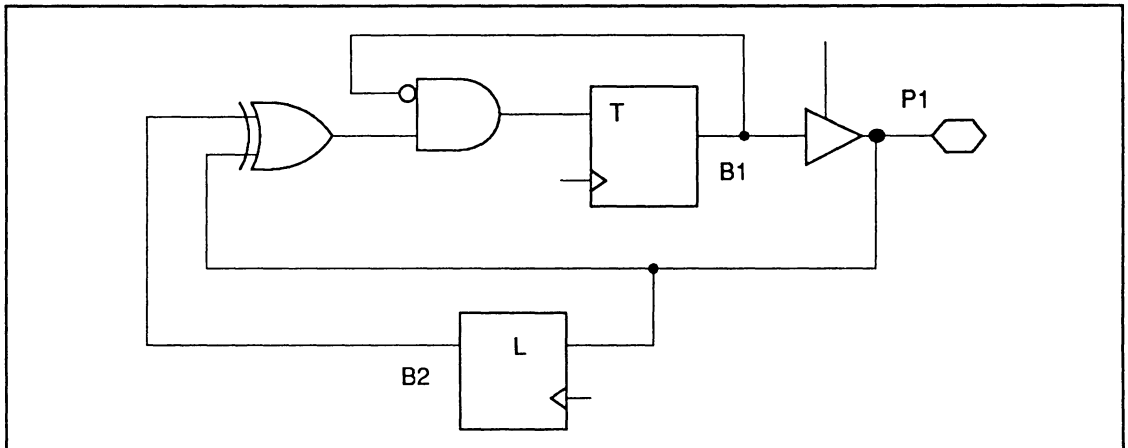current and the previous bus values, and is stored to be read
on that bus.

Figure 16. Flag Register Status Bit

The PALASM 4 syntax below declares the pin signal name and the two node names associated with the stored status bit.

```
PIN   ? P1 IPAIR B2
NODE ? B1 OPAIR P1
NODE ? B2

EQUATIONS

P1.T := /B1 * (B2 :+: P1)
B1.T := {P1.T}           ; Node equation identical to pin equation
B2 *= P1                 ; Transfer function input latch
X = B2                   ; Display output for testing
```

The three signal names (P1, B1 and B2) uniquely specify the desired feedback connection. A special connection equation must be provided to confirm the PAIR configuration. The equation B1.T, identical to the pin equation, insures that the PALASM simulator develops and stores the correct value while remaining device independent. The separate latch equation for node B2 plays a similar role.

When signals are declared without PAIR declarations, the Fitter makes connections using signal paths found in simple PAL devices, according to the type of logic. Registered signals are connected using internally developed register output; combinatorial signals are connected using pin connections. Buried signals are developed using internal connections, leaving the pin free for use as an input.

To complete the sequential differences portion of the design, auxiliary control equations must be added:

```
P1.TRST = ENA
P1.RSTF = R
B1.RSTF = R

P1.CLKF = CLK1          ; Must minimize to 1 PT
B1.CLKF = {P1.CLKF}     ; Must be same as CLK for pin function
B2.CLKF = CLK2
```

The MACH215 only has one product term per clock; therefore the clocking function must minimize to one PT. Note that the same equation is required for both P1 and B1.

Part 2) Merging Stored Conditions

The single status bit reflects the OR of several stored conditions, as shown in Figure 17.
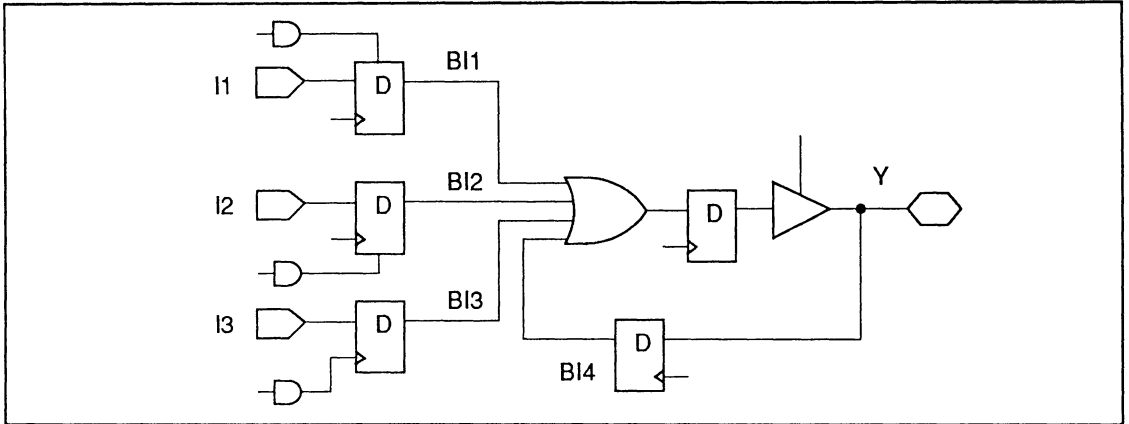

Figure 17. Multiple Stored Conditions

The other stored condition is defined as a registered input cell:

```
PIN  ? I1 IPAIR BI1    NODE ? BI1
PIN  ? I2 IPAIR BI2    NODE ? BI2
PIN  ? I3 IPAIR BI3    NODE ? BI3
PIN  ? Y  IPAIR BI4    NODE ? BI4

EQUATIONS

Y:= BI1 + BI2 + BI3 + BI4
Y.TRST = ENA
Y.CLKF = /CLK1

BI1 := I1   BI1.CLKF = /CLK2     BI1.SETF=P  ; MACH215 macrocell
BI2 := I2  /BI2.CLKF = /CLK2     BI2.RSTF=R  ; Set/Reset on IPAIR
BI3 := I3   BI3.CLKF = /CLK1 + /CLK2         ; CLK PT use
BI4 :=  Y  /BI4.CLKF =  CLKS                 ; Excess CLK resources
```

Part 3) Output from the Fitter

Several warnings appear when the design is compiled, illustrating MACH215 device-specific architectural issues:

```
PAIR Analysis...

|> WARNING F179 - Pair Declaration Ignored: 215 PAIR S/R found - I1/BI1
|> WARNING F179 - Pair Declaration Ignored: 215 PAIR S/R found - I2/BI2
|> WARNING F179 - Pair Declaration Ignored: 215 PAIR Clk PT found - I3/BI3

Pre-Placement and Equation Usage Checks...

|> WARNING F138 - Clock signal(s) connected by PT logic: Y/BI4
```

These warnings are normal. Be sure to evaluate the set-up times of each of the signals.

Part 4) Resource Displays from .RPT file

The Device Resources Checks table counts input registers within the listed Total Macro figure.

```
*** Device Resource Checks

                  Available      Used       Remaining
       Clocks:        2            2             0
         Pins:       38           12            26      ->     31%
    I/O Macro:       32            3            29
  Total Macro:       64            8            56
Product Terms:      128           14            96      ->     24%

MACH-PLD Resource Checks OK!
...
|> INFORMATION F050 - Device Utilization.............. *: 21 %
```

The Block Signal List (below) has an entry for equations with zero signal inputs (B2). These are registered input signals realized in dedicated input cells.

```
*** Block Signal List

Blk-> A       BI4  1        Y   5      BI3  1      BI2  1
              BI1  1        X   1      B2   0      B1   5
              P1   6
```

The Logic Map and accompanying key (below) show that MACH215 cells 16-21 are not available for use; these are array input cells. Alternate cells (1, 3, 5,..) in the MACH215 device can only be used as input registers; a zero indicates when one is used. No array PTs are used in this design.

```
*** Logic Map - Demo 215 Features Test

   Gbl Inp .--.      I/O  .--+--A--+--.   I/O     I/O  .--+--B--+--.   I/O
      CLKS| 0|             | 0| *  #|21|                | 0| .  #|21|
         R| 1|             | 1| -  #|20|                | 1| -  #|20|
          | 2|          P1 | 2| 8  #|19|                | 2| .  #|19|
         P| 3|          B2 | 3|-0  #|18|                | 3| -  #|18|
       ENA| 4|          Y  | 4| 1  #|17|                | 4| .  #|17|
          | 5|             | 5| -  #|16|                | 5| -  #|16|
          '--'          X  | 6| 1  -|15|                | 6| .  -|15|
   K:  * Used              | 7| -  1|14| BI1            | 7| -  .|14|
       . Avail          BI4| 8| 1  -|13|                | 8| .  -|13|
       # N/A               | 9| -  1|12| BI2            | 9| -  .|12|
       0 RegIn          BI3|10| 1  -|11|                |10| .  -|11|
       . Avail              '--+--u--u+--'               '--+--u--u+--'
       - No.PT
```

CHAPTER 6 - MACH220 DEVICE

☞ **What to read in Chapter 6**

If you are using <u>MACH220</u> devices and are a ....

Previous PALASM 4 v1.4 user   *read chapter 6*
New user   *read chapter 6*

☞ **6.1 MACH220 Silicon Information**

A complete description of the MACH220 device can be found in the new MACH Device Databook (PID#14051, revision E - 4Q 1991 or later), available from your local AMD sales representative or AMD Literature.

The MACH220 node numbers and cell names appear in Chapter 11 of the PALASM 4 User's Manual, page 11-190.

☞ **6.2 Documentation Error in PALASM 4 User's Manual**

The table on page 11-167 of the PALASM 4 User's Manual incorrectly identifies pin 17 as CLK1 for the MACH120 and 220 devices. It should be pin 16.

APPENDIX 1 - PALASM 4 VERSION 1.5 FITTER MESSAGES

This appendix lists and explains some of the messages which the PALASM 4 version 1.5 Fitter may report. Other messages are described in chapter 5 of the PALASM 4 User's Manual. Also, the PALASM 4 .LOG file and on-line help provide explanations of many errors not described in these documents.

"F065 - Unused pins default to outputs"

A Fitter option has been selected which configures all unused I/O pins as outputs. This eliminates the need for external pull up or pull down resistors on unused I/O pins. If this message is not displayed, all unused I/O pins are configured as inputs and should be pulled up or down.

"F130 - Extra pin or node declarations"

This warning applies to signals which are declared in the design file, but are not used in any equations. The Fitter ignores them. Beware! The unused signals must be removed from the declarations section of the design file for a .JDC file to be created after simulation.

"F170 - No Set/Reset initialization function found! - <n> missing"

This warning indicates that preset and reset terms are not specified in the design file for a registered signal. If the message reports two missing, then at least one signal is missing both reset and preset terms. If the message reports one missing, one or more signals are missing either reset or preset terms.

Beware! The signal's reset/preset terms will be defaulted to a "don't care" state, not "off". If the signal then gets placed in a block with other registered signals, its reset/preset states will become those of the other signals in the block since all signals in a block have the same reset/preset states.

When this warning message occurs, users should check the reset/preset states assumed by signals listed in the message. If the states which they assume impede functionality, their reset/preset states should be defined appropriately in the design file.

"F176 - MACH230 Block Reletter"

This warning applies to MACH230 designs containing GROUP_MACH_SEG_X statements which manually partition logic into blocks. It indicates that the Fitter has swapped block letters (e.g., the logic partitioned into block B is placed in block G, and the logic partitioned into block G is placed in block B).

To disable Block Relettering, pre-place a signal in the design. Clock pins are generally good choices for this pre-placement.

"F177 - Pair Declaration Ignored"

This warning applies to input register/latch (input paired) signals in MACH230 designs. It indicates that input register/latch signals drive non-sibling blocks (see section 4.1.3). If they do, the Fitter unpairs the signals and configures them as standard registers/latches. Note that standard registers/latches have longer setup times than input registers/latches.

To force the Fitter to use the input register/latch configuration, manually block partition the logic so that the input registers/latches drive only sibling blocks.

"F178 - Pair Declaration Ignored"

The Fitter issues this warning message when it ignores illegal PAIR requests in a design file. Pairing syntax specifies that a buried equation be placed in the same macrocell as a specified I/O pin (output pairing), or in the adjacent macrocell (input pairing).

A typical F178 warning message appears below (note the highlighted warning condition) followed by a list of F178 warning conditions.

> "Warning F178 — Pair Declaration Ignored: Transfer Eq. Missing — <signal name/signal name>"

F178 WARNING CONDITIONS

Warning Condition   *Description*

Multiple PAIRing

The design file contains any of the following syntax errors: 1) a pin is paired with more than one buried equation, 2) a buried equation is paired with more than one pin, or 3) a buried equation is input **and** output paired.

No MACH1XX Reg Inp

An input pair is specified in a MACH1XX design. MACH1XX devices do not support input pairing.

Transfer Eq. Missing

An input pair is specified, but the equation that should be associated with it (e.g., *buried equation := pin equation*) is not included in the equation section of the design file.

Unused node

An input paired equation is not used in any equations. Unused logic such as this is removed by the Fitter.

No PAIRing with CLK

The registered output of an input paired signal is used as a clock. Clocks must be driven by an input pin, not a macrocell.

Physically impossible	A paired buried equation and the associated I/O pin are pre-placed such that pairing is impossible. Paired signals must be located in the same block in identical locations (output pairing) or adjacent locations (input pairing).
Xfer Ops	A buried equation is output paired with an I/O pin that uses a different macrocell configuration (e.g., the buried signal uses a D flip-flop, but the I/O signal uses a T flip-flop). Paired signals must have identical macrocell configurations.
Eqs. not Same	Buried and I/O equations that are "output-paired" must be identical. The equations listed in this message are not.
"F180 - User Pre-placement Ignored"	The Fitter issues this warning message whenever it must ignore statements in a design file that pre-place logic in a block or macrocell. A typical F180 warning message is shown below (note the highlighted warning condition) followed by a list of F180 warning conditions.

```
"Warning F180 - Pre-placement Ignored: Not a clock pin - <signal name>"
```

F180 WARNING CONDITIONS

Warning Condition	*Description*
Block Letter too high	You have used an invalid block letter in a block partitioning or group statement. If, for example, you specify block E in a group statement in a MACH210 design, you will get this warning message. MACH210 devices have only four blocks: A, B, C, and D.
Not a clock pin	This design has a clock signal on an I/O pin. Clock signals must be on dedicated clock (CLK) pins.
Pin is not within Block	A pin pre-placement and a block partitioning statement conflict - the pin pre-placement places the signal in one block, yet the block partitioning (GROUP) statement places the signal in another block.
PAIR Placements	A pair statement and a block partitioning statement conflict - signals which are paired together are placed in different blocks.
"F210 - Design contains both floating and fixed placements"	The Fitter issues this warning when processing a design in which some signals are pre-placed on pins and/or in nodes while others are not. If all signals are pre-placed with the placements generated when the design is compiled (pin and node placements are back-annotated) and the design is re-compiled, it is likely that the logic would not fit.

After fitting the design with partially fixed/partially floating signals, back-annotate all pins and nodes and refit the design. |

"F570 - Invalid signal for MACH architecture"

The Fitter issues this error message when a designer uses reset, preset, clock, or tristate resources improperly. A typical F570 error message appears below (note the highlighted error condition) followed by a list of all F570 error conditions.

```
"ERROR F570 - Invalid Signal for MACH Arch. (Logic on clkf) - <signal name>"
```

F570 WARNING CONDITIONS

Error Condition	*Description*
No TRST on Buried Eq	A tristate is used with a buried equation.
No CLKF on Comb Eq	A clock is used to control a combinatorial equation.
No SETF on Comb Eq	A preset is used to control a combinatorial equation.
No RSTF on Comb Eq	A reset is used to control a combinatorial equation.
No Node1 Logic Eq	An equation has been placed in node1. Node1 is reserved for global set and reset.
No Aux Eq. on Input	A reset, preset, tristate, or clock is used to control an input signal.
Output Always Disabled	A tristate equation is set equal to GND. The pin signal associated with this equation should be declared as an input, the nodal signal should be declared as a buried node, and the tristate equation should be removed.
Invalid Clk Eq	A clock signal is set equal to VCC or GND.
CLKf -> IOM	A clock signal is placed on an I/O pin. A clock signal must be placed on a clock pin.
Logic on Clkf	Equations cannot drive clock inputs. Only input pins can. This design has an equation driving a clock input. If the storage is a latch, the left hand side of the .CLKF equation must be inverted.
clkf Product Terms	Equations cannot drive clock inputs. Only input pins can. This design has an equation driving a clock input.
trst Product Terms	A tristate equation consists of logic requiring more than one product term. One product term is available per tristate.
setf Product Terms	A preset equation consists of logic requiring more than one product term. One product term is available per preset.

rstf Product Terms	A reset equation consists of logic requiring more than one product term. One product term is available per reset.
Inverted Aux Eq	A reset, preset, clock, or tristate equation is inverted. The MACH architecture doesn't support inversion of these resources.
Too many TRST/Bank	More than two tristate equations are used in a bank of eight I/O macrocells. Only two tristates are available per bank.

The problems listed here are an addendum to the problems listed in PALASM 4 on-line help.

General

PALASM 4 is not supported under DR DOS.

PALASM 4 is not supported under any NETWORK environments.

PALASM 4 does not issue an error if there is not enough memory to perform a task. If a process does not run or you are returned back to the MAIN menu unexpectedly, check whether there is enough free memory. PALASM needs 580 KB of available memory, but can operate with less for smaller designs.

The DOS 5.0 standard editor QEDIT is not compatible with PALASM 4. Use another editor such as "ED", the supplied editor, or edit files outside PALASM.

.LOG files larger than 64 KB cannot be VIEWed with the supplied editor, "ED".

Most PALASM sub-programs do not check the available disk space. If a SYSTEM_ERROR message is given and seems to be out of place, check the available disk space and free some if needed.

T input equations are not allowed for MACH devices.

Extended Memory Problems Reported With IBM/AT®s and Compatibles - Designs Not Compiling: Problems have been reported running the extended memory version of PALASM 4 on IBM AT (80286) compatibles. These same PCs are able to run the standard memory version successfully. When the problem occurs, compilation of a design would begin under the extended memory version, only to have the menu reappear in a few seconds with no error message or results.

The problem occurs because the INSTALL program, which calls a "tuning" program, did not successfully adapt the extended memory version of PALASM 4 to your PC. The tuning program is not called if you are using extended memory manager or a machine-specific memory access program.

Work Around: Try "tuning" PALASM 4 manually. After installing this release, type:

```
C:> TUNE_MAN 1
```

The numerical arguments 1, 2 or 3 implement different tuning strategies effective on many PCs. If the same problem in running the extended memory version of PALASM 4 continues, try running the TUNE_MAN program again using the numerical argument 2 or 3 instead. PALASM 4 must be already installed in order to use TUNE_MAN. It does not have to be installed again from floppy disks each time, or re-run each time you reboot or reprocess a design.

Extended Memory Problems Reported With IBM/AT®s and Compatibles - BIOS ROMs and Early IBM/AT Compatibles: Some customers with early IBM/AT compatibles have reported problems with the extended memory version of PALASM 4. The problem has been traced to problems in early versions of the ROM BIOS shipped with IBM/AT compatibles manufactured before 1989. If you have one of these PCs, consider upgrading the BIOS ROMs. This inexpensive investment will solve problems with many software packages, not just PALASM 4.

Pre-processing

PARSE sometimes incorrectly points to a statement that is correct or does not exist, complaining about a syntax error, when the problem is in a statement immediately preceding it.

Vectored signals are not expanded out to individual signals on SET CHECK PRELOAD statements in the simulation section. Work around: Expand the signals by hand or create a string statement which will be substituted.

Large CASE statements or many STRING statements sometimes generate a "SYSTEM_ERROR P103 Out of system memory" message during PARSE. It is caused by a compiler limitation. Work around: Try to cut down the number of STRING statements or the size of the CASE statement.

No error or warning is given by PARSE if a condition is set to the value '1'. The EXPAND program will issue an "UNKNOWN TOKEN" error. Work around: Substitute VCC for 1 and GND for 0. These are equivalent.

POWER_UP can only be used with the START_UP keyword. No error is given if it is used with another state until it is detected by the EXPAND program, which produces an internal error message.

'IF-THEN-ELSE' and 'CASE' statements: When a signal is set to '1' the ON cover is generated, and when a signal is set to '0', the OFF cover is generated. If the logic synthesis option for the 'IF-THEN-ELSE' CASE default value is set to

OFF (treating the "don't cares" as zero) then only one output cover is produced - the cover defined first in the design.

No warning message is given by the pre-processor when a PIN signal is input paired with a NODE signal, and that NODE signal is output paired with another PIN signal until the fitting stage. The Fitter issues a warning that the NODE output pairing is ignored.

For devices with a fixed power-up, reset and set signals are generated only if the global node is specified in the pin list. (Physical node 1 is always designated as the global node in PALASM 4 syntax.)

An incorrect warning message (B616) is given by the pre-processor for an input paired pin, stating a CLKF equation is not generated for that pin. This has no effect on the design and can be ignored.

Minimizer

Sparse equations with more than 32 inputs are noted with a memory allocation error by the Minimizer. Work around: Set MINIMIZE_OFF for that equation or run the Minimizer with the "Use Fast Minimization" logic synthesis option (see section 1.4 of these Release Notes).

If state machine syntax is used and the keyword CHECK is used to verify a signal value at the pin, setting the Logic Synthesis Option "Ensure polarity after minimization" to "Best for Device" may cause check clashes during simulation. Work around: Change the Logic Synthesis Option "Ensure polarity after minimization" to "As specified in the design file".

Fitting MACH devices

If product term reservation is used and any signals require more terms for logic than are provided, no error message is given by the Fitter.

If a design cannot be successfully routed again once it has been fully placed by the Fitter, report the problem to AMD Corporate Applications (USA: 800-222-9323) or your local AMD Field Application Engineer who will give you assistance. This problem is rare and is caused by changes in the signal routing order.

Fitting PAL devices

16V8: Placing a signal name of "TEST" on pin 16 produces the error "Error X3670 illegal signal attribute(s) reg". This affects only pin 16. Work around: Change the name of the signal on pin 16 to something other than "TEST".

16V8: The three-state equation is product-term driven only when used in combinatorial I/O mode. If the equation is set to GND the program does not give an error and a incorrect JEDEC file is produced.

PALCE610: An incorrect JEDEC file can be generated if a "/" is a PIN signal is defined as "/" and active high or low, T, J/K, or R/S equations are used. Work around: Do not define a PIN signal with a "/". Create active high or low equations by defining the "/" in the equation.

Simulation

The user cannot SETF to the value of "X" for unknown inputs.

Some dedicated clock pins can be used as both clock and data pins. Transitions on dedicated clock pins result in a JEDEC "U" or "D" vector being produced, allowing data from all other inputs to stabilize before a latch enable or clock transition happens. This can cause problems later as some data lines will be driven at the same time as clock signals on the JEDEC tester. To avoid this problem, write test patterns so that data and clock transitions occur in separate vectors.

20RA10: If pin 1 is not defined in the design file, simulation appears to continue but produces no output. Work around: Define pin 1 as something.

PLS167, PLS168 and PLS105 devices: If a preset or reset is written for one specific pin, the Simulator cannot apply that preset or reset to other pins. Workaround: Use the global preset and reset equations.

If the MACH Fitting option "Float all signals" has been used, the resulting placement file (.PLC) file and the original pin list will look very different. The simulator looks at both the last pin placement (.PLC) file and the original pin list and lists the conflicts between the two as errors instead of ignoring the orignal pin list.

Avoid using the same signal for the tristate output enable and the output latch enable. The design will fit, however the Simulator may not output the expected vectors. Work around: if check clashes occur on the programmer, modify the vectors, not the design.

Backannotating MACH designs

Signal names are limited to 14 characters; signals names over 14 characters are truncated by PARSE. This limitation will result in truncated signal names being ignored and lost when they are backannotated from the .PLC file.

Pinouts

The program which produces the pinout information does not read the .PLC file; therefore pins that have not been placed do not show on the pinout diagram. Work around: To show these unplaced pins on the pinout diagram, back annotate the signals using the .PLC file. Then change the Compilation run mode to manual, and set all the choices but "Check syntax" to "N". (Check syntax should be set to "Y".) Compile, then view the pinout diagram.

APPENDIX 3 - PALASM 4 .PDS FILE - MACH215 APPLICATION EXAMPLE

MACH215 Application Example - Flag Register

```
    TITLE Demo 215 Features Test
  PATTERN Flg_Reg.PDS
 REVISION 1.1
   AUTHOR Nick Schmitz
  COMPANY ADVANCED MICRO DEVICES, INC.
     DATE 6/16/1992

CHIP  Flg_Reg  MACH215

   PIN  ?  P1 IPAIR B2
   NODE ?  B1 OPAIR P1
   NODE ?  B2

   PIN ?  CLK1      PIN ? /CLK2      PIN ?  CLKS  ; Clocks
   PIN ?  R         PIN ?  P         PIN ?  ENA   ; Control signals
   PIN ?  X                                       ; Display output for testing

   PIN ?  I1 IPAIR BI1      NODE ?  BI1
   PIN ?  I2 IPAIR BI2      NODE ?  BI2
   PIN ?  I3 IPAIR BI3      NODE ?  BI3
   PIN ?   Y IPAIR BI4      NODE ?  BI4

   GROUP MACH_SEG_A BI1 BI2 BI3 BI4 P1 B2 X Y
   GROUP MACH_PTS_8  P1                          ; Reserve Space for PT growth

EQUATIONS

   P1.T := /B1 * (B2 :+: P1)
   B1.T := {P1.T}                  ; Node equation identical to pin eq
     B2 *= P1                      ; Transfer function input latch
      X = B2                       ; Display output for testing

   P1.TRST = ENA
   P1.RSTF = R
   B1.RSTF = R

   P1.CLKF = CLK1                  ; Must minimize to 1 PT
   B1.CLKF = {P1.CLKF}             ; Must be same as CLK for pin function
   B2.CLKF = CLK2
```

```
    Y:= BI1 + BI2 + BI3 + BI4
    Y.TRST = ENA
    Y.CLKF = /CLK1

    BI1 := I1      BI1.CLKF = /CLK2      BI1.SETF=P   ; MACH215 macrocell
    BI2 := I2      /BI2.CLKF = /CLK2     BI2.RSTF=R   ; Set/Rest on IPAIR
    BI3 := I3      BI3.CLKF = /CLK1 + /CLK2           ; CLK PT use
    BI4 := Y       /BI4.CLKF = CLKS                   ; Excess CLK resources


SIMULATION
    Trace_on  R P ENA    Y X     B1 P1 B2    CLK1 /CLK2  CLKS
         BI1 BI2 BI3 BI4       I1 I2 I3

    Setf /CLK1   CLK2 /ENA /CLKS /I1 /I2 /I3 ; Initialize inputs
    Setf CLKS /CLK2   R                      ; Rest state for active-low clocks

    Setf P1 /R                               ; Test out flag register
         Clockf CLK2
    Setf /P1    Y    I2
    Setf  CLK2                               ; Make latch B2 transparent

    Setf  P1                                 ; See data change on X
    Setf /P1       Setf   P1 I3
         ClockF CLK1                         ; Load BI3 = 1

    Setf /CLK2                               ; Close latch
         Clockf /CLKS                        ; Falling edge clock of BI4
    Setf  /P1 /Y    /I2
         Clockf /CLKS

         ClockF CLK1
         ClockF CLK1                         ; No more changes
    Setf ENA   I3                            ; Enable outputs P1 & Y

    Setf R P     Setf /R /P /I3              ; Set/Reset secondary regs
         ClockF CLK2                         ; Edge detect = 0

    Setf I1 /I2   /I3
         ClockF CLK2                         ; Load BI1 BI2 (Different edges)
    Setf /I1 I2 /I3                          ; CLK to BI3 disabled
         ClockF CLK2
    Setf   CLK2
    Setf   CLK1                              ; Enabled
         ClockF /CLK2                        ; Load BI3 = 0

    Setf /ENA
    Trace_off
```

MACH215 Application Example - Flag Register Simulation Results