Floorplanning ProASIC®/ProASIC^PLUS Devices for Increased Performance

Introduction to Floorplanning

This application note provides tips and techniques for floorplanning ProASIC and ProASIC^PLUS devices using Designer. Through floorplanning, you can constrain the placement of important modules in your design to help you meet your performance goals.

This application note begins with a terminology overview, illustrates the floorplanning flow for both ProASIC and ProASIC^PLUS device families, and provides several techniques that can help you floorplan your designs effectively. It concludes with a step-by-step example that uses the Designer's ChipPlanner tool to floorplan an actual design for the ProASIC and ProASIC^PLUS device families.

What is Floorplanning?

Floorplanning is a process where you can manually refine the placement of major modules in your design to meet your performance goals. You can perform this process either once or iteratively if your initial layout does not meet timing constraints. In the iterative flow, use floorplanning to refine the placement of a layout so that it is more likely to meet constraints. Please refer to the "Floorplanning Tips and Guidelines" section on page 4 in this application note for more details.

Why is Floorplanning Useful?

Floorplanning guides the place-and-route process to compact your placement into user defined areas of the device so that the performance of certain modules in your design are preserved. Floorplanning allows your design's logical hierarchy to match the actual placement on the device.

Floorplanning may not be useful for all designs. Guidelines for choosing design(s) suitable for floorplanning are discussed in the "General Guidelines" section on page 4.

Supported Device Families

Designer currently supports floorplanning for the Actel ProASIC, ProASIC^PLUS, and Axcelerator device families. The Actel floorplanning tool, ChipPlanner, is included in Actel's Designer release 5.0 and higher as part of the MultiView Navigator.

Terminology Overview

If you are new to floorplanning, it is important to understand certain terms and definitions first because they are used frequently during the floorplanning process. If you are already familiar with floorplanning, we suggest that you briefly review this section.

What are Regions?

A user-defined area located on the device is called a region. Through floorplanning you can control the placement of logic in these regions. The ChipPlanner supports several types of regions such as: logic regions, empty regions, overlapping regions, and regions with net assignments. For additional information about how to create, assign, or edit logic regions, please consult the "About floorplanning" section in Designer's online help.

Logic Region

A Logic region is a region that has logic assigned to it. Logic may include core logic, memory, and I/O signals. The place-and-route tool will place all the logic assigned to a logic region inside that region. The floorplanning process usually requires you to create several regions and assign logic to them.

Empty Region

If you want to prevent logic from being placed within a predefined area in the device, you can create an empty region. The place-and-route tool will not place any logic within an empty region; however, the routing resources within the region can be used.

Overlapping Regions

If you create two or more logic regions whose areas intersect with each other, these regions are defined to be overlapping. The place-and-route tool will detect the area where these regions intersect and will try to place logic common to both of them within this area.

Exclusive/Inclusive Regions

Logic Regions can either be inclusive or exclusive. If a logic region is exclusive, it means that the placement tool cannot place any logic within the region other than what you have previously assigned to it. If a Logic region is inclusive, the place-and-route tool can place any logic within the region even if it was not assigned to it. Exclusive regions are not supported for the ProASIC/ProASIC^PLUS family.

Assigning Nets to a Region

When assigning a net to a region, all of the logic driven by that net will be assigned to that region. Assigning nets to a region allows you to control the net delays of logic connected to certain nets in the design. You can adjust the size of the region to pack logic more closely together, hence, improving its net delays.

ProASIC/ProASIC^PLUS Useability Flows

- Figure 1 describes the ProASIC/ProASIC^PLUS floorplanning flows. Dark shaded items represent actions you need to perform manually while light shaded items are actions automatically performed by the Designer software. Each step of the usability flow will be discussed in more detail below.

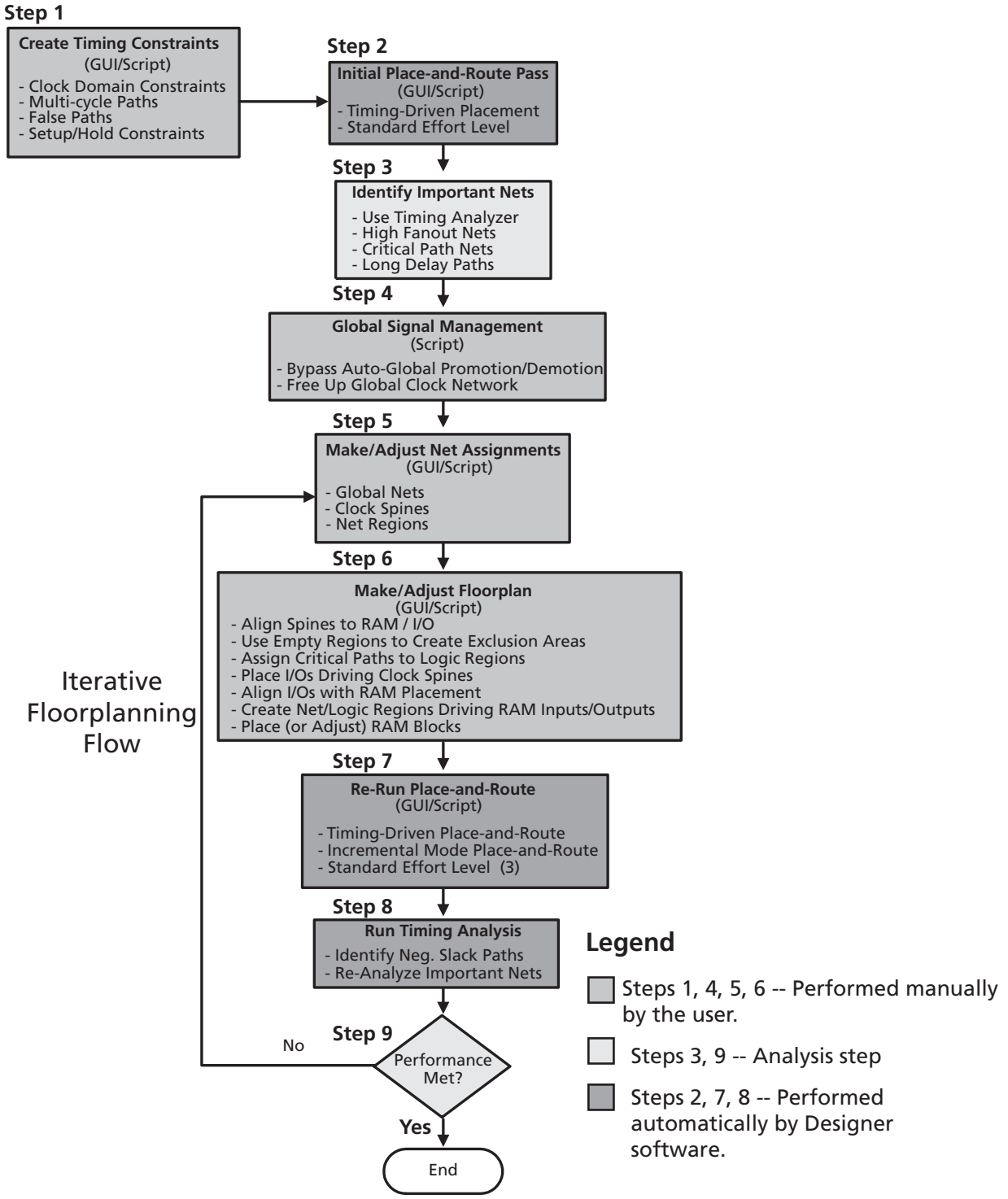

Figure 1 • ProASIC/ProASIC^PLUS Floorplanning Flow

Floorplanning Tips and Guidelines

The following sections describe guidelines for effective floorplanning. They are organized into the following major sub-sections: "General Guidelines" and "ProASIC/ProASICPLUS Guidelines" section on page 6.

General Guidelines

Before you try to floorplan a design, it is important that you understand the architecture of the target FPGA device. Consult the *ProASIC 500K Family* or the *ProASIC^PLUS Flash Family FPGAs* datasheet for more details. Some examples for architectural details you should be aware of are listed below.

Determine if Special Purpose I/Os are Used in Your Design

Are any special purpose I/Os (such as LVPECL pins) being used in the design? Are they located on specific pins or I/O banks? Using special purpose I/Os may restrict your pin placement. If your pin placement is not optimal, I/Os might be placed too far from the logic that interfaces with them. This may violate your timing constraints.

Understand Clock Distribution in the ProASIC/ProASIC^PLUS Architecture

You should understand the clock tree available on the target FPGA and the global and regional routing networks available. Inaccurate floorplanning of clock nets and other high fanout nets may significantly reduce your design's performance.

Understand Memory Distribution in the ProASIC/ProASIC^PLUS Architecture

If your design contains memory, understand how it is distributed on the device (for example, is it placed in a specific location such as the top or bottom of the device, or is it distributed in rows or columns). This will guide your placement if you need to cascade memory or place the logic that is connected to it.

Choose Your Design and Constraints Carefully before Floorplanning

For some designs, no amount of floorplanning can improve their performance compared to push-button place-and-route. Floorplanning works best if a design contains regular structures, which can be partitioned into blocks such that the connectivity of logic in these blocks is localized within the block. For example, designs that contain arithmetic functions, counter, MUXes or datapaths can be floorplanned efficiently. Designs that contain many state machines, random or glue logic may be difficult to floorplan and are better left to push-button place-and-route. If your design has a mixture of regular and non-regular blocks, floorplan the regular blocks and leave the non-regular blocks for place-and-route.

If you already have timing constraints, you may need to adjust your floorplan so that it does not conflict with them. Run the design through place-and-route and use the Timer tool to analyze the paths that fail to meet your timing constraints. If you have floorplanned any of the logic included in these paths, (for example, assigned them in regions, or made direct placement assignments) you may need to remove those constraints so that place-and-route can optimize the entire design.

Analyze the Connectivity in Your Design

The Compilation Report in Designer displays nets with the highest fanout in the design. Designer will display the top 10 nets that have the largest fanout. This information will help you to floorplan the design effectively. For example, you could prioritize nets such that the ones with largest fanout are assigned to global routing (clock) networks. After these global resources are used up, you can assign the remaining nets to regions or clock spines. Please refer to the "ProASIC/ProASICPLUS Guidelines" section on page 6 for more details.

Identify the Critical Paths in Your Design Before Floorplanning

Before creating a floorplan, it is very important to be aware of and identify the critical paths in your design. A path that you have defined to have a specific timing constraint or that violates your timing constraints is considered to be a critical path. The logic on your critical paths must be placed closer together to meet your timing constraints. You can customize your floorplan to include these critical paths into region(s) and resize your region to improve their net delays. For example, use the Timer tool to identify a path that violates your timing constraints in the timing report. Use ChipPlanner to assign logic

included within the path to a region. Refer to Figure 4 on page 10 and Figure 5 on page 10 for an example of this. Before trying this, make sure that the critical path on which you are working does not share too much logic with other critical paths; otherwise, this process could increase the net delays of other critical paths.

Verify that Logic You Assigned to the Region(s) Fits

When you create logic regions, be sure the logic assigned to them will fit into them. Place-and-route will generate errors if the logic assigned to your region exceeds its capacity. If you size the region too tight, place-and-route may not have enough room to route the logic within the region and may fail because of routing congestion. Therefore, it is a good practice to oversize your region(s) by approximately 10-20% to provide place-and-route enough room to route the assigned logic.

The ChipPlanner tool gives you the ability to determine the size of a region based on the amount of resources a logic block needs. Hold your mouse at the end of the region and a tool-tip will show you the amount of logic resources are available. If you place the mouse cursor on top of the logic block in the Logical Hierarchy window, a tool-tip will show you the amount of resources that logic block will need if it were to be assigned to a region. The ChipPlanner also includes a Region Properties dialog box that can help you size your regions. This tool shows how full your region is (in percent capacity) based on the logic you have assigned to it. The capacity estimate is broken down for each logic resource type you have assigned to the region. Figure 2 is an example of the Region Properties dialog box.

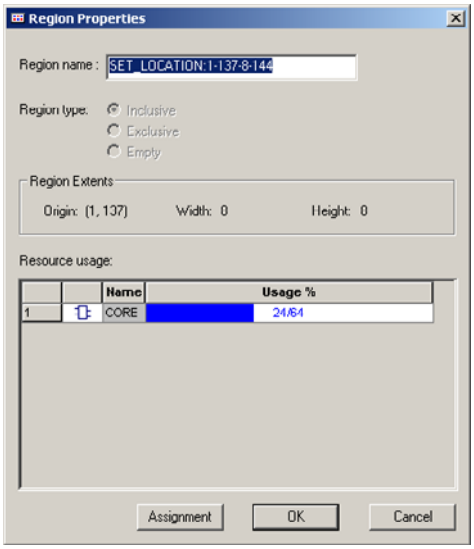

Figure 2 • Region Properties Dialog Box

Provide Realistic Performance and Density Goals for Your Design

Your design constraints should have adequate margin so that the iterative floorplanning process has a chance to work. If your timing constraints are too tight, the following things could happen that may reduce your design's performance:

1. Floorplanning could be redundant. The place-and-route tool is already working to optimize your logic placement to minimize delays and meet your timing constraints.
2. Floorplanning may conflict with the place-and-route process by over-constraining it.
3. Floorplanning has no chance to work, because it is impossible to meet your timing constraints for certain paths in the design, even if they are placed as close together as possible. (For example, the gate delays within a path exceed the constraint delay associated with that path).
4. Congestion for local routing resources occurs and forces the router to use sub-optimal paths to connect logic.

Instantiate and Floorplan ACTgen Memory Macros

It is preferable to use ACTgen created macros to instantiate memory in your design. ACTgen macros are customized to support all of the features unique to the architecture of your target device. ACTgen provides an interactive GUI interface that allows you to customize the features of the memory blocks you want to use and generates an optimized structural netlist for the memory blocks. Analyze how your I/O logic and routing resources such as, globals net or clock spines, interface with your memory blocks. This is necessary to floorplan your memory properly on the device. Normally, place-and-route will do this automatically for you, but if your design has tight timing requirements, it might be necessary to place your RAM blocks manually. For more information about using ACTgen please consult the *ACTgen User's Guide*.

ProASIC/ProASIC^PLUS Guidelines

The ProASIC/ProASIC^PLUS guidelines listed below illustrate in detail the nine steps performed in the ProASIC/ProASIC^PLUS usability flow shown in Figure 1 on page 3. Under each step detailed guidelines are given with the following exceptions:

- A detailed discussion on how to specify, determine, and create timing constraints is beyond the scope of the application note.
- A detailed overview of how to run place-and-route and timing analysis on the design is also outside the scope of this application note.

You can refer to online help in Designer for more information on how to create timing constraints and run place-and-route or refer to the *ProASIC/ProASIC^PLUS Timing Constraints* application note.

Create Timing Constraints – Step 1

Review your design and make appropriate timing assignments. Be sure your assignments are covering all of the critical paths in your design including paths that propagate through your memory elements. Identify non-critical paths and create multi-cycle or false paths for them. These assignments will reduce the number of violations listed in your timing reports and accelerate timing closure. Be aware of the timing requirements between the FPGA and other devices on your board. Create external setup and hold constraints to account for these timing requirements.

Run Initial Place-and-Route Pass (and Timing Analysis) – Step 2

Before trying to floorplan your design, it is a good idea to first lace-and-route. It is possible that most (or all) of your timing constraints are already met after the first placement, making floorplanning unnecessary. Otherwise, this step may identify additional critical paths and nets you were not aware of that will need further constraining to meet your timing requirements. The performance and placement you get from this step also serves as an initial baseline for improvement through floorplanning.

Identify Important Nets – Step 3

There are three types of nets that you might need to assign to either local routing or region resources. These are high fanout, critical path, or long delay nets. Guidelines for identifying each of these net types are discussed below.

High Fanout Nets

Based on the hierarchy and connectivity of your design, you might already be aware of certain high fanout nets such as global clocks, resets, and bus control signals that drive many logic instances. After completing Step 2, review Designer's Compilation report to see a listing of the top 10 highest fanout nets in the design. Based on this list and nets you are already aware of, you will need to prioritize and assign them to the appropriate routing resources available in the device. This will be discussed in more detail in Step 5.

Critical Path Nets

All nets within your Critical paths must be kept as short as possible to meet your timing requirements. After completing place-and-route, run timing analysis using your constraints to report timing violations. Analyze the paths that are violating your timing constraints and determine if specific net delays between two or more logic instances are contributing to many timing violations. These nets should be added to a list of critical path nets that may require floorplanning.

Long Delay Nets

These nets are found during timing analysis. If a path is failing and a net delay between two or more instances in the path is contributing to most of the timing delay within the path, then this net should be subject to floorplanning. Repeat this process for all timing paths that are violating your constraints to identify all of your Long Delay Nets.

Manage Global Signals – Step 4

When Designer is compiling your design, the nets within it are analyzed by fanout and connectivity. Based on this analysis, Designer will automatically place certain nets on global clock networks. This process is called Auto-Global Promotion/Demotion. Unfortunately, Automatic Net Promotion/Demotion might not reflect the best net assignments for the design and may actually reduce overall performance. To prevent this, it may be necessary to force Designer to place certain nets on the global routing networks or clock spines you choose. You can use certain GCF commands to force Designer to place nets where you want them. Some of these commands are discussed below.

Preventing Auto-Global Promotion/Demotion

ProASIC/ProASIC^PLUS has four global clock networks. Designer provides you the flexibility to do the following:

1. Prevent automatic global net promotion.
2. Prevent certain nets from being assigned to global networks.
3. Prevent automatic promotion of the highest fanout nets (fanout >32).

These are performed by the following commands.

```
set_auto_global <0 - 4>
 // If 0, prevent automatic global net promotion.


set_noglobal <net_name1, net_name2,… net_nameN>
 // Prevents the net_name1 through net_nameN from being assigned to global nets.


dont_fix_globals //Prevent automatic fanout promotion to globals
```

Bypassing Auto-Global Promotion/Demotion is only available in the script mode in Designer and the above commands must be included in a GCF file that is imported with the design's netlist.

If your design contains, PLLs or LVPECL I/Os, the above commands will have no effect. PLL outputs or LVPECL I/Os automatically drive the global clock networks. This is required by the ProASIC/ProASIC^PLUS architecture. If you have any remaining global clock networks, you can use `set_global` or `use_global` as described below, assign them.

Freeing up Global Clock Networks

Use the `set_auto_global` and `dont_fix_global` commands to free up your global clock networks. This may be necessary to accommodate specific nets you want to promote to the global network or to make clock spine assignments. You must free up enough global clock networks to accommodate your clock spine assignments; otherwise, place-and-route will generate errors when it processes your GCF constraints.

Make/Adjust Net Assignments – Step 5

Review the high fanout, critical path, and long delay nets in the design. For the high fanout nets, prioritize them based on their fanout and assign them to global routing, clock-spines, or to regions.

Making Global Net Assignments

Determine which of your high fanout nets has a scope that reaches most of your design's modules. Place these nets on one of the four global clock networks on the ProASIC/ProASIC^PLUS device. Be careful not to use up all of your global networks, in case you need to make spine assignments for your remaining nets. Include the following commands in your GCF file (ChipPlanner GUI is not supported) to make global net assignments.

```
set_global <net_name1, .. net_nameN>
// Allow net_name1 through net_nameN to be assigned to the global clock network.
```

Making Clock Spine Assignments

If either the fanout of some of your nets reaches several modules in the design, or you have critical path and/or long delay nets, then try assigning them to clock spines.

It is recommended that you free up at least one global network before using clock spines in the ProASIC/ProASIC^PLUS device. However, if your design requires all global networks to be used, you could still use clock spines under certain conditions. Please refer to the "Using Spines in Occupied Global Networks" section of the *Optimal Usage of Global network spines in ProASIC^PLUS Device* application note for further details.

Include the following commands in your GCF file (ChipPlanner GUI is not supported) to make Spine assignments.

```
use_global <spine_name> <net_name>
// Allows you to assign a net to a specific clock spine.
```

Place I/O Driving Spines Near the Middle of the Device

If your clock spines are driven by nets from I/O pins, you should place them near the middle of the device. All clock spine drivers in the ProASIC^PLUS device are located in the middle of the device on the east and west sides. Placing your I/Os in this location will put them close to the clock spine driver. This minimizes the net delay between the I/O pin and clock spine driver, which adds to the overall insertion delay of the clock spine net.

Create Regions for Clock Spines

Each clock spine spans a fixed number of core cells depending of the device type you have chosen in the ProASIC/ProASIC^PLUS family. These core cells occupy a fixed area on the device called a clock-spine region. If you have determined that the logic spanned by your net will easily fit within a clock spine region, then create a logic region that overlaps and is smaller than the clock-spine region. This logic region helps to cluster the logic into a smaller area within the clock spine region. This might improve the timing of logic driven by your clock spine net. Figure 3 on page 9 shows an example of a clock spine and overlapping region.

Making Regions for Nets

Constraining nets to a region helps to control the connection delays from the net's driver to the logic instances it fans out to. This can be used for high fanout or critical path nets or bus control logic. You can create regions for nets either within the ChipPlanner GUI or in the script mode using a GCF file.

Using Regions for Critical Path and High Fanout Nets

You should assign high fanout or critical path nets to a region only after you have used up your global routing and clock spine networks. If you have determined, through timing analysis, that certain long delay nets are creating timing violations, assign them to regions to reduce their delays.

Before creating your region, determine if any logic connected to instances spanned by these nets have any timing requirements. Your region could alter the placement of all logic assigned to it. This may have an undesired side effect of altering the timing delays of some logic paths that cross through the region but are not assigned to it. These paths could fail your timing constraints depending on which net delays have been altered. Figure 4 on page 10 shows the timer path placed on the ChipPlanner. Figure 5 on page 10

shows a critical path identified by timer. The previous placement of the critical path logic is highlighted. Note from the placement shown that there are two logic instances (ix2725 and ix2711) that are placed two core tile blocks away from the rest of the logic in the path. Figure 4 on page 10 shows a recommended region to place all of the logic in the path.

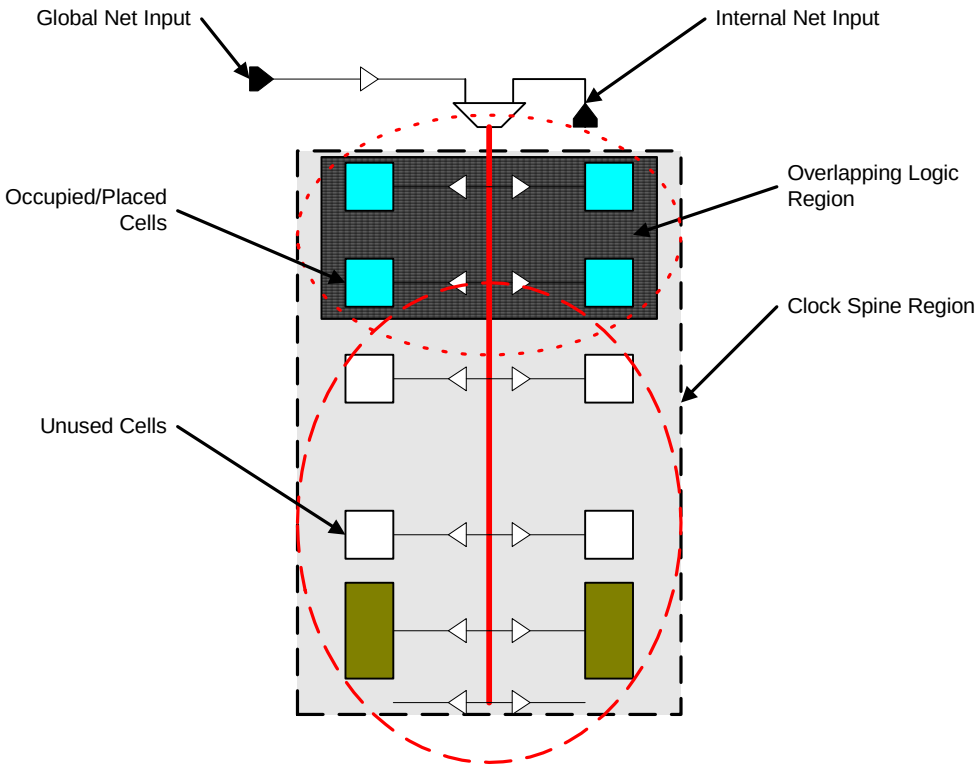

Figure 3 • Clock Spine Region

Using Regions to Localize Placement of I/O Bus Control Nets

If your design has wide busses that connect to many modules throughout the design, it is important to localize placement of nets or logic that controls the I/O bus, such as tristate enable. The tristate enable signal should be placed as close to the I/O bus as possible to minimize the turn-around time of the bus. Create a region that includes both the bus pins and the output enable nets. This will place bus control logic close to the bus I/O signals.

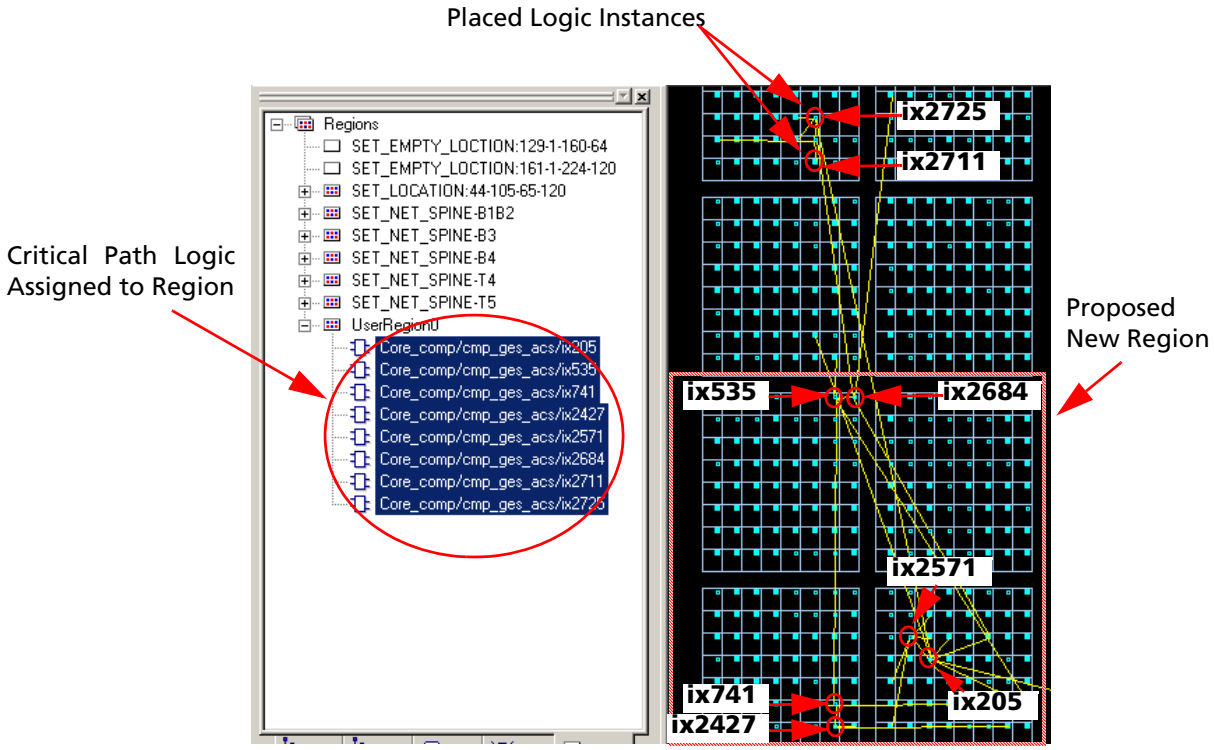

Figure 4 • Critical Path Displayed in Floorplan and Assigned to a Region

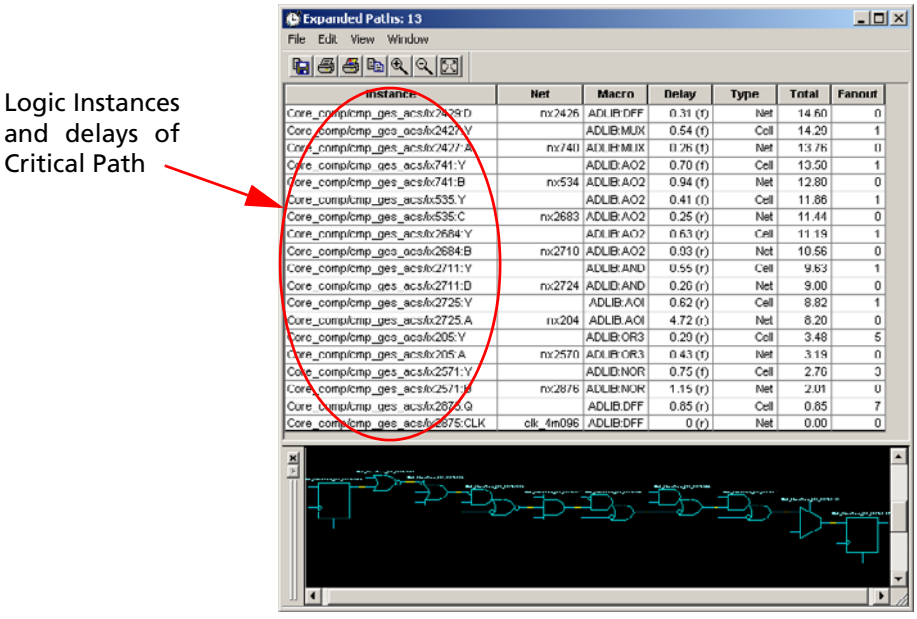

Figure 5 • Path Violating Slack in Timer – Critical Path

Make/Adjust Floorplan – Step 6

Align RAM Placement to Coincide with the Scope of Your Clock Spine Assignments

If you plan to use clock-spines to connect your memory's inputs, be careful which spine(s) you choose. While the clock spine network does extend into the memory array at the top and bottom of the ProASIC/ProASIC^PLUS device, it only connects to certain RAM blocks. If you have placed your RAM in a location that does not connect with your clock spine net, the router may demote your spine assignment and use regular routing resources to connect to the RAM. This may increase the net delay for logic that drives the memory inputs. Please refer the "Spines Driving SRAM Block" section of the *Optimal Usage of Global Network Spines in ProASIC^PLUS Devices* application note and for more details.

Using Empty Regions

Empty regions allow you to create exclusion areas on the device where no logic placement can occur. Empty regions help guide the ProASIC/ProASIC^PLUS placer to pack your logic closer together and thereby use more local routing resources to connect it. Use the following guidelines for empty regions.

Use Empty Regions to Guide the Place-and-Route Process

If your design does not completely use up your target device (for example 60% utilization or lower), use empty regions to cluster your logic placement into specific sub-area(s) of the chip. This helps when you have originally placed-and-routed the design into a smaller device but want to fit it to a larger part while still preserving the performance you have achieved in the smaller device.

Use Empty Regions to Reduce Routing Congestion

Creating empty regions next to the congested area(s) of your design helps reduce congestion. When you place an empty region next to congested logic blocks or regions, the placer cannot place any logic next to your region or logic block. Logic, which would normally be placed there, is forced to be placed somewhere else. Routing resources next to the congested area are, therefore, freed up and provide the router more options to route signals into the congested block. In Figure 6 on page 12, the right side shows a logic region connecting to two congested logic blocks. The left side of the figure shows the effect of placing two empty regions labeled A and B on the top and bottom of the logic regions in the congestion areas.

Before deciding to place empty region(s) in this manner, analyze your design for congestion areas. Use the "ratsnest" view in the ChipPlanner tool to see dense areas of connectivity into and out of your logic blocks or regions. Create empty regions in these congested areas and see it if improves the routability of your logic.

Use Empty Regions to Reserve Device Resources

If you want to preserve the placement of your existing design but plan additional modifications in the future, create empty regions in the areas of the chip where you plan to add additional logic. As you add new logic remove or resize your empty regions accordingly to fit your new logic. Empty regions placed over I/O pins reserve them for future use as the I/O needs of your design changes. There are some restrictions for using empty regions in this manner. Please refer to the "Floorplanning Caveats" section on page 20 for more details.

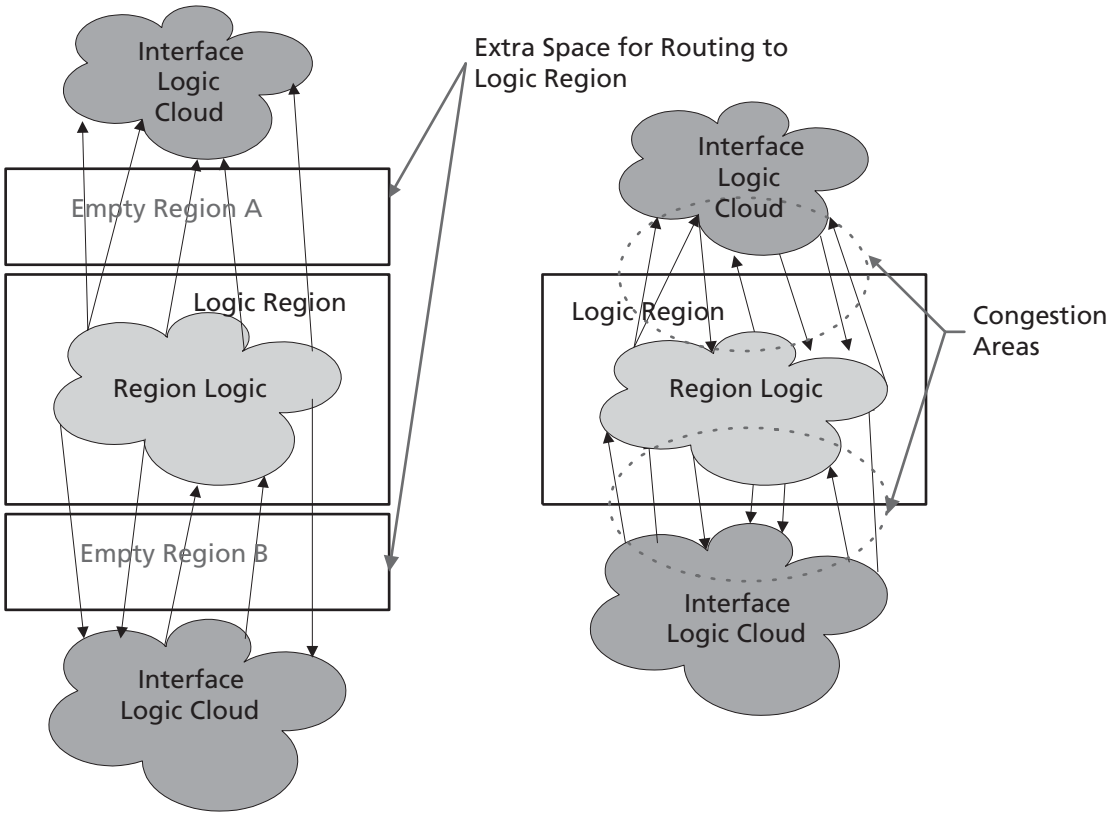

Figure 6 • Using Empty Regions to Reduce Congestion

Using Logic Regions

Use logic regions to compact the placement of certain logic blocks in your design. This allows you to control logic placement at the region or block level. This may simplify your floorplanning task, since you might not have to place logic instances individually on the device. The following sections contain some guidelines for using logic regions.

Use Logic Regions to Localize Placement of Logic Blocks

If you have partitioned your design into several modules and some of these modules contain regular structures such as, arithmetic logic, register arrays, counters or multiplexors, place these modules into Logic regions. These logic functions have a good amount of both local connectivity and regularity to their structure, which makes them good candidate for regions. Interconnects between your region now become interconnects between hierarchical blocks in your design. Floorplan your regions such that there is a smooth horizontal or vertically oriented data flow between each of your logic regions.

For Pipelined Logic, Place Registers on Region Boundaries

If you have assigned logic to a region such that its inputs and outputs are bounded by a register array (pipeline registers), it is a good idea to place these pipeline registers close to the boundary of the region. If you plan to manually fix the placement of your pipeline registers, make sure to orient them in the correct direction to assure a smooth data flow between them and their interfacing logic. The bottom portion of Figure 7 shows the possible placement for pipelined logic arbitrarily assigned to a region where the input and output pipeline registers connect to logic blocks on the left and right side of the device respectively. Note that certain paths feeding into and out of the pipeline registers are longer than others. When the placement is revised with the pipeline registers being placed near the boundary of the region (as shown in the top portion of Figure 7), the net delay between logic interfacing to these registers is equalized. The net delays of logic feeding and out of your pipeline registers is more uniform. This may help to meet your timing requirements.

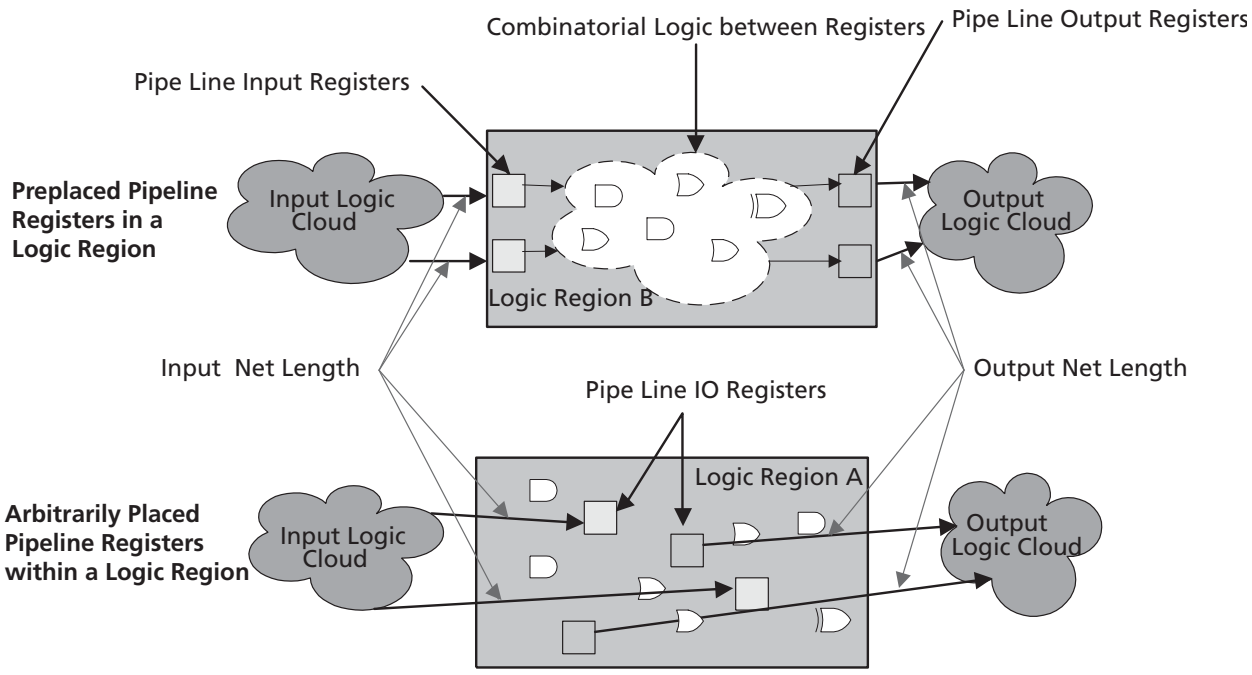

Figure 7 • Assigning Pipelined logic to Regions

Aligning RAM I/O with Placement

Before placing your memory blocks, review your design and understand how data is flowing into and out of them. Determine what logic blocks are driving the memory inputs (for example, address line, control signals) and what logic is driven by the memory outputs (for example, databus lines). Some guidelines to follow are:

1. Place pins that drive or are driven by your memory blocks close to where your memory blocks are placed.
2. Create an empty region next to your memory block to free up local routing resources that may need to be used to connect to the memory blocks.
3. If you are driving high fanin memory inputs such as read/write clocks or read/write enables, try using low-skew routing resources such as global nets or clock spines to connect them. Make sure that your clock spine assignments are aligned with your RAM placement. Please Refer to the "Align RAM Placement to Coincide with the Scope of Your Clock Spine Assignments" section on page 11 for details.

Rerun Place-and-Route – Step 7

After you have completed your first floorplan, commit your changes and rerun place-and-route with the following settings: Timing-Driven Mode and Incremental Place-and-Route.

If some of your floorplanning changes inadvertently affect many logic instances in the design, Designer will warn you that incremental place-and-route will not be performed; instead, a complete replacement and route of the design will occur.

Run Timing Analysis – Step 8

Rerun Timing analysis and analyze the performance of your design. Determine if floorplanning improves the timing of the critical path you initially discovered during your first Timing Analysis run in step 2. If different critical paths appeared, make a note of them so that you can update your floorplan to include them.

Repeat – Step 9

Floorplanning is an iterative process. If the initial floorplan you completed in step 6 does not remove most of your timing violations, you will need to repeat steps 5 through 8 and adjust your floorplan accordingly to resolve them. Your goal in each floorplan iteration should be to progressively remove more timing violations from your design until you are left with an optimized placement that meets all of your timing constraints. This is largely a design-dependent trial and error process and might take you several iterations to achieve your design goals.

Floorplanning Example

ChipPlanner GUI Example

The following design example illustrates many of the floorplanning techniques discussed above.

The first step was to make timing assignment to the specific clocks and I/Os in the design. As shown in the GCF file, clkA and clkB were constrained to 40 and 20 MHz respectively using the create clock command (Figure 11 on page 17). Figure 8 shows the fanout of these nets in a previously placed-and-routed version of the design. Note that clkA connects to most of the top and bottom ram tiles in the device, while clkB connect to core logic distributed across the chip. The reset signal also connects to many registers throughout the core of the chip. Based on this connectivety profile, these signals were assigned to three of the four global routing networks on the device. This design has specific requirements for the setup time of certain I/O signals driving the core. These are specified within the GCF file as `set_input_register_delay` commands. The minimum propagation delay of certain critical nets in the address bus is specified using the `set_max_path_delay` command.

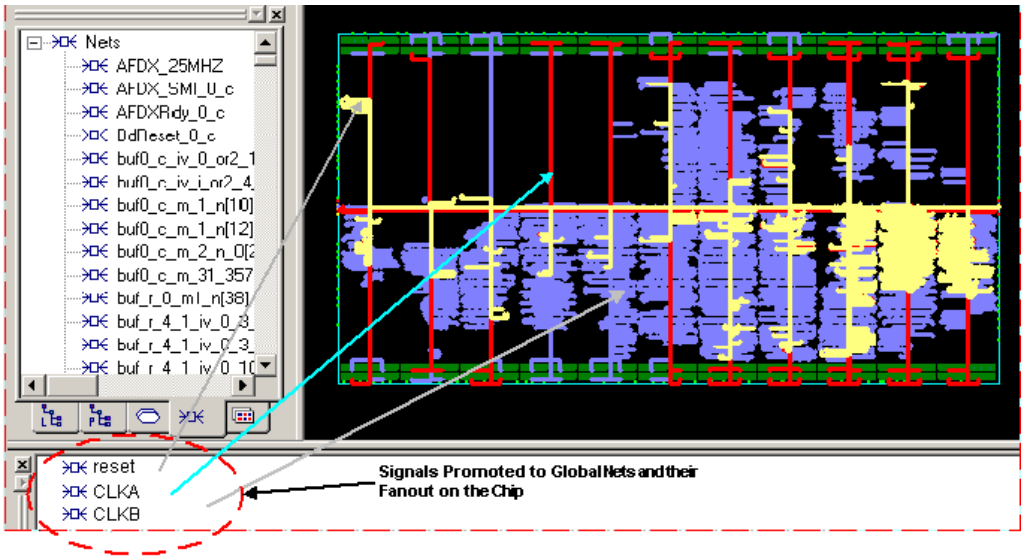

Figure 8 • Display of Highest Fanout Nets

After clkA, clkB, and reset were assigned to the global network, one network was available for clock spine assignments to clocks clkA1, clkA2, clkB1, and clkB2 shown in Figure 11 on page 17 in Step 6. Figure 9 on page 16 shows the clock spine regions assigned each of these clocks.

This design uses almost 100% of the RAM resources in the device. Therefore, there was a high possibility of routing congestion that may occur between the RAM block and the logic that interfaces with it. To alleviate this congestion, an empty region was placed next to the top RAM tile block on the chip. Figure 10 on page 16 shows the placement of this empty region.

As mentioned earlier, to meet the setup time on certain I/O pins, their driving logic needs to be placed as close to the pins as possible; therefore, logic was assigned to the critical path region (shown in Figure 10 on page 16).

Figure 12 on page 18 and Figure 13 on page 19 respectively, show the performance results for this design with and without the floorplanning and timing constraints specified above. As you can see, the careful floorplanning of this design paid off by significantly improving the performance for all six clocks in the design. In particular, the performance of clkA, which has the tightest timing, was more than doubled.

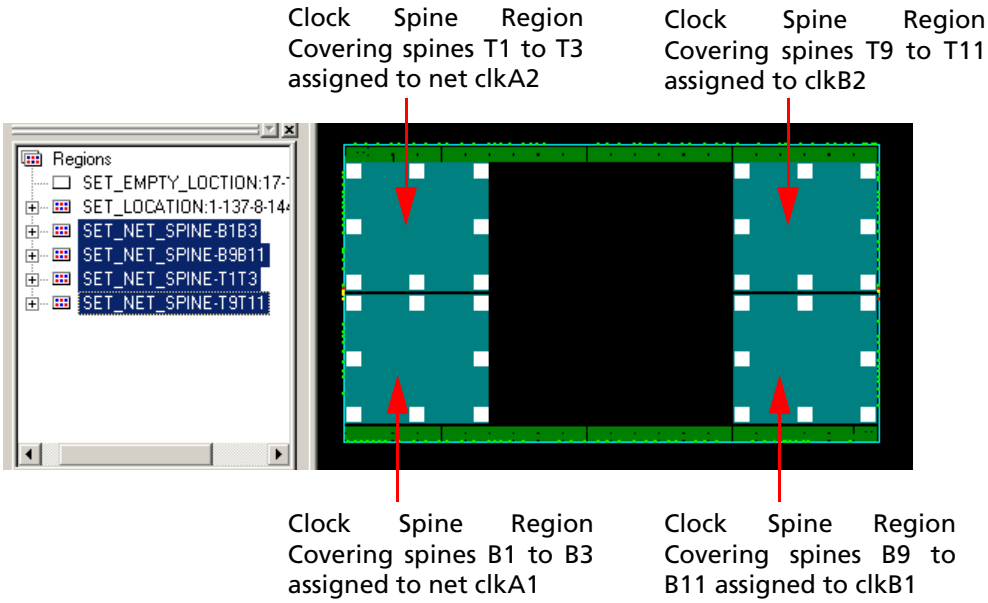

Figure 9 • Display of Clock Spine Regions for Four Non-Global Clocks in Design

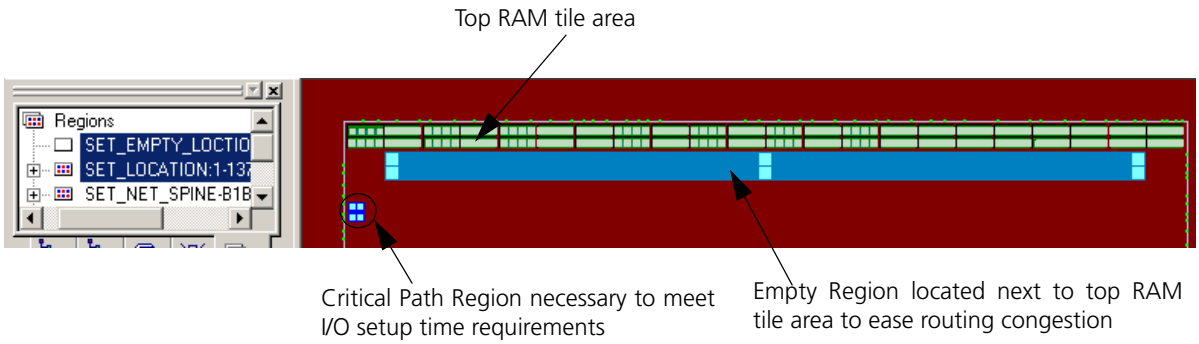

Figure 10 • Display of Empty and Critical Path Regions

// Put timing constraint on the two main clocks of the design
create_clock -period 40.0 clkA;
create_clock -period 20.0 clkB;
 	
// These 6 input pin need input time of 6 ns, with 2.5 ns clock net delay, we can go to 8.5 ns
set_max_path_delay 6 "ADDRBUS_PAD[4].Y", "ADDRBUS_PAD[5].Y", "ADDRBUS_PAD[6].Y", "ADDRBUS_PAD[7].Y", "dly1_0.D", "dly1p_0.D", "dly1w_0.D" ;
set_input_to_register_delay 6 -from TS_0 ;
set_input_to_register_delay 6 -from TA_0 ;
 	
// These 6 input pin need input time of 6 ns, with 2.5 ns clock net delay, we can go to 8.5 ns
set_max_path_delay 6 "ADDRBUS_PAD[4].Y", "ADDRBUS_PAD[5].Y", "ADDRBUS_PAD[6].Y", "ADDRBUS_PAD[7].Y", "dly1_0.D", "dly1p_0.D", "dly1w_0.D" ;
set_input_to_register_delay 6 -from TS_0 ;
set_input_to_register_delay 6 -from TA_0 ;
 	
//set false path through from DMA datareq and addr through shared RAM
set_false_path -from U_MACa/U_MAC/U_DMA/datareq* -through U_SHRAM*/U_ShRAM* ;
set_false_path -from U_MACa/U_MAC/U_DMA/addr* -through U_SHRAM*/U_ShRAM*;
	Step 1 – Assign timing constraints

//set the net criticality high on the interconnecting nets in above
set_critical 5 *cpuaddrm*;
set_critical 5 N_608;
set_critical 5 TS_0_c; // net from input pad
set_critical 5 TA_0_c ; //net from input pad
	Step 3 – Define and assign critical paths

set_auto_global 0; // this prevent automatic promotion to global	**Step 4 – Prevent automatic global signal promotion**

//following nets are going all over the die, so we assign use 3 globals out of 4
set_global clkA ;
set_global clkB ;
set_global Reset;
 	
//Slow Maclk rx and tx clk on spine for low skew net and to avoid hold time violation
use_global B1,B3 clkA1;
use_global T1,T3 clkA2;
 	
//Slow Mbclk rx and tx clk on spine for low skew net and to avoid hold time violation
use_global B9,B11 ClkB1;
use_global T9,T11 CLkB2;
	Step 5 – Global clock assignments free up one global for clock spine assignments

//CPUAddrM* and TA_0 are located close to this region.
//So we keep all the instances in critical path clustered closely in 8x8 = 64 tiles
set_location (1,137 8,144) *statecntrl* ;
set_location (1,137 8,144) *cpuaddrm* ;
set_location (1,137 8,144) *burstaddr*;
set_location (1,137 8,144) *dly1* ;
 	
//keeping the space close to top RAM block empty to ease congestion.
 	
set_empty_location (17,153 336,164) ;
	**Step 6 – Assign critical path instances to a region.
	Empty region created near top RAM tiles on the device.**

Figure 11 • **GCF File Example**

```
ProASIC^PLUS Sample Design - No Floorplanning Results

Timer Version 01.01.01
Actel Corporation - Actel Designer Software Release Designer v5.0,
Copyright (c) 1989-2003
Date: Fri Sep 05 18:36:05 2003

Design: MAC
Family: PA
Die: APA1000
Package: 208 PQFP
Radiation Exposure: 0 KRad
Temperature: COM
Voltage: COM
Speed Grade: STD
Design State: Post-Layout
Timing: Worst Case
Path Tracing: Longest Paths
Break at Clk/G pins: True
Break at Preset/Clr pins: True
Break at Data pins of Latchs: True


Section Clock Frequency
      Actual     Required    ClockName
    21.30MHz     40.00Mhz  clkA
    14.66MHz     20.00Mhz  clkB
    14.03MHz     5.00Mhz   clkA1
    12.11MHz     5.00Mhz   clkA2
    38.97MHz     5.00Mhz   clkB1
    14.40MHz     5.00Mhz   clkB2
End Section
```

Figure 12 • Non-Floorplanned Design

```
Timer Version 01.01.01 - Floorplanning Results
Actel Corporation - Actel Designer Software Release Designer v5.0,
Copyright (c) 1989-2003
Date: Fri Sep 05 21:30:41 2003

Design: MAC
Family: PA
Die: APA1000
Package: 208 PQFP
Radiation Exposure: 0 KRad
Temperature: COM
Voltage: COM
Speed Grade: STD
Design State: Pre-Layout
Timing: Worst Case
Path Tracing: Longest Paths
Break at Clk/G pins: True
Break at Preset/Clr pins: True
Break at Data pins of Latchs: True


Section Clock Frequency
      Actual     Required    ClockName
    43.21MHz     40.00Mhz    clkA
    20.73MHz     20.00Mhz    clkB
    33.33MHz      5.00Mhz    clkA1
    28.24MHz      5.00Mhz    clkA2
    28.07MHz      5.00Mhz    clkB1
    36.50MHz      5.00Mhz    clkB2
End Section
```

Figure 13 • Floorplanned Design

Floorplanning Caveats

Most of the functions of floorplanning, such as assigning logic to regions, are supported in both ProASIC/ProASIC^PLUS device families. However, there are some family-specific limitations, which are summarized in the Comments section of Table 1. Any restriction on a particular feature is also summarized in the Comments section. Features and/or restrictions may change on later releases of Designer.

Table 1 • ProASIC/ProASIC^PLUS Floorplanning Exceptions List

Feature Description	ProASIC/ ProASIC^PLUS Support	Comments
Creating Rectangular Regions	Yes	No Restrictions
Creating Inclusive Regions	Yes	No Restrictions
Creating Exclusive Regions	No	Not Supported in ProASIC^PLUS
Creating Empty Regions	Yes	Empty regions cannot be created over RAM, PLL, and Globals
Creating Overlapping Regions	Yes	
Creating Net Regions	Yes	Applies both to Global and internal nets. No restrictions
Clock Spine Creating Regions	Yes	
Assign Logic to Regions	Yes	
Assign RAM to Regions	No	RAM, PLLs or global signal cannot be assigned to a region currently prevents RAM or PLLs or global signals from being assigned to a Region
Assign I/O to Regions	Yes	Global I/O cannot be assigned to regions
Assign Mixed Logic to Regions (I/O, RAM, NETS, LOGIC)	No	Global IO, RAMs, and PLL cannot be assign to regions
Assign same Logic to Multiple Regions	Yes	The Regions must overlap
Unassign Logic/Nets from Regions	Yes	Regions can be redeclared with different logic. Order or declaration is important. Declarations specified later in a GCF file take precedence over earlier declarations. In the GUI, use the assign/unassign dialog box to update logic or net assignments to region(s)

Summary

This application note summarized general guidelines for floorplanning and documented a process to floorplan designs for the ProASIC/ProASIC^PLUS device families. An example was provided that illustrates the techniques discussed in the application note and showed the performance gain that resulted by using them. The "Floorplanning Caveats" section on page 20 discussed family-specific restrictions you should be aware of when floorplanning for the ProASIC/ProASIC^PLUS device families.

Effective floorplanning is learned through experience by working with many designs. Each design presents its own challenges that make its floorplan unique compared with other designs. The tips and techniques presented in this application note cannot guarantee performance improvements for your design but can give you a set of tools to work with when tackling a floorplanning task for your design.

In some cases, performance cannot be improved through floorplanning alone but requires additional constraints to be applied to the design (for example, timing and/or synthesis constraints). Effective techniques for applying these constraints are beyond the scope of this application note.

Related Documents

Datasheets

ProASIC 500K Family

http://www.actel.com/documents/Proasic_DS.pdf

ProASIC^PLUS Flash Family FPGAs

http://www.actel.com/documents/ProASICPlus_DS.pdf

Application Notes

Optimal Usage of Global network spines in ProASIC^PLUS Device

http://www.actel.com/documents/APA_Spines_AN.pdf

I/O Features in ProASIC^PLUS Flash FPGAs

http://www.actel.com/documents/APA_LVPECL_AN.pdf

Efficient Use of ProASIC Clock Trees

http://www.actel.com/documents/A500K_Clocktree_AN.pdf

ProASIC/ProASIC^PLUS Timing Constraints

http://www.actel.com/documents/Flash_TimConst_AN.pdf

User's Guides

ACTgen User's Guide

http://www.actel.com/documents/genguide_UG.pdf

Designer User's Guide

http://www.actel.com/documents/designer_UG.pdf

http://www.actel.com

Actel Corporation
2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Tel: (650) 318-4200
Fax: (650) 318-4500

Actel Europe Ltd.
Maxfli Court, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
Tel: +44 (0)1276 401450
Fax: +44 (0)1276 401490

Actel Japan
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Tel: +81 03-3445-7671
Fax: +81 03-3445-7668

Actel Hong Kong
39th Floor
One Pacific Place
88 Queensway
Admiralty, Hong Kong
Tel: 852-22735712

51900064-0/11.03