17. Engineering Change Management with the Chip Planner

Programmable logic can accommodate changes to a system specification late in the design cycle. In a typical engineering project development cycle, the specification of the programmable logic portion is likely to change after engineering begins or while integrating all system elements. Last-minute design changes, commonly referred to as engineering change orders (ECOs), are small targeted changes to the functionality of a design after the design has been fully compiled. This chapter discusses the design flow for making ECOs, addresses the impact that ECOs have on the design cycle, and describes how you can use the Chip Planner to make ECOs.

The Chip Planner supports ECOs by allowing quick and efficient changes to your logic late in the design cycle. The Chip Planner provides a visual display of your post-place-and-route design mapped to the device architecture of your chosen FPGA and allows you to create, move, and delete logic cells and I/O atoms.

For a list of supported devices, refer to About the Chip Planner in Quartus® II Help.

In addition to making ECOs, the Chip Planner allows you to perform detailed analysis on routing congestion, relative resource usage, logic placement, LogicLock™ regions, fan-ins and fan-outs, paths between registers, and delay estimates for paths.

For more information about using the Chip Planner for design analysis, refer to the Analyzing and Optimizing the Design Floorplan chapter in volume 2 of the Quartus II Handbook.

ECOs directly apply to atoms in the target device. As such, performing an ECO relies on your understanding of the device architecture of the target device. For more information about the architecture of your device, refer to the appropriate device handbook on the Literature page of the Altera website.

This chapter includes the following topics:

- “Engineering Change Orders” on page 17–2
- “ECO Design Flow” on page 17–4
- “The Chip Planner Overview” on page 17–5
- “Performing ECOs with the Chip Planner (Floorplan View)” on page 17–6
- “Performing ECOs in the Resource Property Editor” on page 17–7
- “Change Manager” on page 17–21
- “Scripting Support” on page 17–22
- “Common ECO Applications” on page 17–22
“Post ECO Steps” on page 17–27

**Engineering Change Orders**

In the context of an FPGA design, you can apply an ECO directly to a physical resource on the device to modify its behavior. ECOs are typically made during the verification stage of a design cycle. When a small change is required on a design (such as modifying a PLL for a different clock frequency or routing a signal out to a pin for analysis) recompilation of the entire design can be time consuming, especially for larger designs. Because several iterations of small design changes can occur during the verification cycle, recompilation times can quickly add up. Furthermore, a full recompilation due to a small design change can result in the loss of previous design optimizations. Making ECOs, instead of performing a full recompilation on your design, limits the change only to the affected portions of logic.

This section discusses the areas in which ECOs have an impact on a system design and how the Quartus II software can help you optimize the design in these areas. The following topics are discussed in this section:

- “Performance Preservation”
- “Compilation Time”
- “Verification”
- “Change Modification Record”

**Performance Preservation**

You can preserve the results of previous design optimizations when you make changes to an existing design with one of the following methods:

- Incremental compilation
- Rapid recompile
- ECOs

Choose the method to modify your design based on the scope of the change. The methods above are arranged from the larger scale change to the smallest targeted change to a compiled design.

The incremental compilation feature allows you to preserve compilation results at an RTL component or module level. After the initial compilation of your design, you can assign modules in your design hierarchy to partitions. Upon subsequent compilations, incremental compilation recompiles changed partitions based on the chosen preservation levels.

The rapid recompilation feature leverages results from the latest post-fit netlist to determine the changes required to honor modifications you have made to the source code. If you turn on the rapid recompilation feature, the Compiler attempts to refit only the portion of the netlist that is related to the code modification.

ECOs provide a finer granularity of control compared to the incremental compilation and the rapid recompilation feature. All modifications are performed directly on the architectural elements of the device. You should use ECOs for targeted changes to the post-fit netlist.
In the Quartus II software versions 10.0 and later, the software does not preserve ECO modifications to the netlist when you recompile a design with the incremental compilation feature turned on. You can reapply ECO changes made during a previous compilation with the Change Manager.

For more information about the incremental compilation feature, refer to the *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*.

**Compilation Time**

In the traditional programmable logic design flow, a small change in the design requires a complete recompilation of the design. A complete recompilation of the design consists of synthesis and place-and-route. Making small changes to the design to reach the final implementation on a board can be a long process. Because the Chip Planner works only on the post-place-and-route database, you can implement your design changes in minutes without performing a full compilation.

**Verification**

After you make a design change, you can verify the impact on your design. To verify that your changes do not violate timing requirements, perform static timing analysis with the Quartus II TimeQuest Timing Analyzer after you check and save your netlist changes in the Chip Planner.

For more information about the TimeQuest analyzer, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Additionally, you can perform a gate-level or timing simulation of the ECO-modified design with the post-place-and-route netlist generated by the Quartus II software.

**Change Modification Record**

All ECOs made with the Chip Planner are logged in the Change Manager to track all changes. With the Change Manager, you can easily revert to the original post-fit netlist or you can pick and choose which ECOs to apply.

Additionally, the Quartus II software provides support for multiple compilation revisions of the same project. You can use ECOs made with the Chip Planner in conjunction with revision support to compare several different ECO changes and revert back to previous project revisions when required.
ECO Design Flow

Figure 17–1 shows the design flow for making ECOs.

For iterative verification cycles, implementing small design changes at the netlist level can be faster than making an RTL code change. As such, making ECO changes are especially helpful when you debug the design on silicon and require a fast turnaround time to generate a programming file for debugging the design.
A typical ECO application occurs when you uncover a problem on the board and isolate the problem to the appropriate nodes or I/O cells on the device. You must be able to correct the functionality quickly and generate a new programming file. By making small changes with the Chip Planner, you can modify the post-place-and-route netlist directly without having to perform synthesis and logic mapping, thus decreasing the turnaround time for programming file generation during the verification cycle. If the change corrects the problem, no modification of the HDL source code is necessary. You can use the Chip Planner to perform the following ECO-related changes to your design:

- Document the changes made with the Change Manager
- Easily recreate the steps taken to produce design changes
- Generate EDA simulation netlists for design verification

For more complex changes that require HDL source code modifications, the incremental compilation feature can help reduce recompilation time.

The Chip Planner Overview

The Chip Planner provides a visual display of device resources. It shows the arrangement and usage of the resource atoms in the device architecture that you are targeting. Resource atoms are the building blocks for your device, such as ALMs, LEs, PLLs, DSP blocks, memory blocks, or I/O elements.

The Chip Planner also provides an integrated platform for design analysis and for making ECOs to your design after place-and-route. The toolset consists of the Chip Planner (providing a device floorplan view of your mapped design) and two integrated subtools—the Resource Property Editor and the Change Manager.

For analysis, the Chip Planner can show logic placement, LogicLock regions, relative resource usage, detailed routing information, routing congestion, fan-ins and fan-outs, paths between registers, and delay estimates for paths. Additionally, the Chip Planner allows you to create location constraints or resource assignment changes, such as moving or deleting logic cells or I/O atoms with the device floorplan. For ECO changes, the Chip Planner enables you to create, move, or delete logic cells in the post-place-and-route netlist for fast programming file generation. Additionally, you can open the Resource Property Editor from the Chip Planner to edit the properties of resource atoms or to edit the connections between resource atoms. All changes to resource atoms and connections are logged automatically with the Change Manager.

Opening the Chip Planner

To open the Chip Planner, on the Tools menu, click Chip Planner. Alternatively, click the Chip Planner icon on the Quartus II software toolbar.

Optionally, you can open the Chip Planner by cross-probing from the shortcut menu in the following tools:

- Design Partition Planner
- Compilation Report
- LogicLock Regions window
The Chip Planner Tasks and Layers

The Chip Planner allows you to set up tasks to quickly implement ECO changes or manipulate assignments for the floorplan of the device. Each task consists of an editing mode and a set of customized layer settings.

For more information about tasks and layers in the Chip Planner, refer to About the Chip Planner in Quartus II Help.

For more information about creating assignments and performing analysis with the Chip Planner, as well as the Chip Planner floorplan views, refer to the Analyzing and Optimizing the Design Floorplan chapter in volume 2 of the Quartus II Handbook.

For more information about making ECOs with the ECO mode, refer to “Performing ECOs with the Chip Planner (Floorplan View)” on page 17–6.

Performing ECOs with the Chip Planner (Floorplan View)

You can manipulate resource atoms in the Chip Planner when you select the ECO editing mode. The following ECO changes can be made with the Chip Planner Floorplan view:

- Create atoms
- Delete atoms
- Move existing atoms

To configure the properties of atoms, such as managing the connections between different LEs/ALMs, use the Resource Property Editor.

For more information about editing atom resource properties, refer to “Performing ECOs in the Resource Property Editor” on page 17–7.

To select the ECO editing mode in the Chip Planner, in the Editing Mode list at the top of the Chip Planner, select the ECO editing mode.
Creating, Deleting, and Moving Atoms

You can use the Chip Planner to create, delete, and move atoms in the post-compilation design.

⚠️ For more information about creating, deleting, and moving atoms, refer to Creating, Deleting, and Moving Atoms in Quartus II Help.

Check and Save Netlist Changes

After making all the ECOs, you can run the Fitter to incorporate the changes by clicking the Check and Save Netlist Changes icon in the Chip Planner toolbar. The Fitter compiles the ECO changes, performs design rule checks on the design, and generates a programming file.

Performing ECOs in the Resource Property Editor

You can view and edit the following resources with the Resource Property Editor:

- “Logic Elements” on page 17–7
- “Adaptive Logic Modules” on page 17–10
- “FPGA I/O Elements” on page 17–12
- “PLL Properties” on page 17–24
- “FPGA RAM Blocks” on page 17–19
- “FPGA DSP Blocks” on page 17–20

Logic Elements

An Altera® LE contains a four-input LUT, which is a function generator that can implement any function of four variables. In addition, each LE contains a register fed by the output of the LUT or by an independent function generated in another LE.

You can use the Resource Property Editor to view and edit any LE in the FPGA. To open the Resource Property Editor for an LE, on the Project menu, point to Locate, and then click Locate in Resource Property Editor in one of the following views:

- RTL Viewer
- Technology Map Viewer
- Node Finder
- Chip Planner

⚠️ For more information about LE architecture for a particular device family, refer to the device family handbook or data sheet.

You can use the Resource Property Editor to change the following LE properties:

- Data input to the LUT
- LUT mask or LUT equation
Logic Element Schematic View

Figure 17–2 shows how the LE appears in the Resource Property Editor. By default, the Quartus II software displays the used resources in blue and the unused resources in gray.

Figure 17–2. Stratix LE Architecture (1)

Notes to Figure 17–2:
(1) For more information about the Stratix device's LE architecture, refer to the Stratix Device Handbook.

Logic Element Properties

Figure 17–3 shows an example of the properties that can be viewed for a selected LE in the Resource Property Editor. To view LE properties, on the View menu, click View Properties.

Figure 17–3. LE Properties
Modes of Operation

LUTs in an LE can operate in either normal or arithmetic mode.

When an LE is configured in normal mode, the LUT in the LE can implement a function of four inputs.

When the LE is configured in arithmetic mode, the LUT in the LE is divided into two 3-input LUTs. The first LUT generates the signal that drives the output of the LUT, while the second LUT generates the carry-out signal. The carry-out signal can drive only a carry-in signal of another LE.

For more information about LE modes of operation, refer to volume 1 of the appropriate device handbook.

Sum and Carry Equations

You can change the logic function implemented by the LUT by changing the sum and carry equations. When the LE is configured in normal mode, you can change only the sum equation. When the LE is configured in arithmetic mode, you can change both the sum and the carry equations.

The LUT mask is the hexadecimal representation of the LUT equation output. When you change the LUT equation, the Quartus II software automatically changes the LUT mask. Conversely, when you change the LUT mask, the Quartus II software automatically computes the LUT equation.

sload and sclr Signals

Each LE register contains a synchronous load (sload) signal and a synchronous clear (sclr) signal. You can invert either the sload or sclr signal feeding into the LE. If the design uses the sload signal in an LE, the signal and its inversion state must be the same for all other LEs in the same LAB. For example, if two LEs in a LAB have the sload signal connected, both LEs must have the sload signal set to the same value. This is also true for the sclr signal.

Register Cascade Mode

When register cascade mode is enabled, the cascade-in port feeds the input to the register. The register cascade mode is used most often when the design implements shift registers. You can change the register cascade mode by connecting (or disconnecting) the cascade in the port. However, if you create this port, you must ensure that the source port LE is directly above the destination LE.

Cell Delay Table

The cell delay table describes the propagation delay from all inputs to all outputs for the selected LE.
Logic Element Connections
To view the connections that feed in and out of an LE, on the View menu, click View Port Connections. Figure 17–4 shows the LE connections in the Connectivity window.

![Figure 17–4. View LE Connections](image)

Delete a Logic Element
To delete an LE, follow these steps:

1. Right-click the desired LE in the Chip Planner, point to Locate, and click Locate in Resource Property Editor.

2. You must remove all fan-out connections from an LE prior to deletion. To delete fan-out connections, right-click each connected output signal, point to Remove, and click Fanouts. Select all of the fan-out signals in the Remove Fan-outs dialog box and click OK.

3. To delete an atom after all fan-out connections are removed, right-click the atom in the Chip Planner and click Delete Atom.

Adaptive Logic Modules
Each ALM contains LUT-based resources that can be divided between two adaptive LUTs (ALUTs). With up to eight inputs to the two ALUTs, each ALM can implement various combinations of two functions. This adaptability allows the ALM to be completely backward-compatible with four-input LUT architectures. One ALM can implement any function with up to six inputs and certain seven-input functions. In addition to the ALUT-based resources, each ALM contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. The ALM can efficiently implement various arithmetic functions and shift registers with these dedicated resources.

You can implement the following types of functions in a single ALM:

- Two independent 4-input functions
- An independent 5-input function and an independent 3-input function
- A 5-input function and a 4-input function, if they share one input
- Two 5-input functions, if they share two inputs
- An independent 6-input function
- Two 6-input functions, if they share four inputs and share the same functions
Certain 7-input functions

You can use the Resource Property Editor to change the following ALM properties:
- Data input to the LUT
- LUT mask or LUT equation

**Adaptive Logic Module Schematic**

You can view and edit any ALM atom with the Resource Property Editor by right-clicking the ALM in the RTL Viewer, the Node Finder, or the Chip Planner, and clicking **Locate in Resource Property Editor** (Figure 17–5).

For a detailed description of the ALM, refer to the device handbooks of devices based on an ALM architecture.

---

**Figure 17–5. ALM Schematic (1)**

---

**Note to Figure 17–5:**

(1) By default, the Quartus II software displays the used resources in blue and the unused in gray. For **Figure 17–5**, the used resources are in blue and the unused resources are in gray.

---

**Adaptive Logic Module Properties**

The properties that you can display for the ALM include an equations table that shows the name and location of each of the two combinational nodes and two register nodes in the ALM, the individual LUT equations for each of the combinational nodes, and the combout, sumout, carryout, and shareout equations for each combinational node.
Adaptive Logic Module Connections

On the View menu, click View Connectivity to view the input and output connections for the ALM.

FPGA I/O Elements

Altera FPGAs that have high-performance I/O elements, including up to six registers, are equipped with support for a number of I/O standards that allow you to run your design at peak speeds. Use the Resource Property Editor to view, change connectivity, and edit the properties of the I/O elements. Use the Chip Planner (Floorplan view) to change placement, delete, and create new I/O elements.

For a detailed description of the device I/O elements, refer to the applicable device handbook.

You can change the following I/O properties:

- Delay chain
- Bus hold
- Weak pull up
- Slow slew rate
- I/O standard
- Current strength
- Extend OE disable
- PCI I/O
- Register reset mode
- Register synchronous reset mode
- Register power up
- Register mode

Stratix V I/O Elements

The I/O elements in Stratix® V devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional single data rate (SDR) or double data rate (DDR) transfer (shown in Figure 17–6).

I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output enable path for handling the output enable signal to the output buffer. These registers allow faster source-synchronous register-to-register transfers and resynchronization. The input path consists of the DDR input registers, alignment and synchronization registers, and half data rate blocks; you can bypass each block in the input path. The input path uses the deskew delay to adjust the input register clock delay across process, voltage, and temperature (PVT) variations.
By default, the Quartus II software displays the used resources in blue and the unused resources in gray.

Figure 17–6. Stratix V Device I/O Element Structure

For more information about I/O elements in Stratix V devices, refer to the *Stratix V Device Handbook*. 
**Arria GX, Stratix, Stratix II, and Stratix GX I/O Elements**

The I/O elements in Arria® GX, Stratix, Stratix II, and Stratix GX devices contain a bidirectional I/O buffer, six registers, and a latch for a complete bidirectional SDR or DDR transfer.

Figure 17–7 shows the Stratix and Stratix GX I/O element structure. The I/O element structure contains two input registers (plus a latch), two output registers, and two output enable registers. By default, the Quartus II software displays the used resources in blue and the unused resources in gray.

Figure 17–7. Stratix and Stratix GX Device I/O Element and Structure

For more information about I/O elements in Stratix and Stratix GX devices, refer to the [Stratix Device Handbook](#) and the [Stratix GX Device Handbook](#).
Arria II GX, Stratix III, and Stratix IV I/O Elements

The I/O elements in Arria II GX, Stratix III, and Stratix IV devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional SDR or DDR transfer (shown in Figure 17–8). The I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output enable path for handling the output enable signal for the output buffer. Each path consists of a set of delay elements that allow you to fine-tune the timing characteristics of each path for skew management. By default, the Quartus II software displays the used resources in blue and the unused resources in gray.

Figure 17–8. Stratix III Device I/O Element and Structure

For more information about I/O elements in Stratix III and Stratix IV devices, refer to the Literature page of the Altera website.

For more information about programmable I/O elements in Stratix III devices, refer to AN 474: Implementing Stratix III Programmable I/O Delay Settings in the Quartus II Software.
**Cyclone and Cyclone II I/O Elements**

The I/O elements in Cyclone® and Cyclone II devices contain a bidirectional I/O buffer and three registers for complete bidirectional single data-rate transfer. Figure 17–9 shows the Cyclone and Cyclone II I/O element structure. The I/O element contains one input register, one output register, and one output enable register. By default, the Quartus II software displays the used resources in blue and the unused resources in gray.

*Figure 17–9. Cyclone and Cyclone II Device I/O Elements and Structure*

For more information about I/O elements in Cyclone II and Cyclone devices, refer to the Cyclone II Device Handbook and Cyclone Device Handbook, respectively.
Cyclone III I/O Elements

The I/O elements in Cyclone III devices contain a bidirectional I/O buffer and five registers for complete embedded bidirectional single data rate transfer. Figure 17–10 shows the Cyclone III I/O element structure. The I/O element contains one input register, two output registers, and two output-enable registers. The two output registers and two output-enable registers are utilized for double-data rate (DDR) applications. You can use the input registers for fast setup times and the output registers for fast clock-to-output times. Additionally, you can use the output-enable (OE) registers for fast clock-to-output enable timing. You can use I/O elements for input, output, or bidirectional data paths. By default, the Quartus II software displays the used resources in blue and the unused resources in gray.

Figure 17–10. Cyclone III Device I/O Elements and Structure

For more information about I/O elements in Cyclone III devices, refer to the Cyclone III Device Handbook.
MAX II I/O Elements

The I/O elements in MAX® II devices contain a bidirectional I/O buffer. Figure 17–11 shows the MAX II I/O element structure. You can drive registers from adjacent LABs to or from the bidirectional I/O buffer of the I/O element. By default, the Quartus II software displays the used resources in blue and the unused resources in gray.

Figure 17–11. MAX II Device I/O Elements and Structure

For more information about I/O elements in MAX II devices, refer to the MAX II Device Handbook.
FPGA RAM Blocks

With the Resource Property Editor, you can view the architecture of different RAM blocks in the device, modify the input and output registers to and from the RAM blocks, and modify the connectivity of the input and output ports. Figure 17–12 shows an M9K RAM view in a Stratix III device.

Figure 17–12. M9K RAM View in a Stratix III Device

Note to Figure 17–12:
(1) By default, the Quartus II software displays the used resources in blue and the unused resources in gray. In Figure 17–12, the used resources are in blue and the unused resources are in gray.
FPGA DSP Blocks

Dedicated hardware DSP circuit blocks in Altera devices provide performance benefits for the critical DSP functions in your design. The Resource Property Editor allows you to view the architecture of DSP blocks in the Resource Property Editor for the Cyclone and Stratix series of devices. The Resource Property Editor also allows you to modify the signal connections to and from the DSP blocks and modify the input and output registers to and from the DSP blocks.

Figure 17–13 shows the DSP architecture in a Stratix III device.

Figure 17–13. DSP Block View in a Stratix III Device

Note to Figure 17–13:
(1) By default, the Quartus II software displays the used resources in blue and the unused resources in gray. In Figure 17–13, the used resources are in blue and the unused resources are in red.
Change Manager

The Change Manager maintains a record of every change you perform with the Chip Planner, the Resource Property Editor, the SignalProbe feature, or a Tcl script. Each row of data in the Change Manager represents one ECO.

The Change Manager allows you to apply changes, roll back changes, delete changes, and export change records to a Text File (.txt), a Comma-Separated Value File (.csv), or a Tcl Script File (.tcl). The Change Manager tracks dependencies between changes, so that when you apply, roll back, or delete a change, any prerequisite or dependent changes are also applied, rolled back, or deleted.

For more information about the Change Manager, refer to About the Change Manager in Quartus II Help.

Complex Changes in the Change Manager

Certain changes (including creating or deleting atoms and changing connectivity) can appear to be self-contained, but are actually composed of multiple actions. The Change Manager marks such complex changes with a plus icon in the Index column.

You can click the plus icon to expand the change record and show all the component actions performed as part of that complex change.

For more information about complex change records and about managing changes with the Change Manager, refer to Examples of Managing Changes With the Change Manager in Quartus II Help.

Managing SignalProbe Signals

The SignalProbe pins that you create from the SignalProbe Pins dialog box are recorded in the Change Manager. After you have made a SignalProbe assignment, you can use the Change Manager to quickly disable SignalProbe assignments by selecting Revert to Last Saved Netlist on the shortcut menu in the Change Manager.

For more information about SignalProbe pins, refer to the Quick Design Debugging Using SignalProbe chapter in volume 3 of the Quartus II Handbook.

Exporting Changes

You can export changes to a .txt, a .csv, or a .tcl. Tcl scripts allow you to reapply changes that were deleted during compilation.

For more information about exporting changes, refer to Managing Changes With the Change Manager in Quartus II Help.

For more information about netlist types and the Quartus II incremental compilation feature, refer to the Quartus II Incremental Compilation for Hierarchical and Team-Based Design chapter in volume 1 of the Quartus II Handbook.
Scripting Support

You can run procedures and make settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. The Tcl commands for controlling the Chip Planner are located in the chip_planner package of the quartus_cdb executable.

A comprehensive list of Tcl commands for the Chip Planner can be found in About Quartus II Scripting in Quartus II Help.

For more information about Tcl scripting, refer to the Tcl Scripting chapter in volume 2 of the Quartus II Handbook. For more information about all settings and constraints in the Quartus II software, refer to the Quartus II Settings File Manual. For more information about command-line scripting, refer to the Command-Line Scripting chapter in volume 2 of the Quartus II Handbook.

Common ECO Applications

This section provides examples of how you might use an ECO to make a post-compilation change to your design. To help build your system quickly, you can use Chip Planner functions to perform the following activities:

- Adjust the drive strength of an I/O with the Chip Planner
- Modify the PLL properties with the Resource Property Editor (see “Modify the PLL Properties With the Chip Planner” on page 17–23)
- Modify the connectivity between new resource atoms with the Chip Planner and Resource Property Editor

Adjust the Drive Strength of an I/O with the Chip Planner

To adjust the drive strength of an I/O, follow the steps in this section to incorporate the ECO changes into the netlist of the design.
1. In the **Editing Mode** list at the top of the Chip Planner, select the ECO editing mode.

2. Locate the I/O in the **Resource Property Editor**, as shown in Figure 17–14.

![Figure 17–14. I/O in the Resource Property Editor](image)

3. In the **Resource Property Editor**, point to the **Current Strength** option in the **Properties** pane and double-click the value to enable the drop-down list.

4. Change the value for the **Current Strength** option.

5. Right-click the ECO change in the Change Manager and click **Check & Save All Netlist Changes** to apply the ECO change.

You can change the pin locations of input or output ports with the ECO flow. You can drag and move the signal from an existing pin location to a new location while in the Post Compilation Editing (ECO) task in the Chip Planner. You can then click **Check & Save All Netlist Changes** to compile the ECO.

**Modify the PLL Properties With the Chip Planner**

You use PLLs to modify and generate clock signals to meet design requirements. Additionally, you can use PLLs to distribute clock signals to different devices in a design, reducing clock skew between devices, improving I/O timing, and generating internal clock signals.
The Resource Property Editor allows you to view and modify PLL properties to meet your design requirements. Using the Stratix PLL as an example, the rest of this section describes the adjustable PLL properties and the equations as a function of the adjustable PLL properties that govern the PLL output parameters.

Figure 17–15 shows a Stratix PLL in the Resource Property Editor.

**Figure 17–15. PLL View in a Stratix Device**

---

**PLL Properties**

The Resource Property Editor allows you to modify PLL options, such as phase shift, output clock frequency, and duty cycle. You can also change the following PLL properties with the Resource Property Editor:

- Input frequency
- $M V_{CO}$ tap
- $M$ initial
- $M$ value
- $N$ value
- $M$ counter delay
- $N$ counter delay
You can also view post-compilation PLL properties in the Compilation Report. To do so, in the Compilation Report, select **Fitter** and then select **Resource Section**.

### Adjusting the Duty Cycle

Use **Equation 17–1** to adjust the duty cycle of individual output clocks.

**Equation 17–1.**

\[
\text{High \%} = \frac{\text{Counter High}}{(\text{Counter High} + \text{Counter Low})}
\]

### Adjusting the Phase Shift

Use **Equation 17–2** to adjust the phase shift of an output clock of a PLL.

**Equation 17–2.**

\[
\text{Phase Shift} = (\text{Period } V_{\text{CO}} \times 0.125 \times \text{Tap } V_{\text{CO}}) + (\text{Initial } V_{\text{CO}} \times \text{Period } V_{\text{CO}})
\]

For normal mode, **Tap** \( V_{\text{CO}} \), **Initial** \( V_{\text{CO}} \), and **Period** \( V_{\text{CO}} \) are governed by the following settings:

- **Tap** \( V_{\text{CO}} \) = Counter Delay – \( M \) Tap \( V_{\text{CO}} \)
- **Initial** \( V_{\text{CO}} \) = Counter Initial – \( M \) Initial
- **Period** \( V_{\text{CO}} \) = In Clock Period \( \times N \times M \)

For external feedback mode, **Tap** \( V_{\text{CO}} \), **Initial** \( V_{\text{CO}} \), and **Period** \( V_{\text{CO}} \) are governed by the following settings:

- **Tap** \( V_{\text{CO}} \) = Counter Delay – \( M \) Tap \( V_{\text{CO}} \)
- **Initial** \( V_{\text{CO}} \) = Counter Initial – \( M \) Initial
- **Period** \( V_{\text{CO}} \) = \( \frac{\text{In Clock Period} \times N}{(M + \text{Counter High} + \text{Counter Low})} \)
For a detailed description of the settings, refer to the Quartus II Help. For more information about Stratix device PLLs, refer to the *Stratix Architecture* chapter in volume 1 of the *Stratix Device Handbook*. For more information about PLLs in Arria GX, Cyclone, Cyclone II, and Stratix II devices, refer to the appropriate device handbook.

### Adjusting the Output Clock Frequency

Use Equation 17–3 to adjust the PLL output clock in normal mode.

**Equation 17–3.**

\[
\text{Output Clock Frequency} = \text{Input Frequency} \times \frac{M \text{ value}}{N \text{ value} + \text{Counter High} + \text{Counter Low}}
\]

Use Equation 17–4 to adjust the PLL output clock in external feedback mode.

**Equation 17–4.**

\[
\text{OUTCLK} = \frac{M \text{ value} + \text{External Feedback Counter High} + \text{External Feedback Counter Low}}{N \text{ value} + \text{Counter High} + \text{Counter Low}}
\]

### Adjusting the Spread Spectrum

Use Equation 17–5 to adjust the spread spectrum for your PLL.

**Equation 17–5.**

\[
\% \text{spread} = \frac{M_2N_1}{M_1N_2}
\]

### Modify the Connectivity between Resource Atoms

The Chip Planner and Resource Property Editor allow you to create new resource atoms and manipulate the existing connection between resource atoms in the post-fit netlist. These features are useful for small changes when you are debugging a design, such as manually inserting pipeline registers into a combinational path that fails timing, or routing a signal to a spare I/O pin for analysis. Use the following procedure to create a new register in a Cyclone III device and route register output to a spare I/O pin. This example illustrates how to create a new resource atom and modify the connections between resource atoms.

To create new resource atoms and manipulate the existing connection between resource atoms in the post-fit netlist, follow these steps:

1. Create a new register in the Chip Planner.
2. Locate the atom in the Resource Property Editor.
3. To assign a clock signal to the register, right-click the clock input port for the register, point to Edit connection, and click Other. Use the Node Finder to assign a clock signal from your design.
4. To tie the SLOAD input port to VCC, right-click the clock input port for the register, point to Edit connection, and click VCC.
5. Assign a data signal from your design to the SDATA port.
6. In the Connectivity window, under the output port names, copy the port name of the register.
7. In the Chip Planner, locate a free I/O resource and create an output buffer.
8. Locate the new I/O atom in the Resource Property Editor.
9. On the input port to the output buffer, right-click, point to Edit connection, and click Other.
10. In the Edit Connection dialog box, type the output port name of the register you have created.
11. Run the ECO Fitter to apply the changes by clicking Check and Save Netlist Changes.

A successful ECO connection is subject to the available routing resources. You can view the relative routing utilization by selecting Routing Utilization as the Background Color Map in the Layers Settings dialog box of the Chip Planner. Also, you can view individual routing channel utilization from local, row, and column interconnects with the tooltips created when you position your mouse pointer over the appropriate resource. Refer to the device data sheet for more information about the architecture of the routing interconnects of your device.

Post ECO Steps

After you make an ECO change with the Chip Planner, you must perform static timing analysis of your design with the TimeQuest analyzer to ensure that your changes did not adversely affect the timing performance of your design.

For example, when you turn on one of the delay chain settings for a specific pin, you change the I/O timing. Therefore, to ensure that the design still meets all timing requirements, you should perform static timing analysis.

For more information about performing a static timing analysis of your design, refer to The Quartus II TimeQuest Timing Analyzer chapter in volume 3 of the Quartus II Handbook.

Conclusion

The Chip Planner allows you to analyze and modify your design floorplan. Also, ECO changes made with the Chip Planner do not require a full recompilation, eliminating the lengthy process of RTL modification, resynthesis, and another place-and-route cycle. In summary, the Chip Planner speeds design verification and timing closure.
Table 17–1. Document Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 2012</td>
<td>12.0.0</td>
<td>Removed survey link.</td>
</tr>
<tr>
<td>November 2011</td>
<td>10.1.1</td>
<td>Template update.</td>
</tr>
<tr>
<td>December 2010</td>
<td>10.1.0</td>
<td>Updated chapter to new template</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed “The Chip Planner FloorPlan Views” section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added Stratix V I/O elements in “FPGA I/O Elements” on page 17–12.</td>
</tr>
<tr>
<td>July 2010</td>
<td>10.0.0</td>
<td>Added information to page 17–1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added information to “Engineering Change Orders” on page 17–2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed heading from “Performance” to “Performance Preservation” on page 17–2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated information in “Performance Preservation” on page 17–2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed heading from “Documentation” to “Change Modification Record” on page 17–3.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed heading from “Resource Property Editor” to “Performing ECOs in the Resource Property Editor” on page 17–15.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed “Using Incremental Compilation in the ECO Flow” section. Preservation support for ECOs with the incremental compilation flow has been removed in the Quartus II software version 10.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed “Referenced Documents” section.</td>
</tr>
<tr>
<td>November 2009</td>
<td>9.1.0</td>
<td>Updated device support list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Made minor editorial updates</td>
</tr>
<tr>
<td>March 2009</td>
<td>9.0.0</td>
<td>Updated Figure 17–17.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Made minor editorial updates.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chapter 15 was previously Chapter 13 in the 8.1.0 release.</td>
</tr>
<tr>
<td>November 2008</td>
<td>8.1.0</td>
<td>Corrected preservation attributes for ECOs in the section “Using Incremental Compilation in the ECO Flow” on page 15–32.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minor editorial updates.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed to 8½” x 11” page size.</td>
</tr>
<tr>
<td>May 2008</td>
<td>8.0.0</td>
<td>Updated device support list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modified description for ECO support for block RAMs and DSP blocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corrected Stratix PLL ECO example</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added an application example to show modifying the connectivity between resource atoms</td>
</tr>
</tbody>
</table>

For previous versions of the *Quartus II Handbook*, refer to the *Quartus II Handbook Archive.*