AN 453: HardCopy II ASIC Fitting Techniques

© November 2008 AN-453-2.0

Introduction

Engineers often use a flexible, reprogrammable Stratix® II FPGA for prototyping a project, and then transfer the design to a faster, more economical HardCopy® II ASIC for mass production. This application note describes some possible design considerations and solutions for fitting a Stratix II FPGA design into a HardCopy II ASIC.

HardCopy II ASICs are low-cost, high-performance 1.2-V, 90-nm ASICs with pinouts, densities, and architectures that complement Stratix II FPGAs. HardCopy II ASIC features, such as phase-locked loops (PLLs), memory, and I/O elements (IOEs), are functionally and electrically equivalent to the Stratix II FPGA features. The combination of Stratix II FPGAs for in-system prototype and design verification, HardCopy II ASICs for high-volume production, and the Quartus® II design software, provides a complete, low-risk ASIC solution.

HardCopy II ASIC Compatibility with Stratix II FPGA Prototypes

For the HC210(W), HC220, and HC230 devices, there are three possible Stratix II FPGAs you can choose as the companion FPGA prototyping devices. For example, you can prototype an HC230 design using EP2S90, EP2S130, or EP2S180, depending on the design requirements for logic gates, memory, and digital signal processing (DSP) functions. You prototype the HC240 using an EP2S180.

Table 1 shows Stratix II FPGA to HardCopy II ASIC compatibility.

Table 1. Stratix II FPGA to HardCopy II ASIC Footprint Compatibility

HardCopy II ASICs	Packages	Stratix II Devices				
		EP2S30	EP2S60	EP2S90	EP2S130	EP2S180
HC210W	484-pin FineLine BGA *(1)*	✓	✓	✓ *(2)*	—	—
HC210	484-pin FineLine BGA	✓	✓	✓ *(2)*	—	—
HC220	672-pin FineLine BGA	—	✓	—	—	—
HC220	780-pin FineLine BGA	—	—	✓	✓ *(2)*	—
HC230	1,020-pin FineLine BGA	—	—	✓	✓	✓ *(2)*
HC240	1,020-pin FineLine BGA	—	—	—	—	✓
HC240	1,508-pin FineLine BGA	—	—	—	—	✓

Notes to Table 1:

(1) The HC210W device uses a wire-bond package; the Stratix II FPGA prototype device uses a pin-compatible flip chip package.

(2) Depending on design-specific resource use, this very aggressive die-shrink might be possible. Be sure to confirm whether or not your design is a potential candidate for such an aggressive shrink by fitting it with the Quartus II software and/or consulting an Altera® Applications Engineer.

Each path from a Stratix II FPGA to a HardCopy II ASIC represents a very significant physical die area reduction. The bigger the FPGA you choose as the prototype, the greater the die area reduction. This die area reduction is made possible by the combination of fine-grained logic fabric and the removal of programmability in logic and routing.

Due to this aggressive die reduction and the difference in cell and interconnect architectures between HardCopy II and Stratix II devices, it is important that you fit the design in both HardCopy II and Stratix II revisions in the Quartus II software during the front-end design flow phase. To ensure that the design can map between Stratix II and HardCopy II devices, it is necessary to get a successful compile in both revisions. A successful compile is defined as a compiled design project in the Quartus II software that has been successfully placed and routed, has achieved timing closure, and has passed revision comparison between the two revisions.

In most cases, the HardCopy II ASIC compiles without changes after compiling the Stratix II FPGA. For cases where the logic from the Stratix II FPGA does not fit in the HardCopy II ASIC, this document provides an extensive guide to HardCopy II device fitting techniques.

For more information, refer to the *Area and Timing Optimization* chapter in volume 2 of the *Quartus II Handbook*.

Indications Fitting Might be a Problem

By default, the Quartus II software tries to fit three times. If it does not fit on the first try, it might fit on a subsequent attempt. Requiring more than one attempt to fit is an indication that fitting might be a problem. If the peak interconnect use is over 100%, or if the average interconnect use is over 45%, it might not fit. HardCopy II ASICs are routed with two custom metal layers. The peak interconnect use number indicates what percentage of routing resources are used in the most congested area. The design will not fit if more resources are needed than are available.

Factors Affecting Fit

As a designer, you need to consider factors that affect the fit of your prototype Stratix II FPGA onto a Hardcopy II ASIC. These factors include high HCell use, high DSP use, design architecture, and high RAM use.

High HCell Use (Post Synthesis, Including DSP)

Over 75% HCell use is very problematic and at risk of not fitting in the HardCopy II ASIC. You will find HCell use in the Summary section of the Fitter report. HCell use cannot be accurately estimated based on the FPGA compile because the HCell and adaptive logic module (ALM) ratio is a function of the design and depends on how much of the FPGA ALM hardware is being used. An ALM in an FPGA is either used or not used. For a HardCopy II revision, an ALM that is using most of its logic will translate into more HCells than one that is using little of its logic.

High DSP Use

HardCopy II DSPs are built out of HCell fabric using large contiguous areas. Designs with high DSP use might use a majority of the routing resources in an area and cause significant routing blockage.

Design Architecture

Certain topologies are intrinsically routing intensive; for example, large cross-bar switches. High fan-out nets can cause routing congestion if they are distributed in the HardCopy II custom layers rather than the global routing resources, which are built into the base layers.

High RAM Use

RAM blocks are highly optimized and tightly packed. RAM blocks might restrict the use of some routing channels in the area in which they reside.

Techniques to Reduce Congestion and Improve Fitting

There are a variety of techniques that you can employ in your design to reduce congestion and improve the fit of your prototype device. If you are able to use some or all of these techniques, you can make your end-product more efficient and cost-effective.

Promote High Fan-Out Nets to Global Resources

In Stratix II FPGAs and HardCopy II ASICs, there are global and regional resources typically used for clocks, but also used for high fan-out signals. These global resources are pre-built and do not take up regular programmable routing resources. There are 16 global clocks and 32 regional clocks (8 regional clocks per device quadrant).

While compiling the Stratix II FPGA, the Quartus II software promotes signals to use global resources, as needed, based on the FPGA floorplan. When the HardCopy II companion device is compiled, the Quartus II software uses the same global resources as it did during the FPGA compile because the global resource usage must match (for revision comparison purposes). This does not take into account the HardCopy II floorplan and its routing congestion.

In the Resource section of the Fitter report, you can see if there are high fan-out nets that are not assigned to global resources. If there are global resources available, you can assign the high fan-out nets to the global resources. This needs to be done in both the FPGA and HardCopy II revisions. If the assignment is made to the FPGA revision first, after the FPGA revision is compiled and migrated, the HardCopy II revision assumes these assignments.

An example of a high fan-out net is a reset that goes to 10,000 registers. The reassignments to global or regional resources can be done using the `set_instance_assignment` command with the Tool Control Language (Tcl) console, or added in the Quartus II settings file (.**qsf**).

To assign a signal to a global resource, use the assignment:

```
set_instance_assignment -name GLOBAL_SIGNAL "GLOBAL CLOCK" -to
<signal_name>
```

To assign a signal to a regional clock, use the assignment:

```
set_instance_assignment -name GLOBAL_SIGNAL "FAST_REGIONAL CLOCK"
-to <signal_name>
```

Be sure to check that the regional clock covers the area needed for both the FPGA and HardCopy II designs.

Figure 1 shows a case where the Quartus II software has promoted low fan-out signals to the global clocks. If there are higher fan-out signals, assign them to global clocks or regional clocks.

Figure 1. Example of Low Fan-Out Promotion to the Global Clock

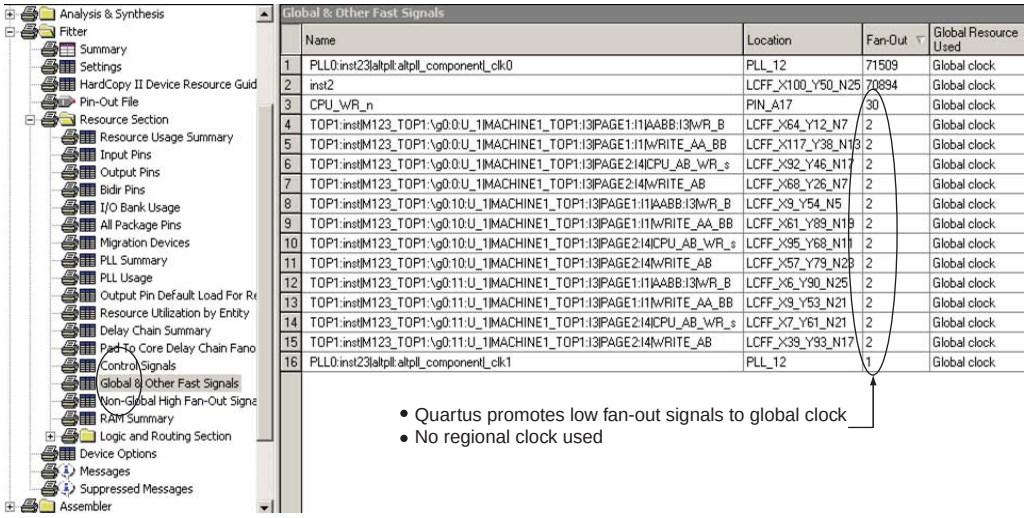


Figure 2 shows high fan-out signals not assigned to global resources. Assigning as many of these signals as possible to global or regional clocks helps with routing congestion.

Figure 2. High Fan-Out Signals Not Using Global/Regional Routing Resources

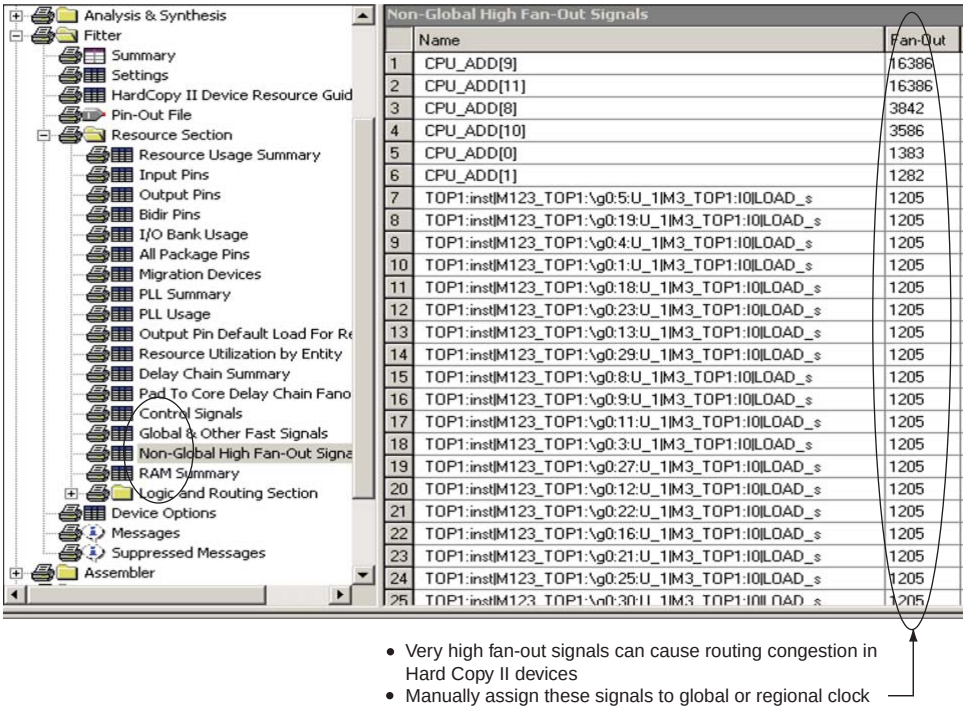
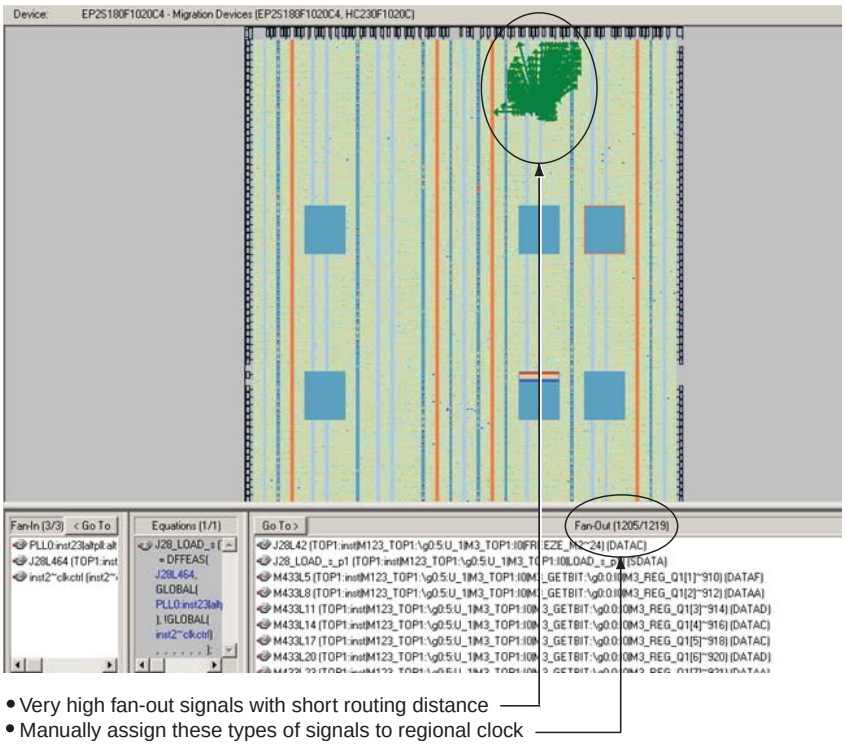


Figure 3 shows high fan-out signals with short routing distances. Reassign these fan-outs to a regional clock resource.

Figure 3. High Fan-Out Signals with Short Routing Distances



Adjust Optimization Settings

Optimization settings control the priorities the Quartus II software uses when compiling a design. You can control optimization at three different levels: the whole design, an entity, or an instance. A design can be optimized for **Area**, for **Speed**, or it can be **Balanced**.

To control optimization for the whole design, the syntax used in the Quartus II settings file is:

```
set_global_assignment –name STRATIXII_OPTIMIZATION_TECHNIQUE <value>
```

where <*value*> is **Area**, **Speed**, or **Balanced**.

To control optimization for an entity, the syntax used in the Quartus II settings file is:

```
set_global_assignment –name STRATIXII_OPTIMIZATION_TECHNIQUE -entity
<entity_name> <value>
```

To control optimization for an instance of a module or entity, the syntax is:

```
set_instance_assignment –name STRATIXII_OPTIMIZATION_TECHNIQUE -to
<instance_name> <value>
```

With a design that is not speed critical, you can elect to optimize for **Area**. This logic option also controls how adder logic is implemented. If you choose **Balanced** or **Speed**, the Quartus II software uses the faster version of the adder, which uses more HCells, thus potentially impacting fitting. If you choose **Area**, the Quartus II software uses the normal version. Use the logic option **Use High Speed Adder** to turn off the use of faster adders. Do this if the adder logic is not on the critical path for performance. For a design with moderate speed requirements, use **Balanced**. Portions of a design that are not speed critical can be optimized for **Area** as well. For example, you can use an assignment like:

```
set_instance_assignment -name STRATIXII_OPTIMIZATION_TECHNIQUE -to
"my.slow.adder:b2v_inst5" AREA
```

These assignments, when used, have to be identically set in both the Stratix II FPGA and HardCopy II revisions. This can reduce HCell use and improve routability. Optimizing for speed tends to make routing more difficult. Over-constraining clock frequencies is not recommended because it uses more resources and increases routing congestion.

Reduce Use of Hard DSP Blocks

It is best to use dedicated DSP blocks to implement DSP functionality. Using hard DSP blocks gives better performance in the Stratix II FPGA. In the HardCopy II ASIC, the DSP HCell library equivalents give the best performance. If enough positive slack exists in the timing, using Logic Elements (LEs) to implement DSP blocks can reduce congestion and HCell use in the HardCopy II revision. This can be done if the FPGA implementation has enough free ALMs to implement the DSP blocks. DSP block implementation using hard DSP blocks or LEs must be the same between the Stratix II FPGA and the HardCopy II revisions.

You can control DSP block use at three levels: the whole design, an entity, or an instance. You can use the following settings:

- **Auto** allows the Quartus II software to choose when to use DSP blocks.
- **DSP Blocks** forces the Quartus II software to use DSP blocks.
- **Logic Elements** forces the Quartus II software to use LEs instead of DSP blocks.

To set an entire design to use automatic selection of DSP block implementation, use this assignment:

```
set_global_assignment -name DSP_BLOCK_BALANCING "Auto"
```

To set an entity to use **Logic Elements**, use this assignment:

```
set_global_assignment -name DSP_BLOCK_BALANCING -entity <entity
name> "Logic Elements"
```

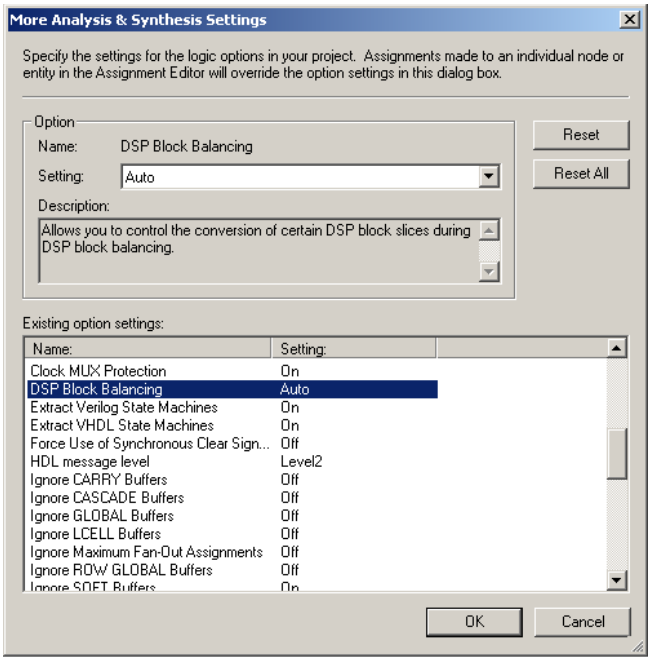
To set an instance to use hard **DSP blocks**, use this assignment:

```
set_instance_assignment -name DSP_BLOCK_BALANCING DSP -to
"mult_mux:u1|Mult0"
```

To control the global use of DSP blocks using the Quartus II software, perform the following steps:

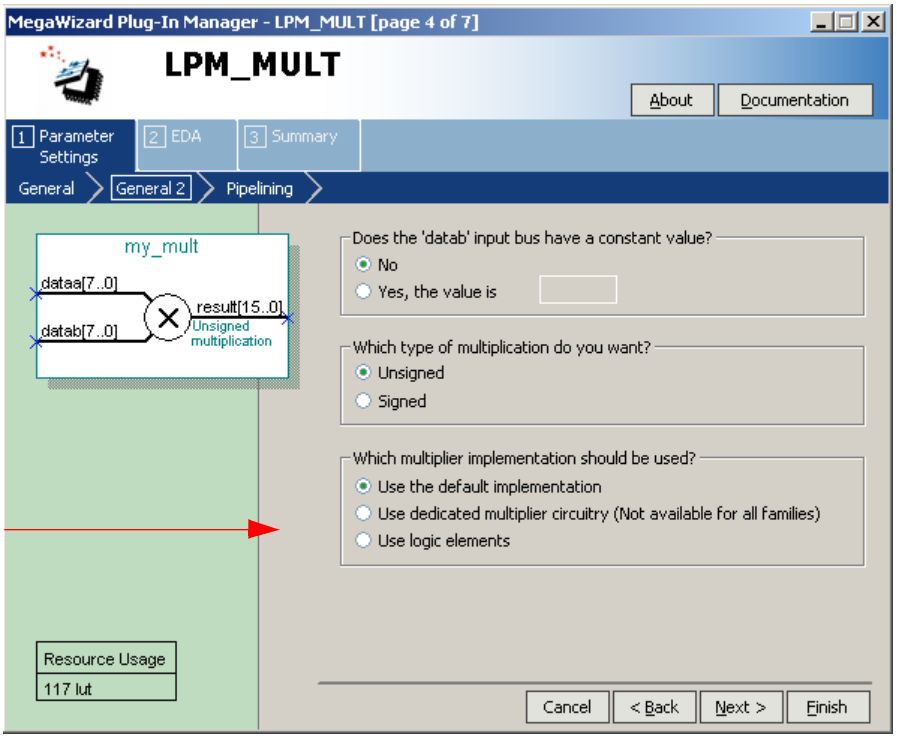
1. On the Assignments menu, click **Settings**.
2. In the **Category** list, select **Analysis & Synthesis Settings**.
3. Click the **More Settings** button.
4. Under **Existing option settings**, scroll down to **DSP Block Balancing** and click it.
5. In the **Setting** box, select how you want the DSP blocks implemented, as shown in (Figure 4).

Figure 4. Implementing DSP Block Balancing in the More Analysis & Synthesis Settings Menu



Certain megafunctions that use DSP blocks also allow you to select Hard DSP block or LE implementation. An example of this is the LPM_MULT megafunction, as highlighted by the red arrow in Figure 5. In this instance, page 4 of the MegaWizard® Plug-In Manager indicates that the Quartus II software will use the default implementation. In conjunction with the global settings described above, you can use these options to tailor a specific entity.

Figure 5. MegaWizard Plug-In Manager - LPM_MULT [page 4 of 7]

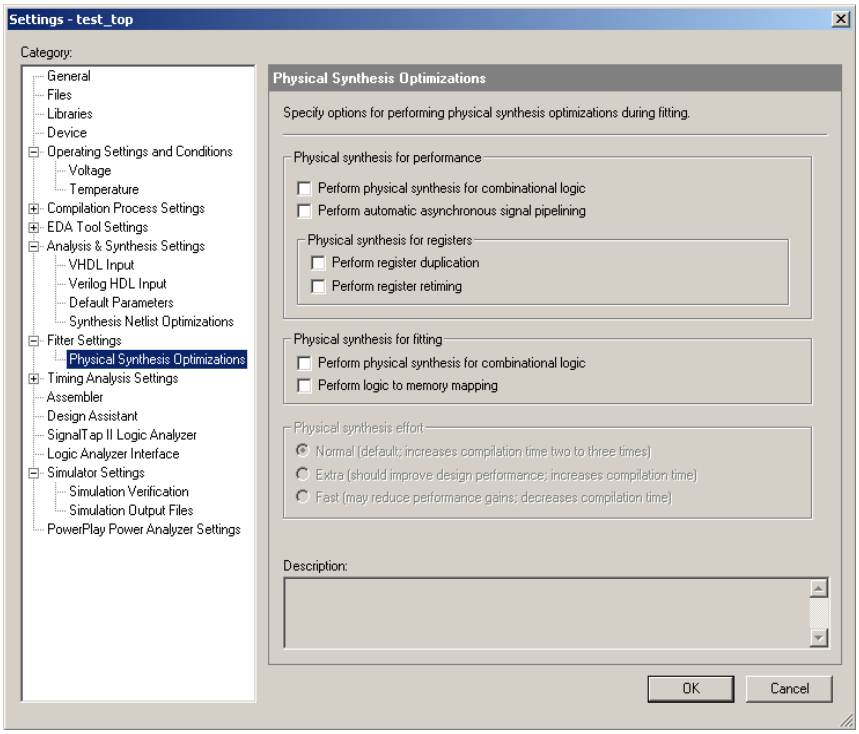


Turn Off Physical Synthesis

Physical synthesis causes duplication of signals and registers in an effort to meet timing requirements. If your design can meet timing without using physical synthesis, it will more easily fit in your HardCopy II revision. Using physical synthesis can increase congestion and HCell use in the HardCopy II implementation. Turning physical synthesis off can improve routability and reduce HCell use. You must configure physical synthesis identically in both the Stratix II FPGA and HardCopy II revisions.

Physical synthesis settings are located under the **Fitter Settings** in the **Category** list, as shown in Figure 6. Leaving the selections blank turns off physical synthesis.

Figure 6. Fitter Settings, Physical Synthesis Optimizations



For more information about physical synthesis, refer to the *Netlist Optimizations and Physical Synthesis* chapter in volume 2 of the *Quartus II Handbook*.

Adjust Fitter Hold Time Optimization Settings

Migrating from a Stratix II FPGA to a HardCopy II ASIC involves mapping ALM logic blocks to HCell macros. In the FPGA, hold time requirements are guaranteed by design. In the HardCopy II revision, due to different clock structures and HCell macros that are faster than the equivalent ALM logic blocks, the Quartus II software or HardCopy II Design Center (HCDC) backend tools might have to insert buffers to guarantee that hold time requirements are met for all operating conditions and processing variations. In general, set the Quartus II software Fitter options to **Optimize fast-corner timing** and **Optimize hold timing** for **All paths**. For some hard-to-fit designs that are close to or just out of routing resources, you can de-select **Optimize fast-corner timing**. These and other typical hold time requirement settings for the Quartus II Fitter are shown in Figure 7. Additionally, you can try the **Optimize hold timing** for the **I/O paths and minimum TPD paths** option, as shown in Figure 8.

Contact the HCDC before using these settings to ensure that the backend tools will still be able to close hold time requirements.

Figure 7 shows the Quartus II software **Fitter Settings** options as typically set to meet hold time requirements.

Figure 7. Typical Hold Time Requirement Settings for the Quartus II Fitter

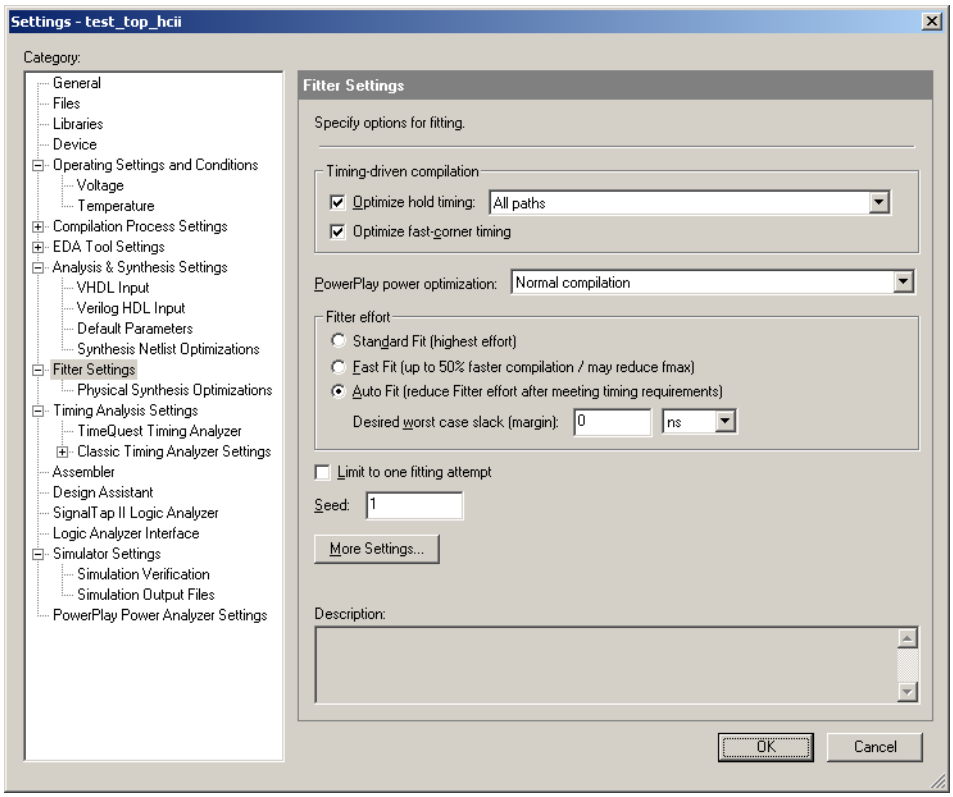
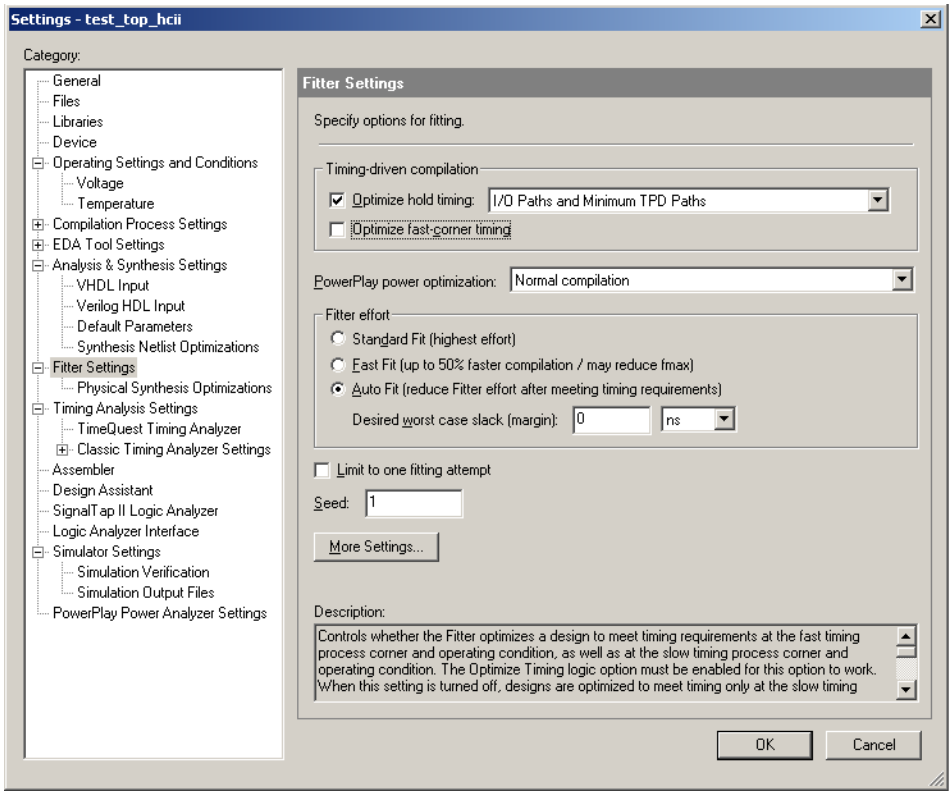


Figure 8 shows the Quartus II Software Fitter options set to let the HCDC backend tools do the optimizations necessary to meet the hold time requirements for hard-to-fit designs.

Figure 8. Quartus II Fitter HCDC Backend Tools Optimization Settings for Hard-to-Fit Designs

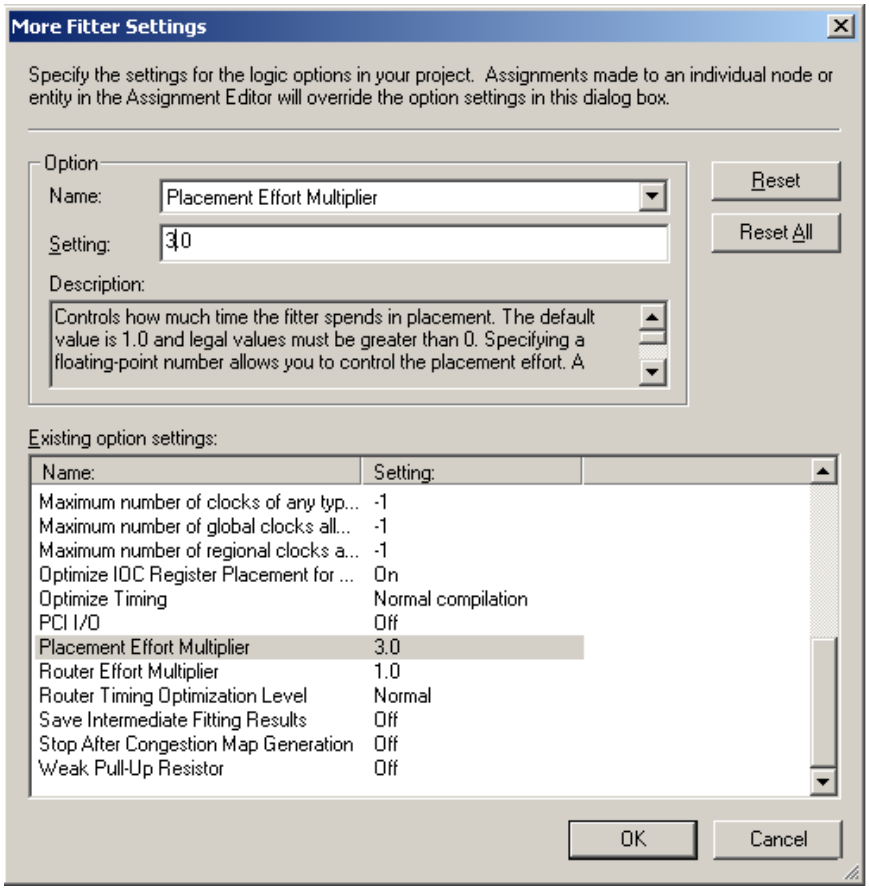


Another setting that is sometimes useful in getting a hard-to-route design to fit is the **Placement Effort Multiplier** setting in the **More Fitter Settings** dialog box, which you can access by selecting the **More Settings** button on the **Fitter Settings** page. This setting causes the Quartus II software to try multiple possibilities in its effort to optimize your design, though the compile time will increase. Increasing the multiplier to as much as 3 can produce a better starting placement and improve routing.

Figure 9 shows the **More Fitter Settings** dialog box and an example of the **Placement Effort Multiplier** set to 3.

The **More Fitter Settings** dialog box also has the **Router Effort Multiplier** setting. Increasing this has not been found to help routing for HardCopy II designs, though it does make compile times much longer. Altera recommends you leave this set to 1.

Figure 9. Example of Placement Effort Multiplier Setting



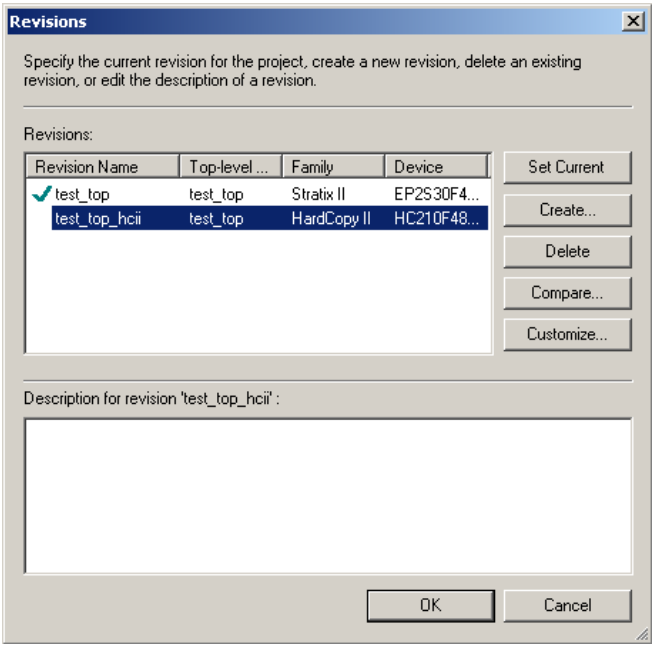
Use a HardCopy II First Flow

Sometimes the optimizations made to create the Stratix II FPGA are not the best design choices for the HardCopy II ASIC. Compiling the HardCopy II revision first, then mapping it to the FPGA, can produce a better fit.

To employ a HardCopy II first flow, if you have already created an FPGA with a HardCopy II companion revision, perform the following steps:

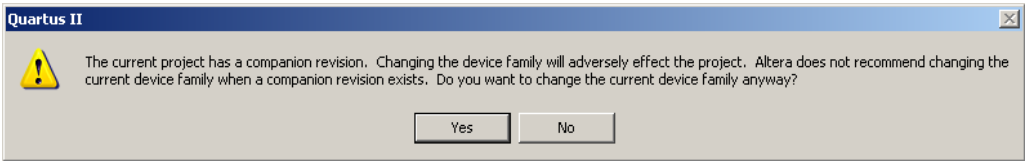
1. On the Project menu, click **Revisions**.
2. Delete the HardCopy II companion revision (**test_top_hcii** in this example), as shown in Figure 10.

Figure 10. Example of Revisions Dialog Box Used to Delete a HardCopy II Companion Revision



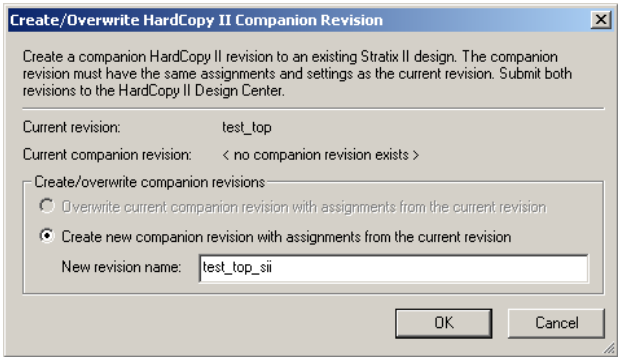
3. Select the main revision (**test_top** in this example) and change to the HardCopy II device that you wish to use. You will see the warning shown in Figure 11.

Figure 11. Warning



4. Click **Yes** because you have already deleted the companion revision.
5. Compile the HardCopy II revision and migrate it to a new FPGA companion device, as shown in Figure 12.

Figure 12. Create New Companion Revision with Assignments from the Current Revision



6. Finally, change to the FPGA revision and compile it.

For more information about HardCopy II First Flow, refer to the *Quartus II Support for HardCopy II Devices* chapter in volume 1 of the *HardCopy II Device Handbook*.

Optimize Floorplan

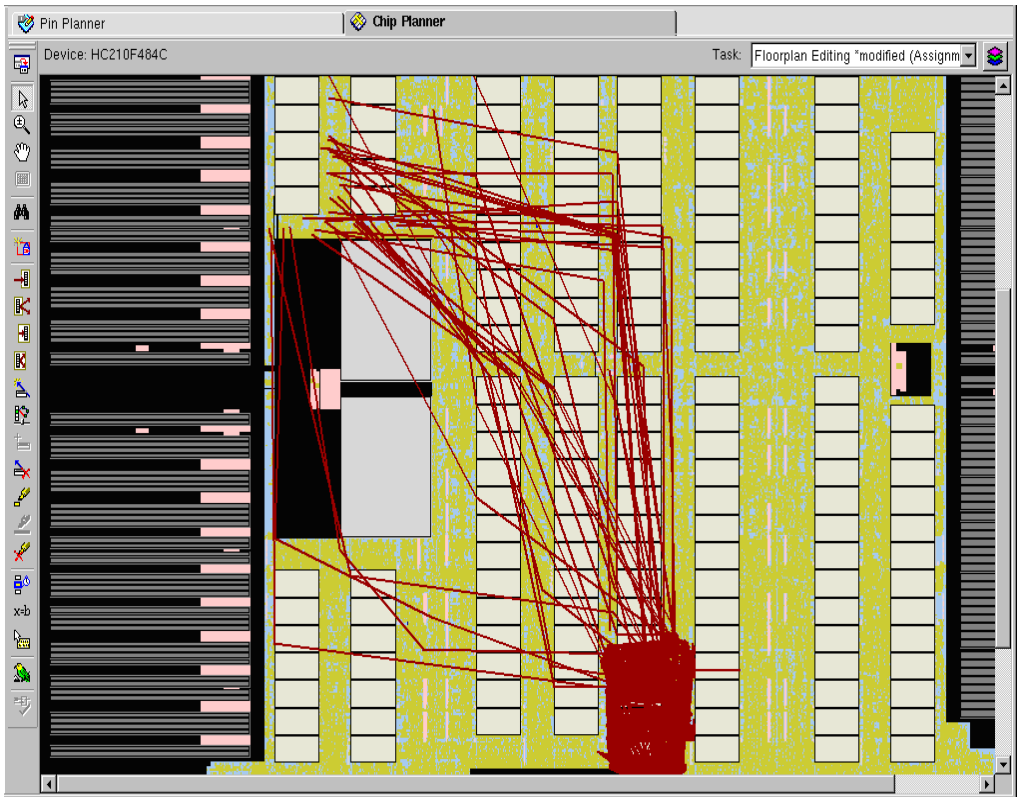
A good floorplan has blocks that need to communicate with each other in close proximity, particularly blocks on critical timing paths. For example, a block that is connected to I/O cells is best placed near those I/O cells. You can determine the quality of the floorplan by reading the congestion map. To view the congestion map, open the **Chip Planner** and click **Show Routing Congestion**.

LogicLock™ regions do not automatically migrate from the Stratix II FPGA to the HardCopy II ASIC. If they are needed, you must recreate them in the HardCopy II revision. Altera suggests that you try to compile the design first, without recreating the LogicLock regions in the HardCopy II revision, to determine if you can obtain a fit.

Use Fly Lines to Understand Connectivity

In **Chip Planner**, you can select a component that has connections that go through a congested area. If you turn on **Show Node Fan-In** and **Show Node Fan-Out**, the fly lines show where the component is connected. By moving the component, you can try to get its connections to go through areas that are not highly congested. If most of the fly lines go in one direction, move the component in that direction. An example of a block that would benefit from being moved is shown in Figure 13.

Figure 13. Example of a Poorly Placed Block, as Indicated by the Show Node Fan-In and Show Node Fan-Out Fly Lines



For more information about moving blocks of logic, refer to the *Analyzing and Optimizing the Design Floorplan* chapter in volume 2 of the *Quartus II Handbook*.

Due to differences in the physical implementation between the Stratix II FPGA and the HardCopy II ASIC, LogicLock regions are created separately for each. A design that is helped by LogicLock regions in the FPGA might not need them in the HardCopy II revision.

RTL Coding Style

Use an RTL coding style that avoids producing structures that are difficult to implement in a HardCopy II ASIC. For example, coding extremely wide muxes and crossbar switches tends to cause routing congestion.

Inefficient use of DSP blocks can waste resources. Multipliers come in multiples of 9×9; therefore, a 12×12 multiplier uses the same resources as an 18×18 multiplier. It is best to let the Quartus II software choose which kinds of memory to use to implement your design. In particular, using M-RAMs tends to increase routing congestion.

For more information, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

Initially, Restrict the Fitter to One Attempt

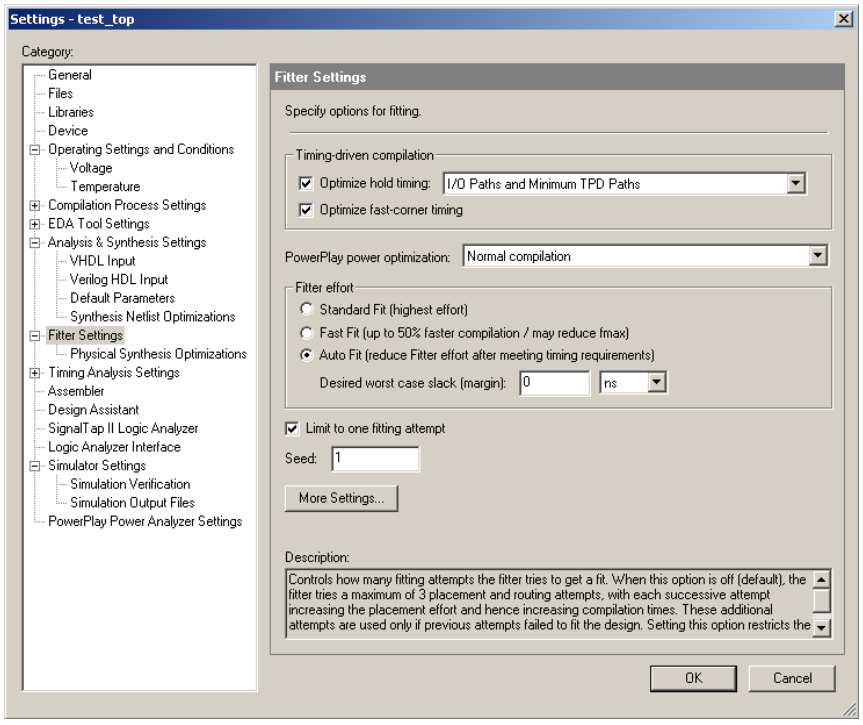
In the early stages of trying to fit a design, restricting the fitter to one attempt can reduce compile times and allow you to try more options. The assignment in the Quartus II settings file is:

```
set_global_assignment -name FIT_ONLY_ONE_ATTEMPT ON
```

To limit the number of fitting attempts using the Quartus II software, turn on the **Limit to one fitting attempt** check box on the **Fitter Settings** page, as shown in Figure 14.

You can eliminate the fitting attempt restriction when the design appears to be close to fitting.

Figure 14. Fitter Settings to Restrict Fitter to One Attempt



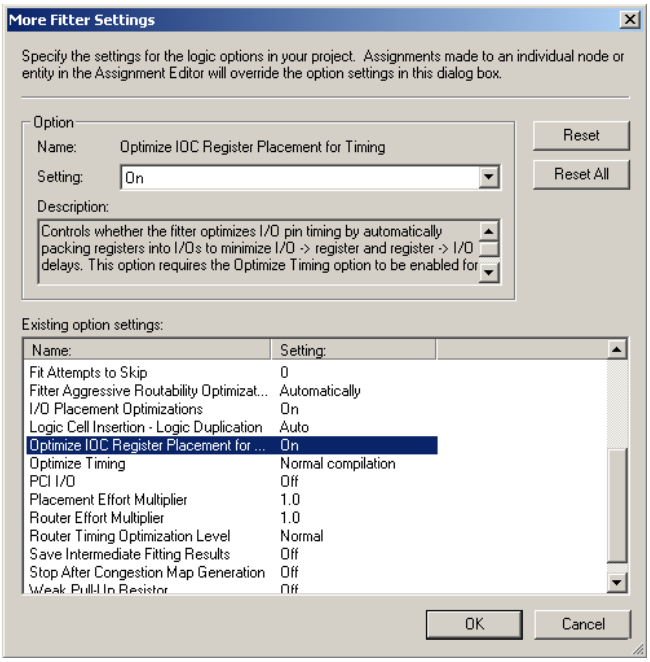
Use Fast Input and Fast Output Registers

Fast input and fast output registers are packed into I/O cells so they take fewer HCell resources.

To control I/O cell register use in the Quartus II software, perform the following steps:

1. On the Assignments menu, click **Settings**.
2. In the **Category** list, select **Fitter Settings**.
3. On the **Fitter Settings** page, click **More Settings**.
4. Under **Existing option** settings, scroll down to **Optimize IOC Register Placement for Timing** and select it.
5. In the **Setting** box, select **On**, as shown in Figure 15.

Figure 15. Optimize IOC Register Placement for Timing Turned On in More Fitter Settings Dialog Box



Use a Larger HardCopy II ASIC

If a successful fit cannot be achieved because of a shortage of HCells, routing resources, memory, or DSP blocks, you might need to use a larger device. The HC230F1020 device can be migrated to the HC240F1020 device and retain the same package. Other upgrade paths require different packages.

For this reason, Altera recommends that you successfully compile both the Stratix II FPGA and the HardCopy II ASIC before doing a PCB layout. Other options are to optimize the RTL by removing redundant, optional, or unused features. Vertical mapping to a larger device comes at a price in package pin count, additional board real estate for the larger package body, additional power consumption, and additional device cost.

Conclusion

Taking a Stratix II FPGA to a HardCopy II ASIC produces a dramatic die size reduction, equal or better performance, equal or lower power consumption, and cost savings. To achieve all of these benefits, some designs require additional effort to fit into a HardCopy II ASIC.

The techniques outlined in this application note are a supplement to *Section III. Area, Timing, and Power Optimization* in volume 2 of the *Quartus II Handbook*.

Document Revision History

Table 2 shows the revision history for this application note.

Table 2. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2008, v2.0	■ Added additional fitter settings. ■ Minor text and format edits.	—
June 2007, v1.0	Initial release.	—

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support