Fitting Designs Efficiently Into CoolRunner CPLDs

XAPP327 (v1.0) November 24, 1999

Application Note

Summary

Design performance is directly related to how well a design is fit to a CoolRunner™ CPLD, therefore it is important to understand the architecture of the CoolRunner CPLDs as well as how the compiler/fitter behaves. This document briefly reviews the architecture of the CoolRunner CPLDs, but more specifically describes how to control the compiler and fitter options in XPLA Professional to obtain the desired fit.

Introduction to CoolRunner Architecture

There are currently three different basic architectures in the CoolRunner families. These are the XPLA1, XPLA1 Enhanced and XPLA2 architectures. Each have their own subtle differences which will be reviewed here. It is important to understand the architecture of the target device in order to obtain the optimal fit for your particular design.

XPLA1 Devices

This architecture consists of logic blocks containing 16 macrocells each. All logic blocks are connected together by the Zero-power Interconnect Array (ZIA). These logic blocks have a maximum fan-in of 40 signals from the ZIA. The data sheet states that 36 primary fan-in signals exist and by default, the software will use up to 36. It is possible to use the four additional signal fan-ins and this will be discussed later.

Each macrocell uses signals from a PAL (programmable AND array with a fixed OR array) and a PLA (programmable AND array with a *programmable* OR array). Each macrocell has access to five product terms (P-Terms) from the PAL and 32 from the PLA. This allows each macrocell to use up to 37 P-Terms. The advantage of the PLA is macrocells within a logic block can share product term equations without wasting resources. Duplicate equations would generate wasted resources if the PLA was not available. The PLA is key in fitting large designs by reducing redundant logic.

Each macrocell can be configured to be either a combinatorial path, a D flip-flop or a T flip-flop. Available densities consist of 32, 64 and 128 macrocells.

XPLA1 Enhanced Devices

The XPLA1 Enhanced devices are simply XPLA1 devices with a few added features. There are more clocks available to the macrocell, in the form of product term clocks, which can originate from any pin or equation driven source. These product term clocks are referred to as control term clocks or enhanced clocks. The basic construction of the logic blocks, PAL, PLA and macrocells are essentially the same as the XPLA1 devices.

XPLA2 Devices

These devices can be thought of as the next generation of XPLA1 Enhanced devices since much larger macrocell densities are available. In fact this family contains the worlds largest CPLD, the 960 macrocell CoolRunner CPLD. There is also a 320 macrocell device available.

This architecture is very similar to the XPLA1 Enhanced parts. A two level interconnect architecture exists in the XPLA2 devices. The first level is the Local ZIA which is very similar to the previously discussed ZIA. A second level is known as the Global ZIA which connects Fast Modules. Fast Modules house the Local ZIAs and contain four logic blocks each. The logic blocks are connected together with the Local ZIA in much the same fashion as the XPLA1 devices. It is important to know that any time a signal uses the Global ZIA, the signal will experience an additional timing delay.

Logic Blocks, in the XPLA2 devices, also contain PALs and PLAs but differ from XPLA1 devices with two exceptions. The PAL supplies four P-Terms per macrocell (instead of five as in the XPLA1 devices) resulting in a maximum of 36 P-Terms available per macrocell. Second, there is also a hard wired, two input XOR that has one input from the PAL and the other from the PLA. This is useful for reducing the logic required to implement mathematical equations. The PLA remains unchanged from XPLA1 devices. Each logic block contains 20 macrocells.

From a very general view, the XPLA2 devices are essentially constructed of several 80 macrocell devices (Fast Modules) interconnected by the Global ZIA.

Compiler Options

This section discusses the options available to the designer during the compiling process using the CoolRunner compiler/fitter software, XPLA Professional. It is important to keep in mind the resources that are available in the target device during this process. The compilation process of XPLA Professional does not take into account the limitations of the target device. Many times a large design will fit to the target device when the compiler options are adjusted to reflect device resources.

Max P-Terms per Equation

This is probably one of the most important adjustments the designer can make to the compiling process. Values here adjust the maximum number of allowable P-Terms per macrocell equation. This setting is available from the "Properties" window as shown in Figure 1 which is accessed by pressing the Properties button from the XPLA Professional main GUI .

As discussed earlier, there are a fixed number of P-Terms per macrocell depending on the architecture. There are five for XPLA1 and four for XPLA2 devices. These numbers then become the minimum value to which the designer can adjust. The maximum value is then,

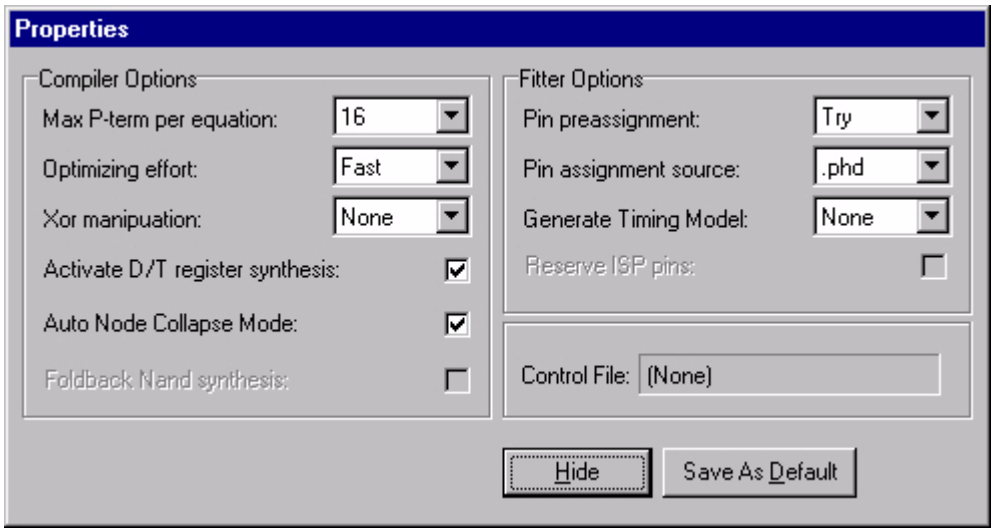

Figure 1: Properties Window

appropriately, the sum of the P-Terms in the PLA and the PAL for each macrocell. Maximum values are then 37 for XPLA1 and 36 for XPLA2.

Wide logic (i.e., large number of product terms) can be implemented in two basic fashions. The first is where PLA terms are used and the second is where intermediate nodes are created which forces more than one pass through the logic array. The latter will reduce the f_{max} of the device.

Designs that use the PLA will add the delay time associated with this structure. Please see the data sheet for specific values. Designs that use additional paths through the logic array will impose a $T_{PDF} = (T_{PD} – T_{BUF})$ delay for each path through the array.

There is a trade off between system speed and density. If speed is crucial, then a larger value for Max P-Terms per Equation should be selected since this will reduce the amount of nodes created. Fewer passes through the logic array increases system speed. If it is difficult to fit the design (i.e., density issue) then the Max P-Terms per Equation value should be reduced. Many times this will allow the design to fit. This may seem counterintuitive since this creates more nodes, but many times the design will not fit due to wide logic using all P-Terms. Also, a design may not fit due to increased fan-in by setting the Max P-Terms per Equation setting to a larger number. This issue will be discussed later.

Optimization Effort

Also available in the Properties window is the Optimization Effort setting. There are three settings:

- Fast
- Exhaust
- None

Selecting None will simply not optimize the equations and all logic will be retained in their un-reduced Boolean form. This setting is not normally used.

The settings Fast and Exhaust are very similar. Both will tend to take the same amount of time to fit (or fail if the design does not fit). The main difference is that they use different seeds to initiate the fitting process. This means that the user may get different results from fitting by using either setting. It is important to try both settings if the design does not fit the first time since the design may fit using a different seed. It is recommended to start with the Fast setting.

XOR Manipulation

The XOR Manipulation setting, available in the Properties window, is used for XPLA2 devices only since these devices have a hard wired XOR as described in the architecture review. There are three settings here:

- None
- All
- Explicit

By selecting None, you are instructing the compiler to reduce all XOR logic to sum and product terms. This tends to waste resources.

Explicit is only useful in the ABEL type design source. This setting allows the compiler to find and use XOR equations that are explicitly called for in the source code where the reserved word and attribute `ISTYPE 'XOR'` is used. This type of XOR entry must be accompanied by the `.X1` and `.X2` dot extensions. The compiler will not find implicit XOR functions. In other words, any XOR function created by reduced logic.

The All setting, for ABEL, will find and use all XOR functions as in the Explicit setting as well as any implicit XOR functions described above. This setting is useful for all forms of design entry. EDIF and higher level languages other than ABEL will use this setting exclusively to implement the XOR function in the hardware XOR gate.

Activate D/T Register Synthesis

Activating this feature in the Properties window instructs the compiler to reduce the registered logic for best fit using either a D or T type flip-flop. Some logic is better suited for one or the other type and therefore, in order to minimize the resources utilized, the compiler will automatically select the best choice. For example, counters are much better suited for T flip-flops than the D type. In this case, implementing a counter in D flip-flops will dramatically increase in logic as the size of the counter gets larger. If this setting is enabled, the compiler will disregard the design source specified flip-flop and choose either D or T type based on which type uses fewer P-Terms. If both types yield the same number of P-Terms, the compiler will default to the D type flip-flop.

Disabling this feature will cause the compiler to use the type of flip-flop specified in the design source. If no type is specified, the compiler will default to the D type.

Auto Node Collapse Mode

Enabling the Auto Node Collapse Mode in the Properties window directs the compiler to minimize passes through the logic array by collapsing intermediate nodes into other logic. The compiler will not collapse any signals that have the keyword and attribute `ISTYPE 'KEEP'` specified.

When this mode is disabled, the compiler will not attempt to collapse any intermediate node with one exception. If the keyword and attribute `ISTYPE 'COLLAPSE'` is specified in the source code or control file, the compiler will collapse this signal regardless of the Auto Node Collapse Mode setting.

Collapsing nodes will tend to result in equations comprised of more product terms. This is always better where timing is a major concern. Doing so will speed up the design since the signal does not pass through the logic array as many times. Although, the PLA may be used to implement a collapsed node which will have its own delay. This delay is much shorter than the t_{pd} delay realized with multiple passes through the logic array. Keep in mind that this may increase the fan-in required by the equation.

Max Signal Fan-In

A feature that is not advertised is the More Options window shown in Figure 2. This window is accessible by simultaneously typing <CTRL><ALT>Z from the XPLA Professional main GUI. The item of interest for the compiler is the "Select Max Signal Fan-in" portion of the screen. The field labeled "Synthesis" is the compiler directive for the maximum number of allowable signal fan-in to the equation. By default it is set to 36. Setting this number to a smaller number will tend to create nodes since the compiler will break up the equations into smaller equations. These pieces will then be assigned to other different blocks which effectively reduces the fan-in to the logic block for that equation. This helps the fitter to place and route signals with large fan-in.

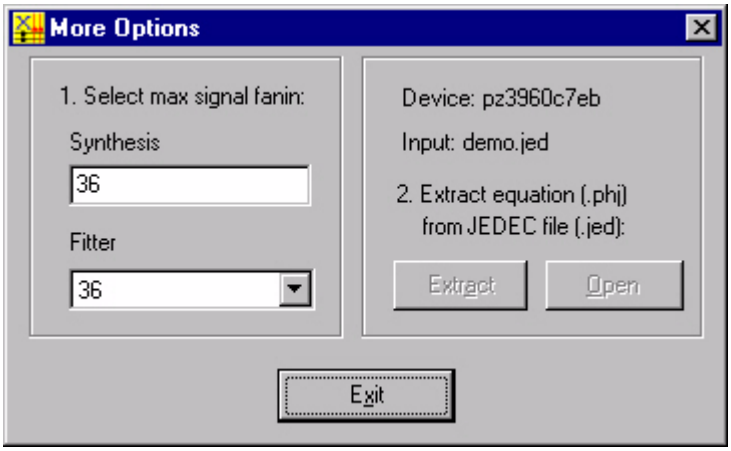

Figure 2: More Options Window

Jumping ahead a little to the fitter options, there is also a field entitled "Fitter". Similarly, this directs the fitter to limit the maximum signal fan-in to the specified value for the logic block and by default is also set to 36. This fitter option is discussed here in preparation of the following important note.

In reality, the XPLA1, XPLA1 Enhanced and XPLA2 devices have 40 signal fan-ins available to each logic block. The compiler and fitter options may be set to 40 for this reason. It is important to realize why the default setting is 36. The CoolRunner family of CPLDs is well known for its pin locking capabilities. One of the factors involved in maintaining pin-out with last minute design changes is the maximum signal fan-in. If the compiler or fitter is set to 36 for maximum signal fan-in, and there is a last minute design change that requires more fan-in for a design already containing many fan-in signals, there are four more fan-in signals available. The probability that the design will fit, in this case, is very high. Consider the scenario where the design was originally compiled/fit with the setting greater than 36 and approaching 40. The probability that a last minute design change will fit becomes significantly reduced the closer to 40 the setting was originally specified. In this case, when a new signal is needed, the ZIA may not be able to provide the required signal due to routing limitations. Therefore, it is recommended to keep these settings at 36 or less.

Fitter Options

This section discusses the fitter options available in XPLA Professional. Fitter options adjust how the fitter will assign and route the signals from the design to the target CoolRunner. This process is performed after the design has been compiled. The fitter uses the compiled output as its input and therefore will fit a design differently as a result of different compiler settings.

Pin Pre-assignments

The Pin Pre-assignments option, available in the Properties window, is used to instruct the fitter how to assign signals to pins. There are three options:

- Try
- Keep
- Ignore

If the selection is set to Try, the fitter will attempt to route the signals to the pins specified in the pin assignment source file. The pin assignment source file is discussed in the next section. If the fitter determines that the design will not fit with the assigned pins, it will notify the user that it cannot keep the pin assignments. Next it will proceed to assign the signals to pins that will

make the design fit. If using the pin assignment file as the pin assignment source file, it will be overwritten by the software and the user cannot automatically revert to the original assignments.

If Keep is selected, the fitter will try to fit the design to the preassigned pins only. If it cannot fit the design, it will give the same warning message and then stop the fitting process. No pin assignments will be modified by the fitter using the Keep setting.

The setting Ignore will cause the fitter to disregard the pin assignments called for by the designer. Pin assignments will then be set in a fashion that corresponds to the first available fit the software encounters. The resulting pin assignments can be found in the Pin Assignment File (`*.paf`). If it is desired to retain this pinout for future use, the designer should use these pin numbers/signals in the design source file. It is often best to allow the software to select the pinouts since this usually is the best fit for performance and subsequent design changes.

Pin Assignment Source

Two methods of pin assignment are available:

- `.phd` (ABEL/PHDL source code or Control File)
- `.paf` (Pin Assignment File)

To select which source the fitter uses for pin assignment, the user should choose either of the above selections in the Properties window under the Pin Assignment Source field.

Pin assignments in the ABEL source code can be implemented in the Declarations portion of the file and in the following manner:

```
signal_in pin 5;
signal_out pin 23 istype 'reg';
```

Notice the numbers in the above examples are the pin numbers, and the names signal_in and signal_out are input and output signals respectively.

The Pin Assignment File contains signal names and their associated pin numbers. This file is viewed and modified by the pin editor as shown in Figure 3.

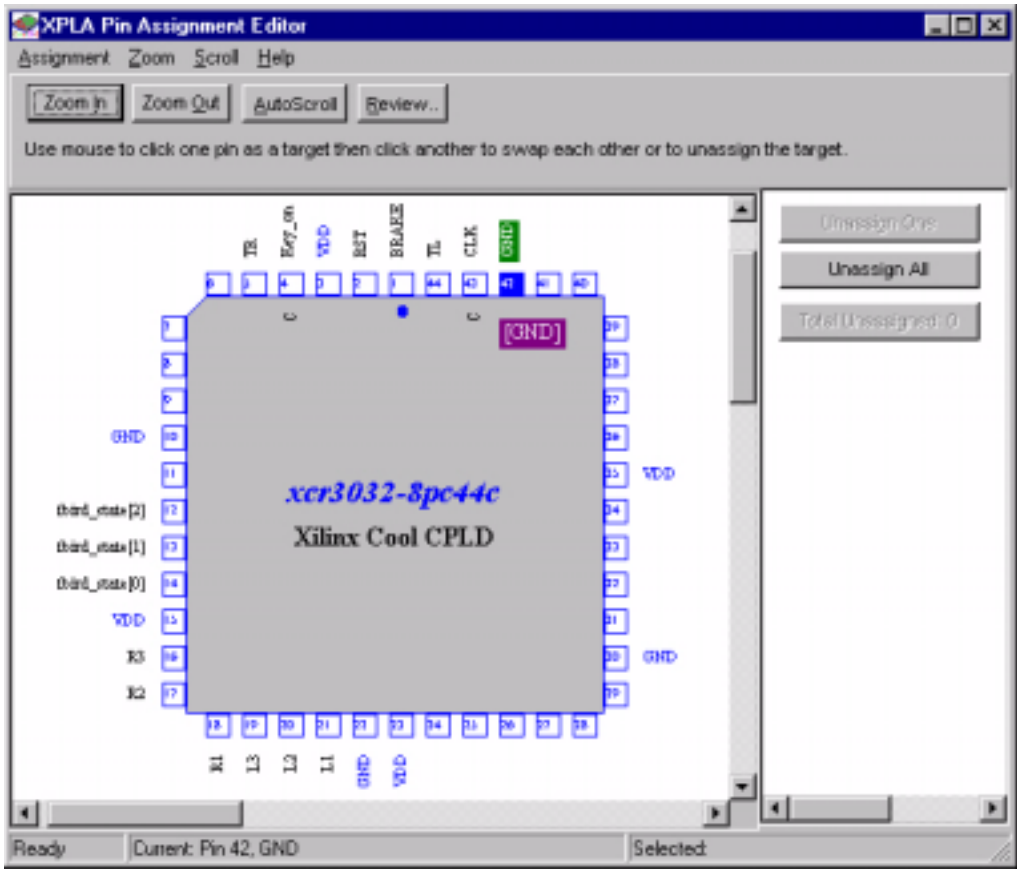

Figure 3: Pin Assignment Editor

The left hand window displays the target package from a top view and the right hand window shows the unassigned signals. When the mouse is moved over a pin on the package, the editor will show the function of that pin. To assign signals to pins, click the signal name in the right hand window and then click the corresponding pin in the left hand window. The signal name will then appear on that pin. To move a signal from pin to pin, click the pin with the signal to be moved and then click the destination pin. The signal name will then move to the destination pin. After assigning all signals to pins, save the file by selecting the <Assignment><Save> entry from the pull-down menu at the top of the window. This writes all signal/pin assignments to the `*.paf` file.

Generate Timing Model

Four choices are available from the Properties window for generating timing models:

- All
- VHDL
- Verilog
- None

Selecting All will generate VHDL and Verilog timing models which contain the actual timing parameters as the design was fit to the target device.

If VHDL or Verilog is selected, only the timing model for that HDL is generated and the other is ignored.

If None is selected, no timing models are created.

Note that when an equation is implemented in both the PAL and the PLA, the timing for all P-Terms will be set to the PLA timing. This means the signals through the PAL P-Terms for that equation will be reported as being slower than they really are since PLA P-Terms are slower. When an equation uses PAL P-Terms exclusively, the timing will reflect the PAL timing. This occurs in the VHDL and Verilog timing models and does not affect the XPLA Professional Timing Simulator. The XPLA Professional Timing Simulator takes into account all appropriate time delays as they will actually occur in the device. In other words, the Timing Simulator will include both the PAL and PLA timing delays for an equation that contains both types of P-Terms.

Control File

A Control file is a text file generated in any text editor and saved with the extension .ctl. This file can contain two sections, the `[property]` section and the `[pin_assignment]` section. Either or both fields may be used. Comments are text preceded by a "`#`".

To tell the fitter that the control file exists and is to be used, the user must right click in the "Control File" field found in the "Properties" window. A menu will appear and the option "Select..." should be chosen. Next a second window will appear that will allow the user to navigate towards the control file location. Once the file has been selected, it will appear in the "Control File" field.

Property Section - [property]

The Property section is used to specify the XPLA Properties used to control the fitting process. Table 1shows the available XPLA Properties. These properties may be used in either the Control file or in the ABEL/PHDL file. If used in the ABEL/PHDL file, the keyword and property name syntax:

```
XPLA PROPERTY 'property_name';
```

where `property_name` is from Table 1, must be used at the top of the file. Properties that are used in the control file will have precedence over those used in the ABEL/PHDL file.

Table 1: XPLA Properties

Property	Description
dut off	Disable global tri-state function
dut on	Enable global tri-state function (GTS pin is now active)
isp on	Reserve TCK, TMS, TDI and TDO pins for ISP usage
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 in all devices except the 32 macrocell devices
tri-state <pinname: pinnumber>	Disable the weak pull-down on the specified unused I/O pin for all devices except the 32 macrocell devices. Note that a pin name must be specified (even though the pin is unused)
fm_group	Group signals within a fast module (XPLA2 only)

Table 1: XPLA Properties (Continued)

Property	Description
lb_group	Group signals within a logic block
slow_slew_rate	Assign SLOW attribute to output buffers to specify the slow slew rate. Default is fast slew rate. (XPLA2 only)
config_master_serial config_master_parallel config_sync_peripheral config_slave_serial config_slave_parallel	Specify the configuration mode so that the fitter will only use the I/O pins required by this configuration mode when necessary. (XPLA2 only)

An example properties field is as follows:

```
[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 p1..o3
slow_slew_rate o1 o2
```

Pin Assignment Section - [pin_assignment]

The Pin Assignment Section defines what signals belong to which pins in a similar fashion to the Pin Assignment File (`*.paf`). This is useful when the control file is favored for specifying how the fitter functions, for example, during EDIF tool flows.

To direct the fitter to use the pin assignments called for in the control file, the user must select Keep from the "Pin Preassignment" field and .phd from the "Pin Assignment Source" field.

The syntax is shown below where the signal name is first followed by a colon and then the pin number. A comma must separate each pin.

```
[pin_assignment]
BIT0:4,
BIT1:5,
BIT2:6
```

Max Signal Fan-In

As discussed in the Compiler Options section of this document, the fitter also has a setting for the maximum signal fan-in to the logic blocks. Again, to get to the "More Options" window, simultaneously type <CTRL><ALT>Z. Adjusting the number in the "Fitter" field will direct the fitter to use no more signal fan-ins than are specified for the logic block.

Together with the "Synthesis" setting, the "Fitter" setting can greatly affect the ability of a large design to fit into a target device. The intricacies of this process will be discussed later.

As a reminder, adjusting either of these settings above 36 will dramatically reduce the ability of the fitter to make a later design change fit while retaining a fixed pin-out, subsequent to a successful fit of a large design.

Tackling the Problematic Design Fit

There are numerous reasons why a particular design will not fit to the target device. This section discusses the possible cases where a design will not fit and their remedies. Remember the old saying: "Patience is a Virtue." Sometimes it is a little time consuming to get a difficult design to fit, but in most cases, high utilization designs can be made to fit. Users should be systematic in their approach to discovering what the device limitations are concerning the design in question.

Clocks

Quite often the user has selected the incorrect target device. This sounds silly, but it happens. For example, if the design requires one more clock than the device can implement, the design will not fit. This is very often the case in designs that are targeting the XPLA1 devices which have limited clocks as shown in Table 2. In this case, the user may have intended to target the XPLA Enhanced devices which supply more clocks as shown in Table 2. If too many clocks are specified in the design, consider clocking some macrocells with existing clocks to reduce the overall number of clocks specified. Another possibility is to make use of clock enables to share existing clocks. Please see the Xilinx Application note XAPP305 "*Understanding Coolrunner Clocking Options*" regarding the use of clock enables. Of course, a larger device in the same package may also be an option.

Table 2: CoolRunner Clock Resources

Part	Global Clocks	Enhanced Clocks	Total Clocks	Global Clocks per Logic Block	Enhanced Clocks per Logic Block	Total Clocks per Logic Block
32	2	-	2	2	-	2
64	4	-	4	4	-	4
128	4	-	4	4	-	4
32 Enhanced	2	4	6	2	2	4
64 Enhanced	4	8	12	4	2	6
128 Enhanced	4	16	20	4	2	6
320	8	32	40	2	2	4
960	8	96	104	2	2	4

If the design is well within these limitations, it is possible that there are too many clocks specified per logic block in a fixed pin design. Table 2 shows the clock resources available for the logic block. Remember that the enhanced clocks are only available to the logic block where it (the control term) resides and not to other logic blocks. XPLA2 devices only allow any two of the eight global clocks to enter the fast module. If too many clocks are called for in the logic the designer will need to move logic to another logic block. The easiest manner to do this is to move a node to another logic block. This is done by specifying the signal name at the node and the node number much in the way that the pin assignment is done. Node numbers are available in

Appendix B of the *XPLA Professional User's Manual*. The designer may also consider clocking macrocells with one of the other clocks available or implementing clock enables in the logic block if moving a node is impossible.

For a better understanding of CoolRunner clocks, please see the Xilinx Application note *XAPP305* "*Understanding CoolRunner Clocking Options*".

So, how does the user find out how many clocks are called for in the design? The compiler will generate a file with a `*.ph1` (Optimized Equation) extension. This file is very handy when it comes to determining design requirements on device resources. Searching for the .clk extension within this file will reveal the clocks required by the design. Simply search for and count the number of different equations for each `.clk` signal and this equates to the number of clocks needed in the device. If too many exist for the device, some equations may be similar enough to combine and reduce the number of overall clocks needed.

Product Terms

The Max P-Term per Equation setting can be adjusted to make a design fit. There are two limitations that must be considered while adjusting this value. First, as the setting gets larger, more PLA P-Terms are used per macrocell which can lead to using all PLA P-Terms in the logic block. Additionally, since more P-Terms are allowed in the logic block, more fan-in may be necessary which can lead to all fan-in being used for that logic block. The advantage here is that the design can be faster. Second, with a smaller value, more macrocells are used as intermediate nodes are created. This can lead to too many macrocells required for the target device. The advantage is that there is higher device utilization concerning the PAL and PLA. Both conditions can lead to an unsuccessful fit. This means that a happy medium must be found for a successful fit.

Design Speed

A larger setting in the Max P-Term per Equation field will allow the compiler to use more P-Terms from the PLA when implementing the equations. This will cause the design to run faster since fewer passes through the logic array are performed. Consider the following example.

Assuming we are using an XPLA1 device, and the Max P-Term per Equation field was set to six, the compiler would be instructed to use one PLA term at the most per equation. Assume that the signals `A` through `H` in the following equation are, in themselves, separate equations requiring one product term each. If the equation was the following:

```
OUT = A # B # C # D # E # F # G # H
```

there are eight P-Terms which is greater than the six allowed by the setting. The equation cannot be implemented in one macrocell. The compiler will break the equation into two equations with one being an intermediate node as follows:

```
BURIED_NODE = A # B # C # D # E;
OUT = BURIED_NODE # F # G # H;
```

This implementation requires that the signal OUT contains equations that make two passes through the logic array. Signals `A` through `E` travel through the PAL array to reach the macrocell named `BURIED_NODE` which then passes through the PAL again to finally arrive at the macrocell named `OUT` for a total of two passes. The path delay then becomes the sum of T_{PD_PAL} for the first pass and T_{PD_PAL} for the second pass. If this equation was registered, the path delay would then become the sum of T_{PD_PAL} for the first pass with T_{SU_PAL} in the second pass.

If the Max P-Terms per Equation setting was adjusted for eight or more, the T_{PD} would simply be one T_{PD_PAL} since only one pass through the logic array was needed. This is the obvious choice for speed.

Device Utilization

There can be equations where many device resources are required which may prevent a successful fit. Consider the case where many macrocells require a large number of P-Terms. The way the Xilinx fitter for the CoolRunner parts fits designs is to first use all of the four or five (depending on the architecture) macrocell dedicated P-Terms in the PAL for each equation and then use additional PLA P-Terms for the remaining products. These P-Terms together comprise the original equation. If all of the P-Terms are used in a logic block, the design will fail to fit.

For example, consider the 32 macrocell XPLA1 device where the design calls for 16 outputs with each requiring ten P-Terms each. Note that there are a total of 64 PLA P-Terms available between the two logic blocks in this particular device. If the Max P-Terms per Equation setting was 10, the software will generate 16 equations with 10 P-Terms each when compiling. During the fitting process, the software will allocate the first five P-Terms to the PAL and then the remaining five P-Terms per equation to the PLA. The design will not fit since each of the 16 equations require five P-Terms from the PLA for a total of 80 P-Terms. The PLA only contains 32 P-Terms per logic block and between the two logic blocks there are 64 available PLA P-Terms for this application. Therefore, since there are not enough PLA P-Terms available, the design will fail the fitting process.

However, if the Max P-Terms per Equation was set to six, the compiler would create 16 nodes of six P-Terms each consisting of five PAL P-Terms and one PLA P-Term. The signal from the buried node would then propagate through the logic array to join the remaining four P-Terms from the original ten. This would then use five PAL P-Terms of another macrocell. Overall, the design would fit using all 32 macrocells and only 16 PLA P-Terms. This implementation would then utilize the architecture to its fullest but will not be as fast. There is a trade off between speed and device utilization as shown in this example.

How does the designer determine the amount of P-Terms required for a particular design? The compiler will generate a `*.ph1` file which contains the compiled and reduced equations. Each equation shows their device demands in the form of P-Terms (e.g., `PT=1`), logic block fan-in (e.g., `FI=2`), and levels of logic required to make the equation which involves counting the number of nodes used for that equation (e.g., `LVL=1`). The P-Term number does not distinguish the PLA from the PAL P-Terms. The designer can review the P-Term numbers and roughly estimate how many PLA P-Terms are required. It is not possible to tell how many are truly needed until the design has successfully fit into the device. The reason for this is that some of the P-Terms will be shared during the fitting process therefore reducing the number of P-Terms appearing in the `*.ph1` file. A handy trick is to select the next larger device during the fitting process and check the `*.fit` (Fitter Report) file to see the number of PLA P-Terms used.

The fitter report will show resources used in sections describing the entire part, fast modules, and logic blocks. Although this is a much better estimate than the *.ph1 file can give, it has its flaws. The number of P-Terms reported in the larger device may be more than what is actually required since some P-Terms may be replicated in other logic blocks to maintain a fast design. This also depends on whether the designer has specified to keep a particular pin-out.

Once it has been roughly determined how many P-Terms are required, the Max P-Terms per Equation setting can be adjusted. One of the most common mistakes a designer makes is to change the setting to be as large as possible thinking that a smaller number is more restrictive and a larger number will "open up" the resources. This is furthest from the truth. As discussed earlier, a large number runs the risk of allowing more P-Terms than the logic block can support. Additionally, a setting too small will tend to use more macrocells than are available since nodes are generated. It is suggested to use a number in the middle of the range and work from there. Most designs will fit with this setting near the middle of allowable values. In order for a design consisting of many P-Terms to fit, there must be a compromise between speed (large setting) and device utilization (small setting).

Logic Block Fan-in

Another area of concern where design fitting is crucial is the ability of signals to reach the logic block via fan-in. It is possible to have a design that uses all fan-ins available to the logic block. By default, the maximum allowable fan-in is 36. But as mentioned earlier, there are 40 possible fan-ins to the logic block. To access these additional four fan-in signals, simultaneously press <CTRL><ALT>Z while in the XPLA Professional main GUI. The two settings for "Synthesis" and "Fitter" were discussed earlier.

To get a design to fit with fan-in issues the designer can try to incrementally increase the "Fitter" setting until the design fits. This, however, is not a recommended method since last minute design changes may not fit. The available resources from the ZIA may be depleted for a particular signal if there are fewer than four available fan-ins.

Another method that will work occasionally without sacrificing the four additional fan-ins is to reduce the "Synthesis" setting somewhere near 15 but keeping the "Fitter" setting at 36. A smaller setting is often counterintuitive to the designer. However, this will cause nodes to be created by the compiler since the fan-in will be limited for a particular equation which will be eventually placed in a logic block. As a by-product, this breaks up the fan-in for that equation allowing other signals to enter the logic block for the other macrocells. The fan-in that was removed from the logic block for that equation will then reside in another logic block with more available fan-in as a node once the fitter has been run. The fitter still has all 36 fan-ins available in the logic block and therefore will have a greater chance of making a design fit since each equation will specify less fan-in.

It is possible to change the "Synthesis" setting to be greater than the "Fitter" setting, but this does not make sense. If the equations are permitted to have more fan-in signals than the logic block is, then those equations will never fit.

To determine how much fan in is required is, again, a little tricky. Using the `*.ph1` file will reveal the fan-in required per equation, but does not indicate the amount of signal sharing that the logic block will implement. Therefore, an absolute value cannot be obtained. Generally speaking, a signal with large fan-in is one where the fan-in is near 15. These equations can be broken into smaller pieces with the "Synthesis" setting set to a number less than what is indicated in the `*.ph1` file. The fitter report will show fan-in used in a section entitled "Partition/ Routing Summary" after a successful fit. Therefore, selecting the next larger device will tell the designer what fan-in is needed as long as the design fit to that part. Again, this number may be a little misleading since some signals will be used in other logic blocks to implement the design and, of course, will not show any number greater than 40. However, this will give a feel for how much fan-in is required.

Keep in mind that adjusting the Max. P-Terms per Equation setting will affect signal fan-in. This requires the designer to adjust both this setting and the fan-in settings to obtain the desired fit.

Macrocells

Often the design requires too many macrocells. Using a combination of the above methods, in particular increasing the Max P-Term per Equation adjustment, will many times reduce the number of macrocells used by the design. This is applicable as long as there are macrocells that are nodes. If all macrocells are I/Os (i.e. no nodes), then it is impossible to further reduce the macrocell count without removing equations. It is always recommended to increase the Max P-Term per Equation parameter first to reduce the number of macrocells used. Secondly, the "Synthesis" setting should be adjusted for Max Signal Fan-in if the first does not quite work. As an absolute last resort, the "Fitter" setting should be incrementally increased until the design fits. Again, the design change may not fit in the future with when the "Fitter" setting has been previously adjusted to a larger value than 36.

Conclusion

This application note describes the methodology of fitting designs to CoolRunner devices. It briefly reviews the XPLA1, XPLA Enhanced, and XPLA2 devices which was necessary for background knowledge. The individual controls available to the designer from within XPLA Designer is also discussed so that the user can become more familiar with the software. CoolRunner devices have an exceptional ability to fit designs with fixed pin-outs when a last minute design change has been implemented. This is mostly due to the PAL and PLA architecture and partly due to the ZIA routing. CoolRunner CPLDs should be the programmable logic device of choice for this reason.

Revision History

Date	Version #	Revision
11.24.99	1.0	Initial release.