Achieving High Performance in a CoolRunner™ XCR3960

XAPP313 (v1.0) October 22, 1999 Application Note

Summary

High-performance designs can be successfully implemented in the XCR3960 with a general understanding of its architecture and guiding the compilation and fitting processes with a control file. This document provides an overview of the XCR3960 architecture, compiler and fitter options along with an example to provide a resource for high-performance system designers.

Introduction

With any programmable device of significant density, an understanding of the architecture and software controls is required to achieve the highest possible performance of the device. The Xilinx CoolRunner™ XCR3960 is the highest density CPLD available on the market, combining high-performance with ultra-low power. It is a 960 macrocell SRAM based device that supports multiple configuration modes with 384 I/Os in a 492-pin BGA package. This document will explain the architecture of XCR3960 device, list software parameters for controlling the partitioning and placement of a design, and provide an example design using these features.

XCR3960 Architecture

Figure 1 shows a representation of the XCR3960 architecture. The XCR3960 consists of 12 Fast Modules interconnected by a Global Zero-Power Interconnect Array (ZIA). Each Fast Module consists of 80 macrocells, has 64 inputs and 64 outputs to the global ZIA, and is connected to 32 I/O pins. The delay across the global ZIA between adjacent Fast Modules is slightly faster than the global ZIA delay for non-adjacent modules. Fast Modules in the top row are numbered 0:5 left to right and Fast Modules on the bottom row are numbered 6:11 right to left. For example, Fast Modules 0, 1, 10, and 11 are considered adjacent.

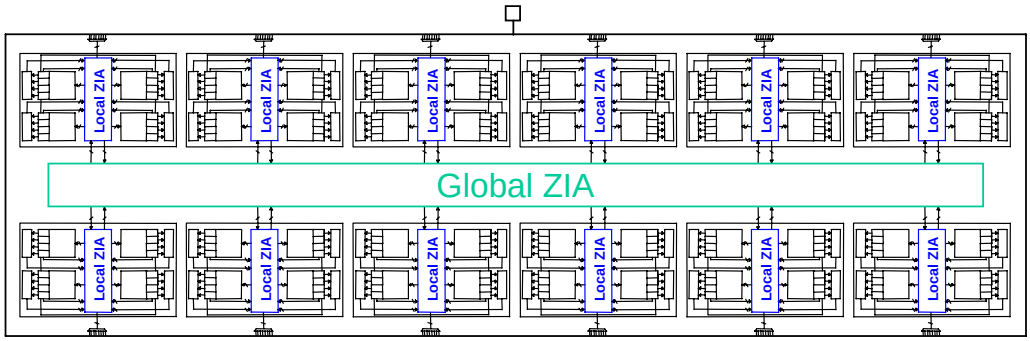

Figure 1: XCR3960 Architecture

Each Fast Module consists of four logic blocks with 20 macrocells interconnected by a local ZIA as shown in Figure 2. Each logic block has a fan-in of 36 from the local ZIA. The 32 I/O pins for the Fast Module are equally distributed – eight per logic block. Therefore, eight macrocells from each logic block are bonded-out to pins, the other remaining 12 macrocells are buried.

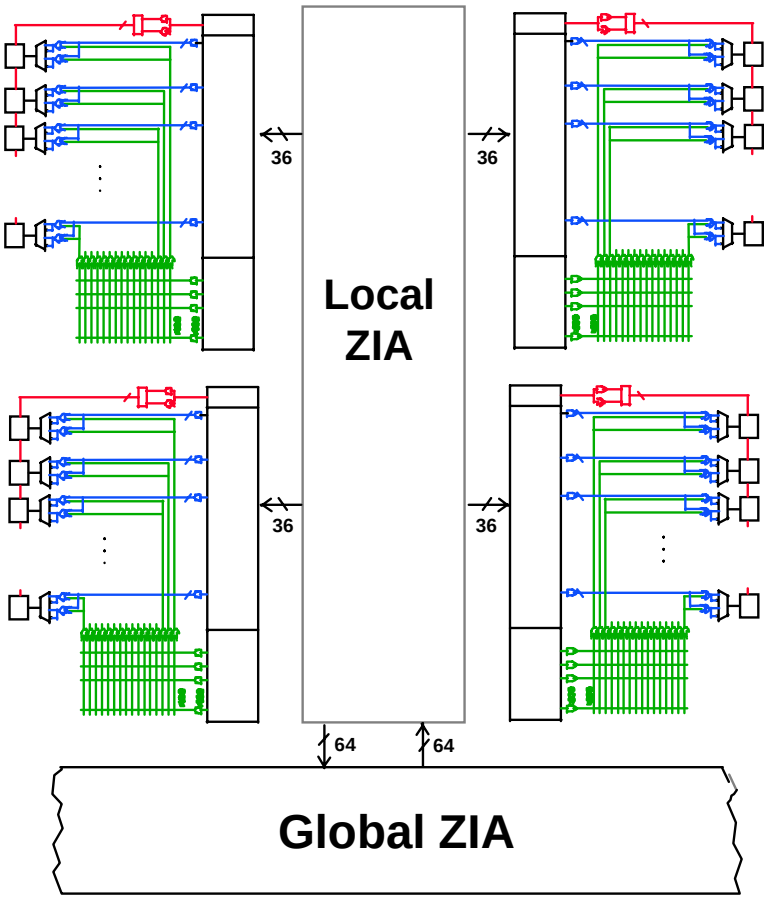

Figure 2: XCR3960 Fast Module Architecture

Each logic block consists of a PAL (programmable AND, fixed OR) and a PLA (programmable AND, programmable OR) array as shown in Figure 3. The PAL provides four dedicated product terms per macrocell. The PLA provides an additional pool of 32 product terms that can be used across all 20 macrocells in the logic block. Note that there is also a two-input hardware XOR gate with an input from the PAL and an input from the PLA for each macrocell. This allows for more efficient implementation of adders, parity functions, and checksum functions.

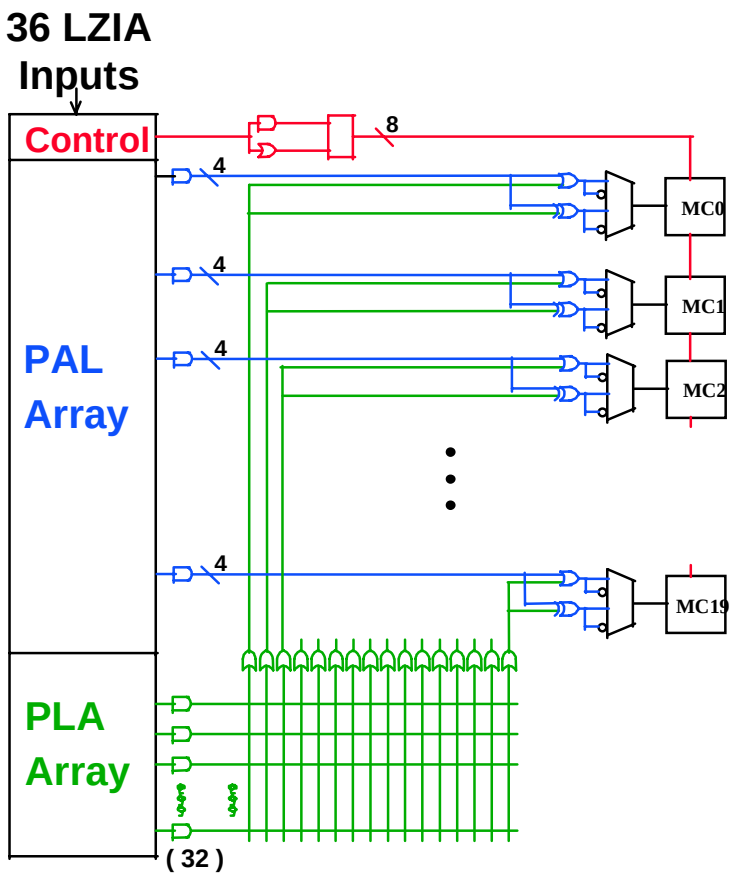

Figure 3: XCR3960 Logic Block

XCR3960 Timing Model

T_{PD_PAL} is the pin-to-pin delay through the combinatorial PAL array and is incurred when an equation is four product terms or less. When an equation requires more than four product terms, additional product terms are utilized from the PLA array and the time delay becomes T_{PD_PLA}. If the XOR gate is utilized, the delay is then T_{PD_XOR}. When a signal traverses the global ZIA from the input pin to the output pin, an additional delay is incurred, T_{GZIA}. The same type of model applies for registered signals, if the input to a register requires four product terms or less, the setup time is T_{SU_PAL}. If the input equation utilizes additional product terms from the PLA, the setup time is T_{SU_PLA} and if the XOR gate is utilized, the setup time becomes T_{SU_XOR}. Again, if the signal traverses the global ZIA from the input pin to the D input of the register, an additional T_{GZIA} delay is incurred. This is shown in Figure 4.

Table 1 gives values for these parameters for a commercial XCR3960. These numbers should be used for example only, refer to the data sheet for the XCR3960 for accurate timing parameters.

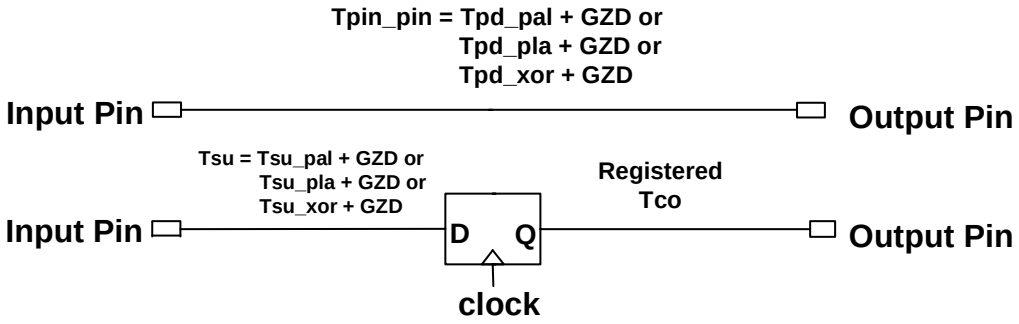

Figure 4: XCR3960 Timing Model

Table 1: XCR3960 Timing Parameters

Timing Parameter	Description	Min (ns)	Max (ns)
T_{PD_PAL}	Input to Output Delay through PAL		7.5
T_{PD_PLA}	Input to Output Delay through PLA		9.0
T_{PD_XOR}	Input to Output Delay through XOR		10
T_{SU_PAL}	PAL setup time to Global clock	4.0	
T_{SU_PLA}	PLA setup time to Global clock	5.5	
T_{SU_XOR}	XOR setup time to Global clock	6.5	
T_{FZIA}	Global ZIA delay for adjacent Fast Modules		3.0
T_{GZIA}	Global ZIA delay for non-adjacent Fast Modules		4.0
T_{CF}	Global clock to registered feedback		2.5
T_{CO}	Global clock to output		6.0

XCR3960 Architectural Features

Fast Modules and Global ZIA Delays

As seen in Table 1, the delay across the global ZIA can be 3 ns or 4 ns depending on the location of the source and target Fast Modules. It is therefore important to minimize the global ZIA delays for critical signals by locating the source and targets of these signals in the same Fast Module. When a global ZIA delay can not be avoided, it is important to place the source and target registers or pins in adjacent Fast Modules to achieve the minimum global ZIA delay.

Each Fast Module has 64 fan-in and 64 fan-out to the global ZIA. In complex designs where the fan-in to equations are quite high, it may become necessary to co-locate the target equation with the fan-in signals in the same Fast Module so that the fitter is not limited by the fan-in/ fan-out connections to the global ZIA.

To locate signals in the same Fast Module, the FM_GROUP property is specified either in a control file or in a PHDL/ABEL source file. This property does not specify a particular Fast Module for the group, it just instructs the placement algorithm to place these signals in the same Fast Module. To assign the group of signals to a particular Fast Module, assign at least

one of the signals in the group to a pin in the Fast Module. This will force the entire group of signals to this Fast Module.

PAL and PLA Product Terms

Each pass through the PAL or PLA in an equation adds the PAL or PLA delay to the path timing. The absolute fastest timing is achieved when equation has four product terms or less. However, if an equation has more than four product terms, allowing all additional product terms to be implemented in the PLA minimizes the overall path delay. The additional delay incurred by using product terms in the PLA is 1.5 ns, which is significantly less than the delay incurred by allowing another pass through the PAL.

Usage of the PLA product terms is controlled by the compiler option, *Max P-terms per Equation.* When this number is set to 36, the compiler is allowed to use four PAL product terms and all 32 PLA product terms for one equation if necessary. While this yields the fastest timing for complex equations, please note that this can restrict the utilization of the device. This can also increase the fan-in required by the equation such that the 36 fan-in to the logic block is violated. Setting this parameter to a lower number forces the compiler to create intermediate nodes for those equations requiring large number of product terms. These intermediate nodes are then fed back to the PAL/PLA arrays as required until the equation is implemented. Each pass through the array incurs additional delay, so the total delay for the equation may be quite large. The recommended approach is to first compile and fit the design with this parameter set to 36 and see if utilization or fan-in problems occur. If no problems occur, then this setting provides the fastest timing through the device. If there are problems fitting the design, note which signals require large number of product terms and/or fan-in. If these are not timing critical, or additional passes through the PAL/PLA still meet the required timing, then reducing this parameter may allow the design to fit. Note that this parameter can be set per equation in a control file.

Hardware XOR

The XCR3960 includes a two-input hardware XOR gate for each macrocell. This allows efficient implementation of arithmetic functions such as parity generators and checkers, checksum generation and verification, adders, and subtractors by reducing the number of product terms required by these equations.

The path through the XOR gate is slightly slower than the path through the PAL or PLA. The equations that are performing XOR functions should be reviewed by the designer to determine if the number of product terms to implement the XOR function can be taken from the PAL and PLA for the fastest time possible. Note, however, that the designer should also take into account the requirements for PLA product terms by other functions in the design to insure that there are enough resources in the device for the XOR function to use the PLA product terms. If the design is such that it is not possible to use a high number of product terms to implement the XOR function, the path delay through the XOR is still less than multiple passes through the PAL and PLA arrays. In this case, the best performance is achieved by allowing use of the hardware XOR gate.

Use of the XOR gate is controlled by the compiler option, *XOR Manipulation*. This option can be set to *All*, *Explicit*, or *None*. Setting this option to *All* means that the compiler will use the XOR gate in all cases where it recognizes the product terms for the XOR function. *Explicit* instructs the compiler to use this gate only when explicitly specified in the input source file (PHDL/ABEL source files only). The XOR gate will not be utilized when this option is set to *None*.

Global Reset and Global Tri-state Signals

In designs where a single input signal is used to reset or tri-state the entire device, the XCR3960 provides a global reset input and a global tri-state input. Note that these signals must be input signals, they can not be results of combinatorial or registered equations within the device. Use of the global reset and global tri-state functions utilize special routing within the XCR3960, therefore not utilizing fan-in and fan-out connections that may be required for actual signals. This allows full utilization of the XCR3960 in high-density designs.

Specification of the global reset and global tri-state signals are done via pin assignments. The global reset signal must be active Low and assigned to pin N23. Likewise the global tri-state signal must be active Low and assigned to pin N5. These pin assignments can be done in the control file or in the Pin Assignment File. The Pin Assignment File (PAF) can be created manually using an editor or by using the graphical pin editor.

Controlling the Compilation and Fitting Processes to Achieve High Performance

Compiler and Fitter Options

All of the compiler and fitter options can be set in a control file. Workstation users will use a control file to specify all options and properties. PC users can set compiler and fitter options (with the exception of *Max signal Fan-in)* in the Properties window of XPLA Professional as shown Figure 5. Since XPLA properties such as FM_GROUP can not be set in this window, a control file can be specified. Control files will be discussed later in this document.

Max P-term per Equation controls how many product terms can be used to implement the equation. For the XCR3960, the number of product terms per equation can range from 4 to 36. There are four dedicated product terms per macrocell in this device, therefore setting this parameter to a number > 4 indicates that the compiler can use PLA product terms for the macrocells. As discussed above, utilizing many PLA product terms reduces the path delay of the equation since multiple paths through the PAL are reduced. However, allowing many product terms per equation can increase the fan-in required by the equation and may cause fitting problems with the fan-in to the logic block and/or to the Fast Module. The designer should also be aware of the number of equations in a complex design that need a large number of product terms when setting this parameter so that there are enough PLA product terms to satisfy the requirements of the design.

Optimizing effort can be set to *Fast* or *Exhaust* to control execution time for the compiler. This parameter specifies the optimizing algorithm to the compiler.

XOR Manipulation field can be set to *None*, *All* or *Explicit* indicating how the compiler should use the hardware XOR gate in these devices. As discussed above, utilizing the XOR gate in the XCR3960 reduces the number of product terms required to implement arithmetic functions.

Activate D/T register synthesis, if checked, allows the compiler to optimize the design by choosing either D-type or T-type registers. If this is not checked, the compiler will only use the register types explicitly called out in the design.

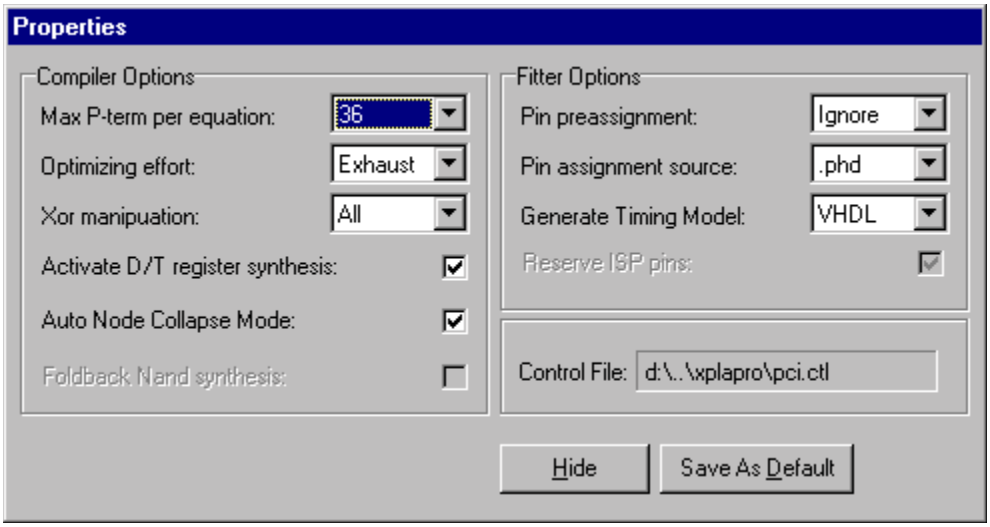

Figure 5: XPLA Professional Properties Window

Auto Node Collapse Mode, if checked, allows the compiler to automatically collapse internal nodes. Both of these options should be selected when optimizing the performance of a design in the XCR3960.

The pin assignment of a design can be specified in a Pin Assignment File (PAF) or in a PHDL/ABEL file if the PHDL/ABEL file is the top-level file of the design. The *Pin Assignment:* field can be set to *Try*, *Keep*, or *Ignore* and the *Pin Assignment Source*: can be set to *PHD* or *PAF*. Setting *Pin Assignment:* to *Try* instructs the fitter to try the current pin assignment specified in the *Pin Assignment Source:*. If the pin assignment can not be kept, a warning is generated and a new pin assignment is made, creating a new PAF file. If *Pin Assignment*: is set to *Keep*, the fitter uses the pin assignment from the file specified in the *Pin Assignment Source:*. If the pin assignment can not be kept, the fitter stops execution. If *Pin Assignment:* is set to *Ignore*, then the fitter ignores all previous pin assignments and creates a new pin assignment and a new PAF file.

It is possible to have both a VHDL and Verilog timing model produced by the design by setting *Generate Timing Model:* to *All*, *VHDL*, *Verilog*, or *None*. These models can be used for timing simulations in other CAE tools.

Setting *Max Signal Fan-in* for synthesis determines the allowed fan-in used per equation to guide the compiler in forming equations. If the fan-in of the logic block or Fast Module is being violated, this parameter can be used to force the compiler to reduce the fan-in to complex equations by creating internal nodes. Likewise, the fitter can be instructed to use additional fan-in resources to each logic block by setting *Max Signal Fan-in* for the fitter. *Max Signal Fan-in* is set by using Ctrl-Alt-Z to bring up the window shown in Figure 6.

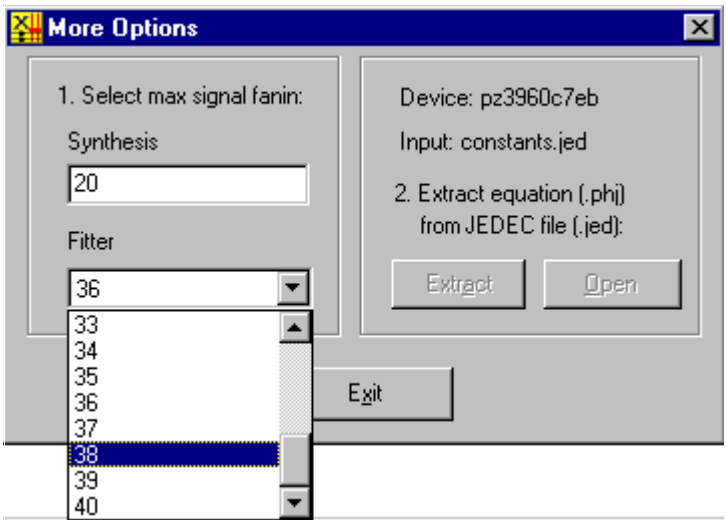

Figure 6: Setting Max Signal Fan-In

Properties

Properties can be set in the PHDL/ABEL source file or can be input into a control file. Table 2 lists the supported properties for CoolRunner CPLDs. Of most importance for achieving performance in the XCR3960 as discussed above is the FM_GROUP property. In cases where more placement control is required, the LB_GROUP properties can also be used to group signals, however, in most cases, grouping signals within a Fast Module achieves the desired performance.

Table 2: XPLA Professional Properties

Property	Description
dut off	Disable global tri-state function (default)
dut on	Enable global tri-state function (GTS pin is now active)
isp on	Reserve TCK, TMS, TDI , and TDO pins for ISP usage (default)
isp off	Disable ISP capability of the device – allow TCK, TMS , TDI, and TDO to be used as general purpose I/O
maxpt	Specify the maximum product terms for a specific pin or node
keep	Specify the keep attribute for a specific signal
retain	Specify the retain attribute for a specific signal
tri-state all	Disable the weak pull-down on unused I/O (all devices except 32 macrocells)
fm_group	Group the specified signals within a Fast Module (XPLA2)
lb_group	Group the specified signal within a logic block
slow_slew_rate	Assign slow attribute to output buffers to specify the slow slew rate, default is fast slew rate
config_master_serial config_master_parallel config_slave_serial config_slave_parallel config_sync_peripheral	Specify the configuration mode so that the fitter will not use the pins required by this mode unless it is necessary to fit the design (XPLA2)

The properties that specify the 960 configuration modes denote the required configuration pins to the fitter so that these pins are allocated as a last resort. This prevents the designer from having to reserve these pins. Board layout and design are much simpler when the configuration pins do not also function as I/O pins for the design.

The syntax for setting these properties in a PHDL/ABEL source file is shown in Figure 7.

```
xpla property '<property>';
```

Figure 7: Property Syntax

Format of the Control File

The contents of a control file can contain three sections, the command section, the property section, and the pin assignment section. Each section starts with the section keyword enclosed in square brackets. A control file can contain any combination of these sections as desired, i.e., not all sections have to be included in the control file. Comments in the control file are designated with a "#". An example control file is shown in Figure 8.

```
# this is a comment

[command]
-bfi 36
-th 20

[property]
maxpt bit0:12 bit1:14
dut on
fm_group a b c d

[pin_assignment]
bit0:4,
bit1:5,
bit2:6
```

Figure 8: Control File Format

Command Section ([command])

The command section of the control file sets the compiler and fitter options. Workstation users will need this section of the control file to set compiler and fitter options. This section is not needed for designers using XPLA Professional since all compiler and fitter options can be set using the GUI. The commands for setting these options are shown in Table 3.

Table 3: Supported Commands in a Control File

Command	Arguments	Description
-i	<design name>.[edf | v | phd]	PHDL, Verilog, or EDIF input file.
-it	<edif | verilog | phdl>	Input file type.
-th	<5 – 37>	Max product terms per equation – default is 11.
-fi	<5 – 37>	Max fan-in per equation – default is 36.
-bfi	<36 – 40>	Max fan-in per logic block – default is 36.
-v	<module name>	Generate delay-annotated Verilog simulation model with module name specified – default module name is design name.
-vho	<entity name>	Generate delay-annotated VHDL simulation with entity name specified - default entity name is the design name.
-reg		Perform D/T register synthesis.
-co	<best | none>	Collapsing method – default is best.

Command	Arguments	Description
-effort	<exhaust | fast >	Optimizing effort – default is fast.
-xor	<all | exp | none >	xor synthesis mode – default is none.
-dev	<device>	Target device.
-pre	<keep | try | ignore>	Pin assignment effort – default is try.

Property Section ([property])

This section is used to specify XPLA Properties when the source file is not a PHDL/ABEL file. XPLA Properties are specified in Table 2. If the source file for the design is a PHDL/ABEL file, these properties can be specified directly in that file or in a control file, however, the properties specified in the control file will overwrite the properties in the PHDL/ABEL file. This section is denoted by [property] and contains properties and their values as shown in Figure 9.

```
[property]
    maxpt BIT0:12 BIT1:12
    keep BIT0
    retain BIT1
    delay_mode BIT2
    dut on
    isp off
    tri-state all
    fm_group a b
    lb_group c d o1..o3
    slow_slew_rate o1 o2
    config_master_serial
```

Figure 9: Properties

Pin Assignment Section ([pin_assignment])

The format of a pin assignment consists of the signal name followed by a `13:"` followed by the desired pin number. To make the fitter use the pin assignments specified in the control file when using XPLA Professional, the *Pin preassignment* option must be set to *Keep* and the *Pin assignment source* must be set to `.phd`. If the Pin assignment source is set to `.paf`, then the pin assignments in the Pin Assignment File (`<design>.paf`) will be used (Figure 10).

```
[pin_assignment]
BIT0:4, BIT1:5
```

Figure 10: Pin Assignment File

PCI Design Example

A 33 MHz, 32-bit PCI target design was done in VHDL for the XCR3960. This design utilized 218 macrocells in four Fast Modules within the device. The fitter algorithm is such that adjacent Fast Modules are used first, therefore, the fitter placed the PCI design in Fast Modules 0, 1, 10, and 11. This minimized the global ZIA delay to T_{FZIA} instead of T_{GZIA}.

The compiler options were set as shown in Figure 5. The *Pin Pre-assignment* was set to *Ignore*, because pin assignments did not need to be kept. *Max Pterms per Equation* was set to 36 to achieve the fastest timing possible. There were no fan-in issues with this design, therefore it was not necessary to adjust the *Max Signal Fan-in* for either synthesis or fitting.

PCI Control File

To meet all of the PCI timing specifications, a control file was used to control the fitting process. This control file is shown in Figure 11 and will be explained in the following paragraphs. This control file simply grouped signals together in Fast Modules. It was not necessary to assign signals to specific Fast Modules, therefore a pin assignment section of the control file was not included. A command section was not needed either since the compiler options were set from the XPLA Professional GUI ().

Input Setup Time

The PCI Specification requires a 7 ns setup time on PCI interface signals as shown in Table 4.

In a XCR3960, the input setup time is determined not only by the equation involving the input signal as the D input of a register, but the placement of this input signal and the register as well. In general, input setup time is minimized by locating the input pin and the register using the input signal in the same Fast Module. If this is not possible, the input pin and the register should be placed in adjacent Fast Modules to reduce the delay through the global ZIA. The input setup time is further reduced by simplifying the equation to four product terms or less making the delay T_{SU_PAL}.

```
[property]
# group state machine with PCI interface signals to minimize input setup time
fm_group frame_n tar_sm__2 tar_sm__1 tar_sm__0 irdy_n

# group the parity checking functions together
fm_group perr_n perr_oe par_check serr_n serr_oe pc_node0..pc_node5 perr_bit_i

# group PCI_OE with PCI output signals to minimize output time (nx5406)
fm_group local_dis trdy_n stop_n nx5406

# group the parity with its OE
fm_group par par_oe

#specify the 960 configuration mode so the fitter will not use these pins unless absolutely
necessary
config_master_serial
```

Figure 11: PCI Control File

Table 4: PCI Input Setup Time Requirements

Signal	Setup Time (ns)
FRAME_N	7
IRDY_N	7
IDSEL	7
PAR	7
AD[31:0]	7
CBE_N[3:0]	7

For most of the PCI interface signals, only four product terms were needed and therefore grouping was not necessary because $T_{FZIA} + T_{SU\_PAL}$ was < 7 ns. However, for some equations, the number of product terms required use of the PLA and thus the delay through the Global ZIA and the delay through the PLA is > 7 ns. These signals were therefore grouped into the same Fast Module using the FM_GROUP property so the only delay on the PCI interface signals was the delay through the PLA. This enables the setup requirement of the PCI specification to be met for all PCI interface signals (Figure 12).

```
# group state machine with PCI interface signals to minimize input setup time
fm_group frame_n tar_sm__2 tar_sm__1 tar_sm__0 devsel_n irdy_n
```

Figure 12: Grouping of PCI Interface Signals Meet Input Setup Times

Clock Frequency

The clock frequency for the PCI design is specified as 33 MHz. The internal clock frequency of a design can be maximized by minimizing the delay between registers. Again, co-locating the source and target register within the same Fast Module to eliminate global ZIA delays minimizes the delay between registers. In cases where this is not possible, locating these registers in adjacent Fast Modules will decrease this delay. Designers should also reduce the complexity of the D equations so that multiple passes through the PAL and PLA arrays are not required, therefore minimizing the internal path delay to:

$$T_{CF} + T_{SU\_PAL}, T_{CF} + T_{SU\_XOR}, \text{ or } T_{CF} + T_{SU\_PLA}.$$

The checking of the incoming parity with the incoming data on the PCI AD and CBE_N busses must be accomplished in one 33 MHz clock cycle. Calculation of this 36-bit parity was the most complex function in the PCI design. By examining the 36-bit parity function, it was determined that the fastest implementation of this function in the XCR3960 was to use a combination of PAL/PLA product terms with the hardware XOR gate. A 3-bit XOR function can be accomplished with four product terms, therefore four PAL product terms were used to XOR three bits and four PLA product terms were used to XOR another three bits. The hardware XOR gate takes a PAL input and a PLA input, so the resulting XOR from the PAL (pc_xor0) and the resulting XOR from the PLA (pc_xor1) were then input into the hardware XOR gate. This produced six internal nodes (pc_node0 – pc_node5) that were then fed back into the PAL and PLA arrays for XORing via four PAL product terms (pc_nodex1) and four PLA product terms (pc_nodex2). Again, these results provided the inputs to the final XOR gate that finally produced the parity function. This is shown in the VHDL code below. Note that nodes pc_xor0 – pc_xor11, pc_nodex1, and pc_nodex2 were all collapsed (Figure 13).

```
pc_xor0 <= (   adi(0) and not (adi(1)) and not (adi(2)))
       or (     adi(0) and      adi(1) and      adi(2) )
       or (not (adi(0)) and      adi(1) and not (adi(2)))
       or (not (adi(0)) and not (adi(1)) and      adi(2) );

pc_xor1 <= (   adi(3) and not (adi(4)) and not (adi(5)))
       or (     adi(3) and      adi(4) and      adi(5) )
       or (not (adi(3)) and      adi(4) and not (adi(5)))
       or (not (adi(3)) and not (adi(4)) and      adi(5) );

pc_xor2 <= (   adi(6) and not (adi(7)) and not (adi(8)))
       or (     adi(6) and      adi(7) and      adi(8) )
       or (not (adi(6)) and      adi(7) and not (adi(8)))
       or (not (adi(6)) and not (adi(7)) and      adi(8) );

…
…
…

pc_xor10 <= ( adi(30) and not (adi(31)) and not (cbe_n_i(0)))
       or (     adi(30) and      adi(31) and      cbe_n_i(0) )
       or (not (adi(30)) and      adi(31) and not (cbe_n_i(0)))
       or (not (adi(30)) and not (adi(31)) and      cbe_n_i(0) );

pc_xor11 <= ( cbe_n_i(1) and not (cbe_n_i(2)) and not (cbe_n_i(3)))
       or (     cbe_n_i(1) and      cbe_n_i(2) and      cbe_n_i(3) )
       or (not (cbe_n_i(1)) and      cbe_n_i(2) and not (cbe_n_i(3)))
       or (not (cbe_n_i(1)) and not (cbe_n_i(2)) and      cbe_n_i(3) );

pc_node5 <= pc_xor10 xor pc_xor11;
pc_node4 <= pc_xor8 xor pc_xor9;
pc_node3 <= pc_xor6 xor pc_xor7;
pc_node2 <= pc_xor4 xor pc_xor5;
pc_node1 <= pc_xor2 xor pc_xor3;
pc_node0 <= pc_xor0 xor pc_xor1;

pc_nodex1 <= (     pc_node0 and not (pc_node1) and not (pc_node2))
              or (     pc_node0 and      pc_node1 and      pc_node2 )
              or (not (pc_node0) and      pc_node1 and not (pc_node2))
              or (not (pc_node0) and not (pc_node1) and      pc_node2 );

pc_nodex2 <= (     pc_node3 and not (pc_node4) and not (pc_node5))
              or (     pc_node3 and      pc_node4 and      pc_node5 )
              or (not (pc_node3) and      pc_node4 and not (pc_node5))
              or (not (pc_node3) and not (pc_node4) and      pc_node5 );

par_check <= pc_nodex1 xor pc_nodex2 after LOGIC_DELAY;
```

Figure 13: Coding of Parity Function Utilizing XCR960 Architecture

By using this combination of PAL/PLA product terms with the hardware XOR gate, this 36-bit parity function was accomplished with only two passes through the XOR gate, therefore meeting the PCI timing specification. To eliminate global ZIA delays, these signals were grouped within the same Fast Module (Figure 14).

```
# group the parity checking functions together
fm_group perr_n perr_oe par_check serr_n serr_oe pc_node0..pc_node5 perr_bit_i
```

Figure 14: Grouping of Parity Signals

Clock-to-Output Timing

Clock-to-output delays are specified at 6 ns as shown in Table 5.

Table 5: PCI Output Timing Requirements

Signal	T_{CO} (ns)
TRDY_N	6
DEVSEL_N	6
PAR	6
AD[31:0]	6
PERR_N	6
SERR_N	6
STOP_N	6

Clock-to-output time can vary based on if the output signal is registered or combinatorial. The fastest clock-to-output time is achieved for registered outputs and is simply T_{CO}. When the output pin is simply the output of a register, the output pin and the register will be placed in the same macrocell, so no global ZIA delays are possible. For high-performance CPLD designs and systems, it is highly recommended that outputs be registered.

In cases where the output pin is combinatorial function using the Q output of a register, the complexity of the combinatorial function determines the timing parameters involved. Designers should locate the register feeding this function and the output pin in the same Fast Module to avoid global ZIA delays and minimize the product terms of this equation so that multiple passes through the PAL or PLA arrays are not required. For equations with four or less product terms, the clock-to-output timing is $T_{CF} + T_{PD\_PAL}$ or $T_{CF} + T_{PD\_XOR}$. In cases where the system will allow slightly longer output delays, additional product terms can be implemented in the PLA. In this case, the clock-to-output timing is $T_{CF} + T_{PD\_PLA}$. Pipelining may be required if it is not possible to reduce the product terms in the equation or the device is so heavily utilized that PLA product terms are all used.

For tri-statable outputs or bidirectional signals, designers should note the output enable delay. In some cases, it may be necessary to group the output enable signal with the outputs it controls to eliminate global ZIA delays.

All outputs in the PCI design were registered, therefore the clock-to-out delay was T_{CO} (6 ns) and the specification was met without any grouping. The output enable signals for the tri-statable PCI control signals were grouped with these signals so that once enabled, the output buffer would drive as quickly as possible. Note that the output enable for the PCI control signals obtained a machine generated name through the synthesis process. Depending on the

synthesis tool used, this name can vary. To easily identify this signal, the pci.ph1 file was searched after a successful compilation and to find the equation for the output enable for any of the PCI control signals. For example, the equation:

```
trdy_n.OE = nx5406.Q; "--- [PT=1, FI=1, LVL=1] ---
```

indicated that the output enable for the TRDY_N signal is NX5406. This signal name was then used in the FM_GROUP property as shown in Figure 15.

```
# group PCI_OE with PCI output signals to minimize output time (nx5406)
fm_group local_dis trdy_n stop_n nx5406

# group the parity with its OE
fm_group par par_oe
```

Figure 15: Grouping Outputs and Output Enables

Revision History

Date	Version #	Revision
10.22.99	1.0	Initial release.