# *Design Constraints User's Guide*

*for Software v9.1*

Actel®
POWER MATTERS

**Actel Corporation, Mountain View, CA 94043**



Printed in the United States of America

Part Number: 5-02-00046-15

Release: November 2010



## Trademarks

# Table of Contents

# Design Constraints

Design constraints are usually either requirements or properties in your design. You use constraints to ensure that your design meets its performance goals and pin assignment requirements.

The Designer software supports both timing and physical constraints. In addition, it supports netlist optimization constraints. You can set constraints by either using Actel's interactive tools or by importing constraint files directly into your design session.

## Timing Constraints

Timing constraints represent the performance goals for your designs. Designer software uses timing constraints to guide the timing-driven optimization tools in order to meet these goals.

You can set timing constraints either globally or to a specific set of paths in your design.

You can apply timing constraints to:

- Specify the required minimum speed of a clock domain
- Set the input and output port timing information
- Define the maximum delay for a specific path
- Identify paths that are considered false and excluded from the analysis
- Identify paths that require more than one clock cycle to propagate the data
- Provide the external load at a specific port

To get the most effective results from the Designer software, you need to set the timing constraints close to your design goals. Sometimes slightly tightening the timing constraint helps the optimization process to meet the original specifications.

## Physical Constraints

Designer software enables you to specify the physical constraints to define the size, shape, utilization, and pin/pad placement of a design. You can specify these constraints based on the utilization, aspect ratio, and dimensions of the die. The pin/pad placement depends on the external physical environment of the design, such as the placement of the device on the board.

There are three types of physical constraints:

- I/O assignments

- Set location, attributes, and technologies for I/O ports
- Specify special assignments, such as VREF pins and I/O banks

- Location and region assignments

- Set the location of Core, RAM, and FIFO macros
- Create Regions for I/O and Core macros as well as modify those regions
    - Clock assignments
- Assign nets to clocks
- Assign global clock constraints to global, quadrant, and local clock resources

## Netlist Optimization Constraints

The Designer software enables you to set some advanced design-specific netlist optimizing constraints.

You can apply netlist optimization constraints to:

- Delete or restore a buffer tree
- Manage the fan-outs of the nets
- Manage macro combinations (for example, IO-REG combining)
- Optimize a netlist by removing buffers and/or inverters, propagating constants, and so on

### See Also

Constraint Support by Family

Constraint Entry Table

Constraint File Format by Family

Designer Naming Conventions

# Families Supported

# Constraint Support by Family

Use the Constraint Family Support table to see which constraints you can use for your device family. Click the name of a constraint in the table for more information about it.

Table 1 · Constraint Support by Family

| | IGLOO | SmartFusion and Fusion | ProASIC3 | ProASIC Plus | ProASIC | Axcelerator | eX | SX-A | SX | MX | DX | ACT1 | ACt2/1200XL | ACT3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | | | | | |
| Create a clock | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Create a generated clock | X | X | X | X | X | X | X | X | | | | | | |
| Remove clock uncertainty | X | X | X | X | X | X | X | X | | | | | | |
| Set clock latency | X | X | X | X | X | X | X | X | | | | | | |
| Set clock uncertainty | X | X | X | X | X | X | X | X | | | | | | |
| Set disable timing | X | X | X | | | X (including RTAX-S) | | | | | | | X | X |
| Set false path | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Set input delay | X | X | X | X | | X | | | | | | | | |
| Set load on output port | X | X | X | X | X | X | X | X | | | | | | |

| | IGLOO | SmartFusion and Fusion | ProASIC3 | ProASIC Plus | ProASIC | Axcelerator | eX | SX-A | SX | MX | DX | ACT1 | ACt2/1200XL | ACT3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | | | | | |
| Set maximum delay | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Set minimum delay | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Set multicycle path | X | X | X | X | | X | | | | | | | | |
| Set output delay | X | X | X | X | | X | | | | | | | | |
| **Physical Placement** | | | | | | | | | | | | | | |
| **-Clocks** | | | | | | | | | | | | | | |
| Assign Net to Global Clock | X | X | X | X | X | | | | | | | | | |
| Assign Net to Local Clock | X | X | X | X | X | X | | | | | | | | |
| Assign Net to Quadrant Clock | X | X | X | | | | | | | | | | | |
| **-Regions** | | | | | | | | | | | | | | |

| | IGLOO | SmartFusion and Fusion | ProASIC3 | ProASIC Plus | ProASIC | Axcelerator | eX | SX-A | SX | MX | DX | ACT1 | ACt2/1200XL | ACT3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | | | | | |
| Assign Macro to Region | X | X | X | X | X | X | | | | | | | | |
| Assign Net to Region | X | X | X | X | X | X | | | | | | | | |
| Create Region | X | X | X | X | X | X | | | | | | | | |
| Delete Regions | X | X | X | X | X | X | | | | | | | | |
| Move Region | X | X | X | X | X | X | | | | | | | | |
| Unassign macro(s) driven by net | X | X | X | X | X | X | | | | | | | | |
| Unassign Macro from Region | X | X | X | X | X | X | | | | | | | | |
| **-I/Os** | | | | | | | | | | | | | | |
| Assign I/O to pin | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Assign I/O Macro to Location | X | X | X | X | X | X | | | | | | | | |

| | IGLOO | SmartFusion and Fusion | ProASIC3 | ProASIC Plus | ProASIC | Axcelerator | eX | SX-A | SX | MX | DX | ACT1 | ACt2/1200XL | ACT3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | | | | | |
| Configure I/O Bank | X | X | X | | | X | | | | | | | | |
| Reset attributes on I/O to default settings | X | X | X | X | X | X | | | | | | | | |
| Reset I/O bank to default settings | X | X | X | | | X | | | | | | | | |
| Reserve pins | X | X | X | X | X | X | X | | | | | | | |
| Unreserve pins | X | X | X | X | X | X | X | | | | | | | |
| Unassign I/O macro from location | X | X | X | X | X | X | | | | | | | | |
| **-Block** | | | | | | | | | | | | | | |
| Move Block | X | X | X | | | X | | | | | | | | |
| Set port block | X | X | X | | | X | | | | | | | | |
| Set Block Options | X | X | X | | | X | | | | | | | | |

| -Nets | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Assign Net to Global Clock | X | X | X | X | X | | | | | | | | | |
| Assign Net to Local Clock | X | X | X | X | X | X | | | | | | | | |
| Assign Net to Quadrant Clock | X | X | X | | | | | | | | | | | |
| Assign Net to Region | X | X | X | X | X | X | | | | | | | | |
| Reset net's criticality to default level | | | | | | X | | | | | | | | |
| Set Net's Criticality | | | | | | X | | | | | | | | |
| Unassign macro(s) driven by net | X | X | X | X | | X | | | | | | | | |
| **Netlist Optimization** | | | | | | | | | | | | | | |
| Delete buffer tree | X | X | X | | | | | | | | | | | |
| Demote | X | X | X | X | X | | | | | | | | | |

| | IGLOO | SmartFusion and Fusion | ProASIC3 | ProASIC Plus | ProASIC | Axcelerator | eX | SX-A | SX | MX | DX | ACT1 | ACt2/1200XL | ACT3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | | | | | |
| Global Net to Regular Net | | | | | | | | | | | | | | |
| Promote regular net to global net | X | X | X | X | X | | | | | | | | | |
| Restore buffer tree | X | X | X | X | X | | | | | | | | | |
| Set preserve | X | X | X | | | X | | | | | | | | |

## See Also

Constraint Entry Table

Constraint File Format by Family

# Constraint Entry

Use the Constraint Entry table to see which tools and file formats you can use to enter constraints for your device family.

Click the name of a constraint, a constraint entry tool, file format type, editor, or checkmark in the table for more information about that item.

Table 2 · Constraint Entry by Tool and File Format

| Constraint | SDC | GDC | PDC | PIN | DCF | ChipPlanner | I/O Attribute Editor | PinEditor | SmartTime, Timer | Compile Options |
|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | |
| Create a clock | X | | | | X | | | | X | |
| Create a generated clock | X | | | | | | | | X | |
| Remove clock uncertainty | X | | | | | | | | X | |
| Set clock latency | X | | | | | | | | X | |
| Set clock uncertainty | X | | | | | | | | X | |
| Set disable timing | X | | | | | | | | X | |
| Set false path | X | | | | X | | | | X | |
| Set input delay | X | | | | | | | | X | |
| Set load on output port | X | | | | X | | X | X | X | |

| Constraint | SDC | GDC | PDC | PIN | DCF | ChipPlanner | I/O Attribute Editor | PinEditor | SmartTime, Timer | Compile Options |
|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | |
| Set maximum delay | X | | | | X | | | | X | |
| Set minimum delay | X | | | | | | | | X | |
| Set multicycle path | X | | | | | | | | X | |
| Set output delay | X | | | | | | | | X | |
| **Physical Placement** | | | | | | | | | | |
| **-Clocks** | | | | | | | | | | |
| Assign Net to Global Clock | | X | X | | | | | | | |
| Assign Net to Local Clock | | X | X | | | X | | | | |
| Assign Net to Quadrant Clock | | | X | | | X | | | | |
| **-Regions** | | | | | | | | | | |
| Assign Macro to Region | | | X | | | X | | | | |

| Constraint | SDC | GDC | PDC | PIN | DCF | ChipPlanner | I/O Attribute Editor | PinEditor | SmartTime, Timer | Compile Options |
|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | |
| Assign Net to Region | | X | X | | | X | | | | |
| Create Region | | X | X | | | X | | | | |
| Delete Regions | | | X | | | X | | | | |
| Move region | | | X | | | X | | | | |
| Unassign macro(s) driven by net | | | X | | | X | | | | |
| Unassign macro from region | | | X | | | X | | | | |
| **-I/Os** | | | | | | | | | | |
| Assign I/O to pin | | X | X | X | | X | X | X | | |
| Assign I/O Macro to Location | | X | X | | | X | | | | |
| Configure I/O Bank | | | X | | | X | | X | | |
| Reset attributes | | | X | | | X | X | | | |

| Constraint | SDC | GDC | PDC | PIN | DCF | ChipPlanner | I/O Attribute Editor | PinEditor | SmartTime, Timer | Compile Options |
|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | |
| on I/O to default settings | | | | | | | | | | |
| Reset I/O bank to default settings | | | X | | | X | X | | | |
| Reserve pins | | | X | | | | X | X | | |
| Unreserve pins | | | X | | | | X | X | | |
| Unassign I/O macro from location | | | X | | | X | | | | |
| **-Blocks** | | | | | | | | | | |
| Move Block | | | X | | | | | | | |
| Set port block | | | X | | | X | | | | |
| Set Block Options | | | X | | | | | | | X |
| **-Nets** | | | | | | | | | | |
| Assign Net to Global Clock | | X | X | | | | | | | |

| Constraint | SDC | GDC | PDC | PIN | DCF | ChipPlanner | I/O Attribute Editor | PinEditor | SmartTime, Timer | Compile Options |
|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | |
| Assign Net to Local Clock | | X | X | | | X | | | | |
| Assign Net to Quadrant Clock | | | X | | | X | | | | |
| Assign Net to Region | | X | X | | | X | | | | |
| Reset net's criticality to default level | | | X | | | | | | | |
| Set Net's Criticality | | X | X | | | | | | | |
| Unassign macro(s) driven by net | | | X | | | X | | | | |
| **Netlist Optimization** | | | | | | | | | | |
| Delete buffer tree | | X | X | | | | | | | X |
| Demote Global Net to Regular Net | | X | X | | | | | | | X |
| Promote | | X | X | | | | | | | X |

| Constraint | SDC | GDC | PDC | PIN | DCF | ChipPlanner | I/O Attribute Editor | PinEditor | SmartTime, Timer | Compile Options |
|---|---|---|---|---|---|---|---|---|---|---|
| **Timing** | | | | | | | | | | |
| regular net to global net | | | | | | | | | | |
| Restore buffer tree | | X | X | | | | | | | |
| Set preserve | | | X | | | | | | | |

## See Also

Constraint Support by Family

Constraint File Format by Family

# Constraint File Format by Family

Use the File Format by Family table to see which file formats apply to each type of constraint and each device family.

Table 3 · Constraint File Format by Family

| Family | Timing | | Physical Placement | | | Netlist Optimiization | |
|---|---|---|---|---|---|---|---|
| | SDC | DCF | PDC | PIN | GCF | PDC | GCF |
| IGLOO | X | | X | | | | |
| SmartFusion / Fusion | X | | X | | | X | |
| ProASIC3 | X | | X | | | | |
| ProASIC PLUS | X | | | | X | | X |
| ProASIC | X | | | | X | | X |
| Axcelerator | X | | X | | | X | |
| eX | X | X | | X | | | |
| SX-A | X | X | | X | | | |
| SX | | X | | X | | | |
| MX | | X | | X | | | |
| DX | | X | | X | | | |
| ACT3 | | X | | X | | | |
| ACT2/1200XL | | X | | X | | | |
| ACT1 | | X | | X | | | |

SDC – Synopsys Design Constraints

PDC – Physical Design Constraints for IGLOO, ProASIC3, SmartFusion, Fusion, and Axcelerator

GCF – Design Constraints Format for ProASIC PLUS and ProASIC

DCF – Actel Design Constraints Format

PIN – Pin location constraints

### See Also

Constraint Support by Family

Constraint Entry Table

# Basic Concepts

## Designer Naming Conventions

The names of ports, instances, and nets in an imported netlist are sometimes referred to as their original names. Port names appear exactly as they are defined in a netlist. For example, a port named A/B appears as A/B in ChipPlanner, PinEditor, and I/O Attribute Editor in MultiView Navigator. Instances and nets display the original names plus an escape character (\) before each backslash (/) and each slash (\) that is not a hierarchy separator. For example, the instance named A/\B is displayed as A\/\\B.

Following are Designer's naming conventions by device.

### SmartFusion, Fusion, ProASIC3, and Axcelerator

The following components use the Tcl-compliant original names:

- PDC reader/writer
- SDC reader/writer
- Compile report
- SDF/Netlist writer for back annotation
- MultiView Navigator tools: NetlistViewer, PinEditor, ChipPlanner, and I/O Attribute Editor
- SmartTime
- SmartPower

### ProASIC, ProASIC$^{PLUS}$, SX-A, and eX

The following components use the Tcl-compliant original names:

- SDC reader/writer
- MultiView Navigator tools: NetlistViewer, PinEditor, ChipPlanner, and I/O Attribute Editor
- SmartTime

GCF follows the netlist original names; therefore, use the original names when referring to ports, instances, and nets in GCF files.

#### See Also

PDC Naming Conventions

# Clock

Specifying clock constraints is the most effective way of constraining and verifying the timing behavior of a sequential design. You must use clock constraints to meet your performance goals and to quickly reach timing closure.

Actel recommends that you specify and constrain all clocks used in the design.

To create a clock constraint, you must provide the following clock information:

**Clock source**Specifies the pin or port where the clock signal is defined.

**Clock period or frequency**: Defines the smallest amount of time after which the signal repeats itself.

**Duty cycle**: Defines the percentage of time during which the clock period is high.

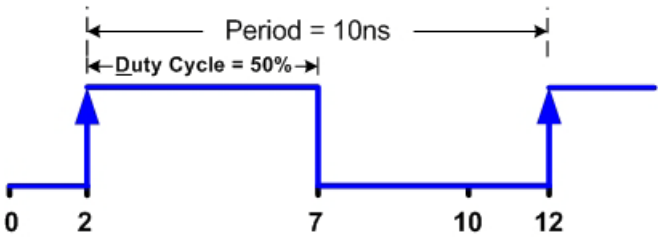
**First edge**: Indicates whether the first edge of the clock is rising or falling.

**Offset**: Indicates the shift of the first edge with respect to instant zero common to all clocks in the design.

## Example 1:

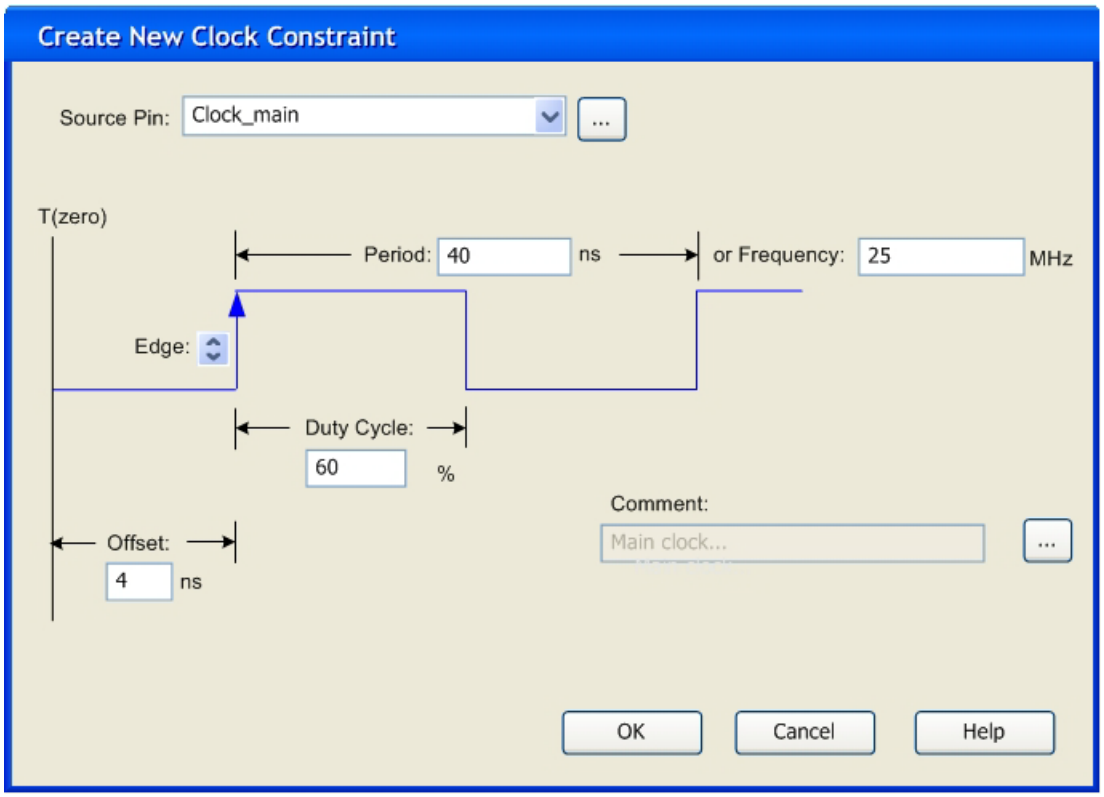
create_clock -period 10 -waveform {2 7}

This example creates a clock with 10ns period, 2ns offset, and 50% duty cycle using the SDC command.




## Example 2:

This example shows how to create a clock with 25MHz frequency, 4ns offset for its first rising edge, and 60% duty cycle using the SmartTime Constraints Editor. Using the Create New Clock Constraint dialog box is equivalent to using the SDC command: create_clock -period 40 -waveform {4, 28}.




## See Also

Constraint support by family

Constraint entry table

create_clock (SDC)

global_clocks (DCF)

Specifying a clock constraint

# Region

A region is a user-defined area on a chip into which you can constrain the physical placement of one or more macros. You can also constrain macros containing multiple tiles for cores, RAMs, and I/Os. The floorplanning process usually requires you to create several regions and assign logic to them. Logic can include core logic, memory, and I/O modules. When you run the place-and-route tool, it places the logic into their assigned regions.

Some regions are user-defined and others are automatically created by the tools to meet routing requirements (for example, Local clock regions).

You can use region constraints to:

- Create user-defined regions such as Inclusive, Exclusive, Empty, LocalClock, and QuadrantClock
- Assign and unassign macros to user-defined regions
- Constrain all the macros connected to a net by assigning them to a specific net region
- Move regions from one set of co-ordinates to another

## See Also

Assign Macro to Region

Create Region

Delete Region

Move Region

Unassign macro from region

*MultiView Navigator User's Guide*: About Floorplanning, Creating Regions, Editing Regions

# Location

Each core, RAM, and I/O macro in the design is associated with a location on the device. When you run the place-and-route tool, it places all of your logic into their assigned locations.

You can use location constraints to:

- Overwrite the existing placements of macros
- Tell the place-and-route tool where to initially place the macros
- Assign I/O macros to specific pins to meet your board's requirements

## See Also

Assign I/O to pin

Assign macro to location

Unassign macro from location

*MultiView Navigator User's Guide*: Assigning Logic to Locations, Moving Logic to Other Locations, Assigning Pins, Unassigning Pins

# I/O Attributes

I/O attributes are the characteristics of logic macros or nets in your design. They indicate placement, implementation, naming, directionality, and other characteristics. This information is used by the design implementation software during the place-and-route of a design.

Input and output attributes are described in the documentation for the I/O Attribute Editor. Attributes applicable to a specific tool are described in the documentation for that tool.

See the topics in I/O Attributes Reference for more detailed information about each attribute. See also , for a table of attributes for each device family, and Welcome to I/O Attribute Editor.

### See Also

I/O Attributes by Family

I/O Standards and I/O Attributes Applicability

I/O Standards Compatibility Matrix

# About the I/O Attribute Editor

The I/O Attribute editor is available from the Libero IDE Project Manager, in the SmartDesign Microcontroller Subsystem configurator, and from MultiView Navigator.

## I/O Attribute Editor Features

The I/O Attribute Editor is a graphical editor that enables you to:

- Create a new physical design I/O constraint
- Modify existing physical I/O constraints
- Import PDC I/O constraint files
- Automatically extract the ports at the top level of an HDL file
- Add, modify, or delete physical I/O constraints from SmartDesign

## I/O Attribute Editor Advantages

- You can create and edit I/O constraints before compiling your design.
- It's efficient. You can re-use the same PDC file for two different modules.
- One module can use several different PDC files.
- You can add, modify, or delete a port from within the SmartDesign Canvas or Grid, and it is automatically updated in the I/O Attribute Editor. The PDC file is automatically passed from SmartDesign to Designer.

In Project Manager, you can edit constraints even before you have written any HDL code. You can edit I/O constraints using any text editor, as you did in previous versions, or you can use the graphical I/O Attribute Editor. You can create a new constraint file from the I/O Attribute Editor in Project Manager if you have a project open.

In Multiview Navigator, you can edit constraints only from a compiled netlist.

You cannot use all design constraints with all families; they are family and die specific.

### Supported families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, and RTAX-S

## I/O Attribute Editor in Libero IDE Project Manager

The I/O Attribute Editor from MultiView Navigator is now integrated in the Libero IDE Project Manager and in SmartDesign. SmartDesign automatically passes I/O constraints to Designer.

You can define physical I/O constraints using I/O Attribute Editor from within the Project Manager.

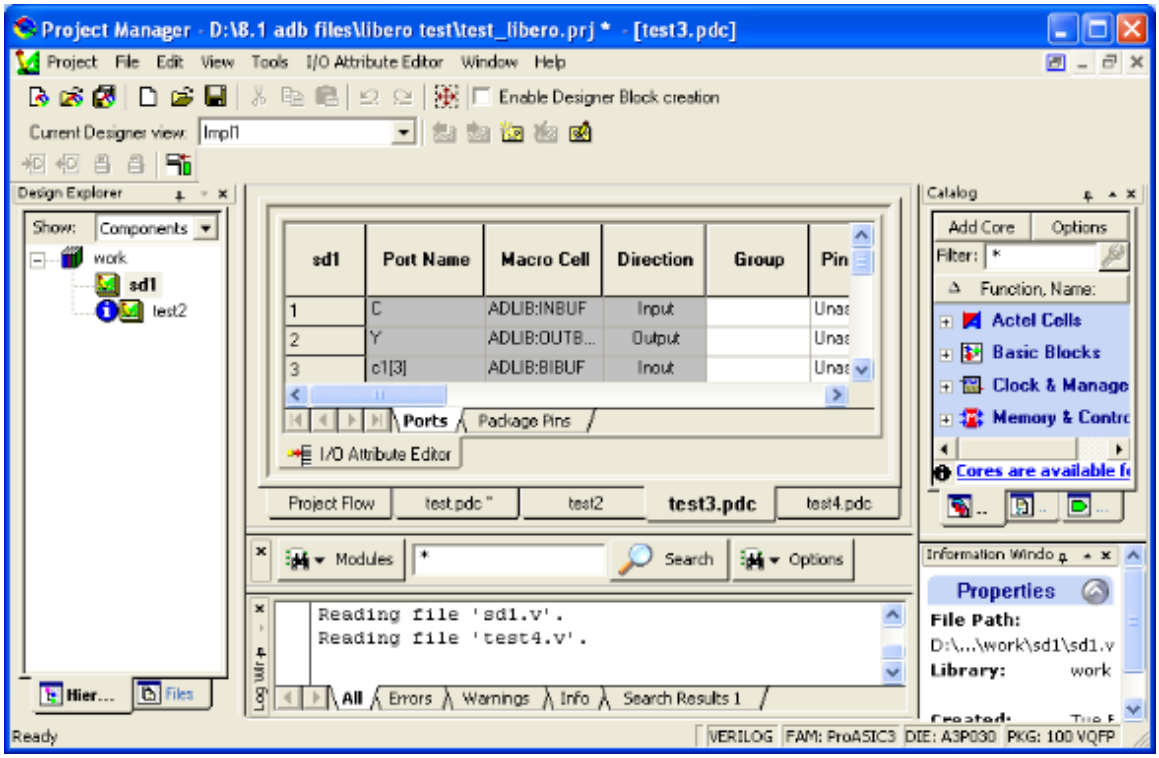
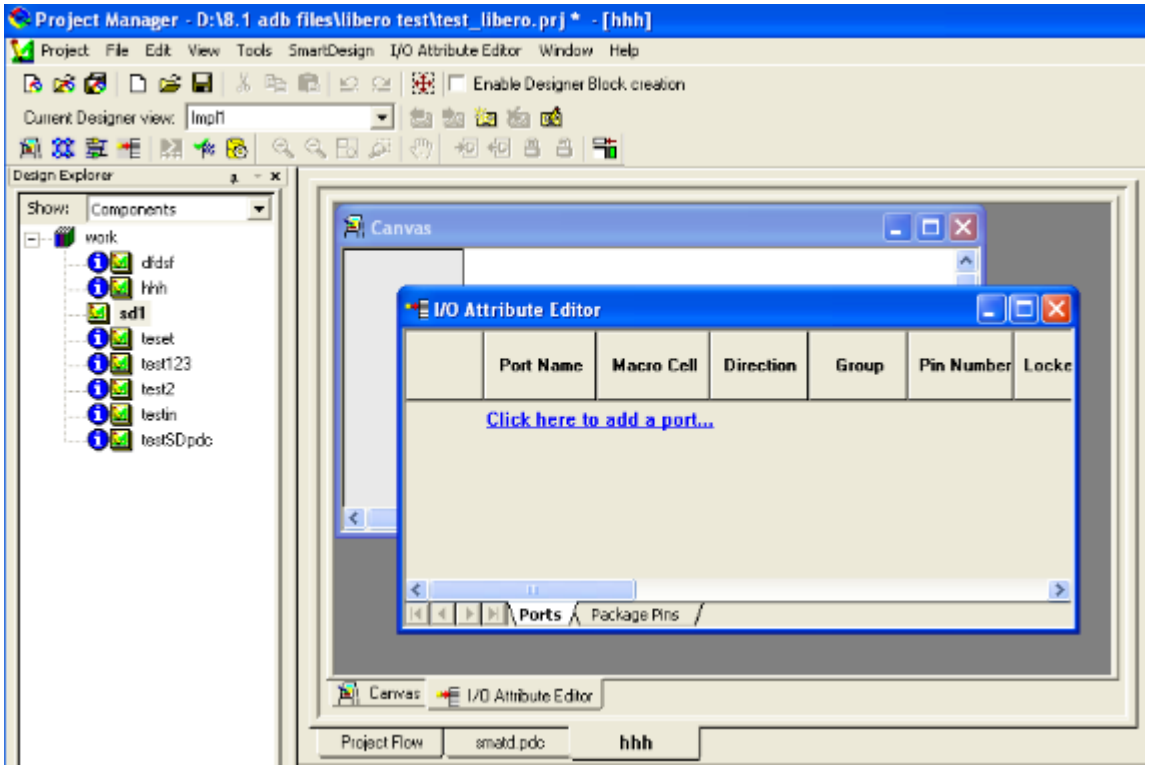



Figure 1 · I/O Attribute Editor in Project Manager

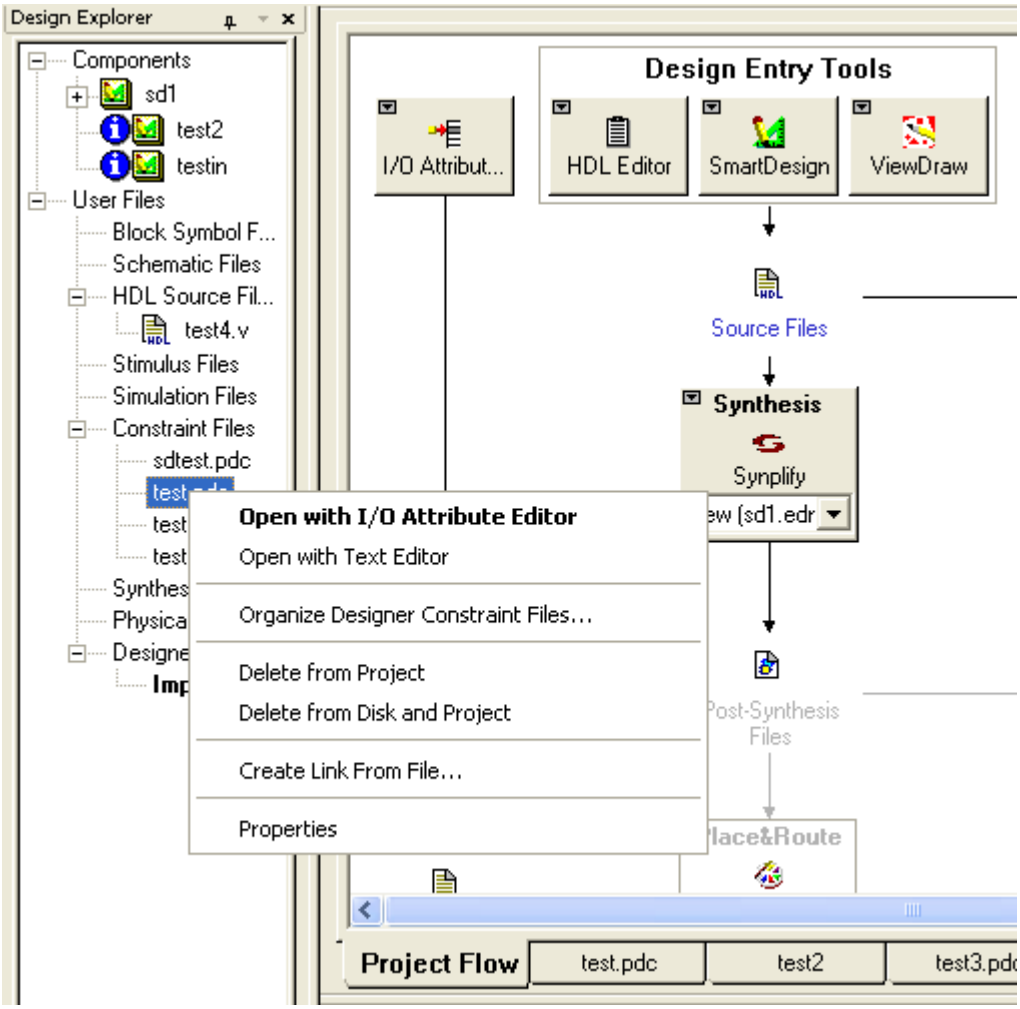
You can start I/O Attribute Editor either from SmartDesign or by opening a PDC file from the Design Explorer.

To start the I/O Attribute Editor from SmartDesign, from the **SmartDesign** menu, choose **Show I/O Attribute Editor View**. The I/O Attribute Editor opens in front of the SmartDesign Canvas as shown below.




I/O Attribute Editor in SmartDesign

To open a PDC file from Design Explorer, right-click the file you want to open in the **Files** tab, and choose **Open with I/O Attribute Editor** as shown below.




Edit a PDC file in the I/O Attribute Editor

You can either load an existing PDC into the I/O Attribute Editor, or you can create a new PDC file using the I/O Attribute Editor. When you load an existing PDC file, only the I/O-related constraints are shown in the I/O Attribute Editor. The other constraints are preserved and inserted at the end of the new, saved file.

# I/O Attribute Editor Project Flow

I/O Attribute Editor enables you to view, create, and modify your physical design constraint (PDC) files. The **Files** tab lists your constraint files in alphabetical order.

You can create I/O physical design constraints using I/O Attribute Editor with the following steps:

**Step One** - Create a new PDC file.

**Step Two** - Add ports to your design.

**Step Three** - Modify the I/O attributes.

**Step Four** - Save the PDC file.

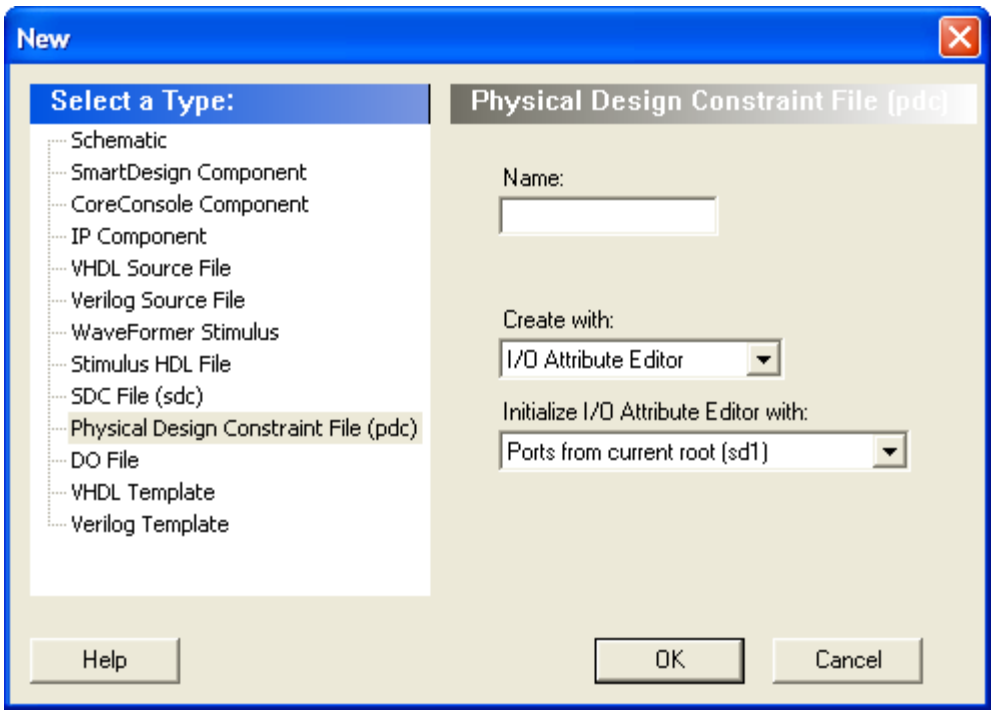
You can save your constraint file at any time.

# Creating a New PDC File

You can create a new constraint file from the I/O Attribute Editor in Project Manager if you have a project open and a module set as root.

### *To create a new PDC file with the I/O Attribute Editor:*

From the **File** menu, choose **New**, or click the I/O Attribute Editor icon in the Project Flow window. The New dialog box appears.



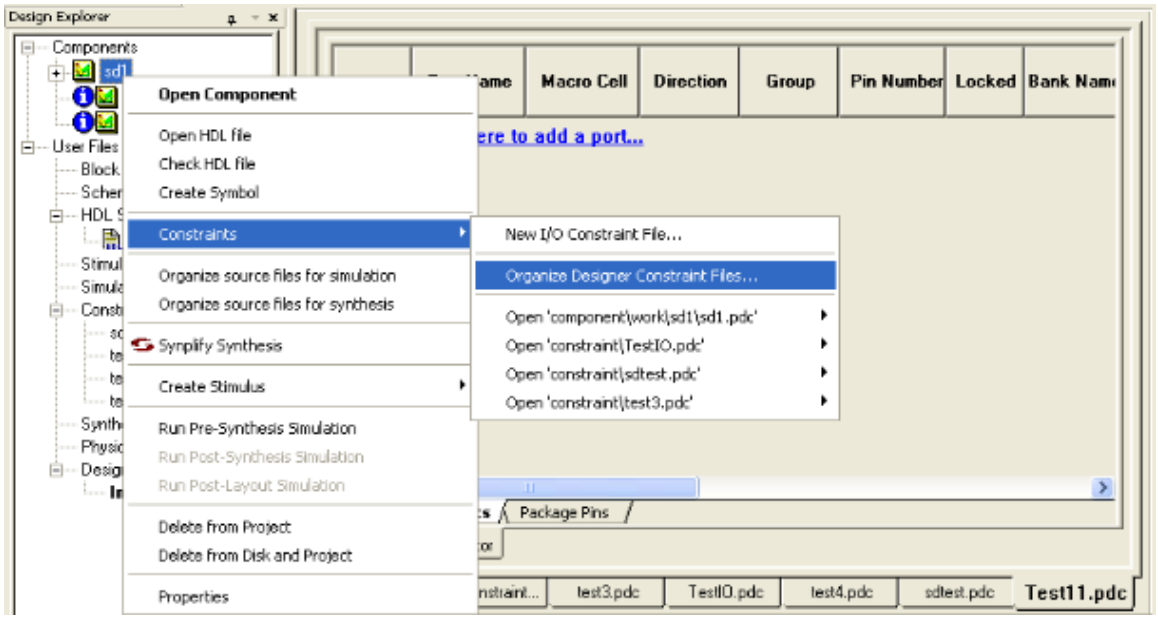

1. Select **Physical Design Constraint File (pdc)** for the file type
2. Type a name for the PDC file.
3. In the **Create with** drop-down list, choose **I/O Attribute Editor**.
4. In the **Initialize I/O Attribute Editor** with drop-down list, choose **Ports from current root (sd1)**, or choose **No Ports** if you do not want to load the ports from the current root.
5. Click **OK**. The I/O Attribute Editor opens, displaying the attributes of the PDC file you just created.
6. From the **File** menu, choose **Save <filename>.pdc**. The saved file is added to your Libero IDE project.

Note: Note: You can also use the right-click menu to create a new PDC file. Select and right-click the root module. Then choose Constraints > New I/O Constraint File from the right-click menu.

Suppose you just want to enter a constraint for a clock and reset the ports. In this case, you do not have to load all the ports before creating the new PDC file.

### *To create a new PDC file with the I/O Attribute Editor without loading all ports:*

1. From the **File** menu, choose **New**, or click the I/O Attribute Editor icon in the Project Flow window. The New dialog box appears.
2. Select **Physical Design Constraint File (pdc)** for the file type.
3. Type a name for the PDC file.
4. In the **Create with** drop-down list, choose **I/O Attribute Editor**.
5. In the **Initialize I/O Attribute Editor with** drop-down list, choose **No Ports** if you do not want to load the ports.
6. Click **OK**. The I/O Attribute Editor opens, displaying the attributes of the PDC file you just created.
7. Save the PDC file.
8. Open the new PDC file in a text editor to confirm that only two constraints were exported for clock and reset.
9. Right-click the root module, and choose **Constraints>Organize Designer Constraint Files** to add the PDC file to Designer as shown below.




10. In the **Organize Constraints for Designer** dialog box, select the files to pass to Designer, and click **Add**.
11. Reopen the PDC file with the I/O Attribute Editor.

When you load ports from a module, the PDC file is automatically associated with that module. However, if no ports are loaded, you need to associate the PDC file using the **Organize Designer Constraint Files** command.

# Opening an Existing PDC File

You can open an existing PDC file after you have opened a project and selected a module that contains the PDC file.

### *To open an existing PDC file do one of the following:*

- In the Design Explorer, from the **Files** tab, expand the Constraint Files list, and double-click the PDC file you want to open. The main window displays the I/O Attribute Editor with the contents of the PDC fle.
- In the Design Explorer, click the **Hierarchy** tab, right-click the PDC file you want to open, and choose **Constraints>Open <pdc filename>**.
- Right-click the I/O Attribute Editor icon in the main window, choose the PDC file to open, and then choose whether to open it with the I/O Attribute Editor or with a text editor, as shown in the figure below.

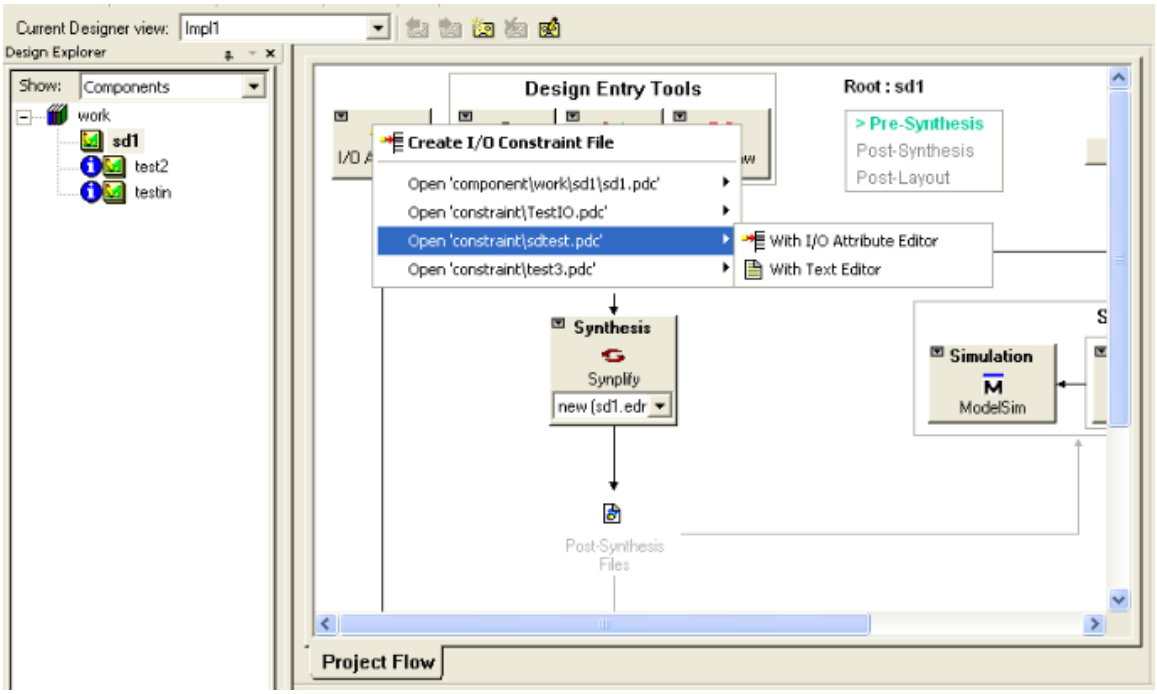



Figure 2 · Right-click the I/O Attribute Editor Icon to Display Its Right-click Menu

The I/O Attribute Editor opens with the I/O attributes of the selected file. When you open an existing PDC file, only the I/O-related constraints are displayed in the I/O Attribute Editor.

# Importing I/O Assignments

You can import I/O attributes from an existing PDC file into the I/O Attribute Editor. Constraints in the PDC file that are not I/O constraints are not read nor are they saved when you save the PDC file.

### *To import a PDC file into the I/O Attribute Editor of your design:*

1. From the **I/O Attribute Editor** menu, choose **Import I/O Assignments**. The Open dialog box appears.
2. In the **Open** dialog box, find and select the PDC file to import, and click **Open**.

# Use Designer PDC Constraints

You can enter pin constraints from Libero Project Manager by either using the text editor to add them to a PDC file or by using the I/O Attribute Editor.

Once you have imported I/O constraint files into Designer, you can modify the constraints with the MultiView Navigator. After modifying the constraints, you can import them back into Project Manager to use the most updated PDC file when performing an iterative design process.

When you use the Use Designer PDC feature, Libero Project Manager will do the following:

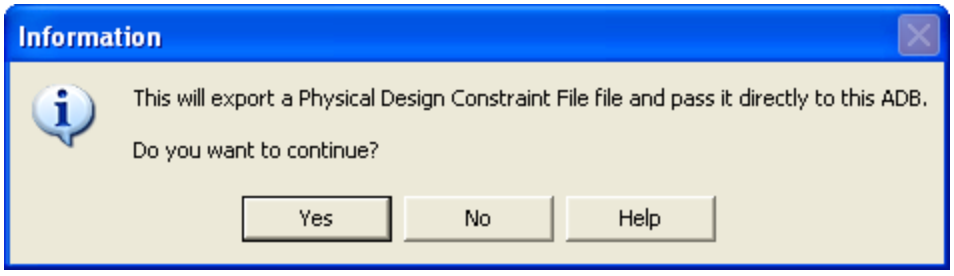
1. Open the ADB file.
2. Export all physical constraints.
3. In Libero IDE, from the **File** menu, choose **Import Files**, and select the PDC file to import into the Libero project.
4. Update the list of constraint files to be imported to Designer. The next time you start Designer, it will include this new PDC file instead of the other PDC files in the Libero project.

Note: Note: After selecting the Use Designer PDC feature, you can make more modifications to the newly added PDC file in Libero Project Manager.

### *To use the Use Designer PDC feature:*

- Right-click an ADB file, and choose **Use Designer PDC** from the right-click menu.
- Right-click a Designer tool, and choose **Use Designer PDC** from the right-click menu.
- From the **Organize Constraint Files for Designer** dialog box, click **Use Designer PDC**. (The Use Designer PDC button does not appear until Layout is complete in Designer.)

An Information box appears asking you to confirm that you want to export the Designer PDC file.




After you click **Yes**, a dialog box appears in which you enter a file name for this Designer PDC file. The default file name appears in the dialog. Click **Save** to replace the old PDC file with the newly modified one, or enter a new, unique name for the PDC which will be used for further design iterations. While converting the PDC file, a message appears, as shown below.


Figure 3 · Information Message Appears During PDC Conversion

After selecting the Use Designer PDC feature, the **Organize Constraints for Designer** dialog box will organize the PDC file as the constraint file to be used by Designer the next time you start it up.

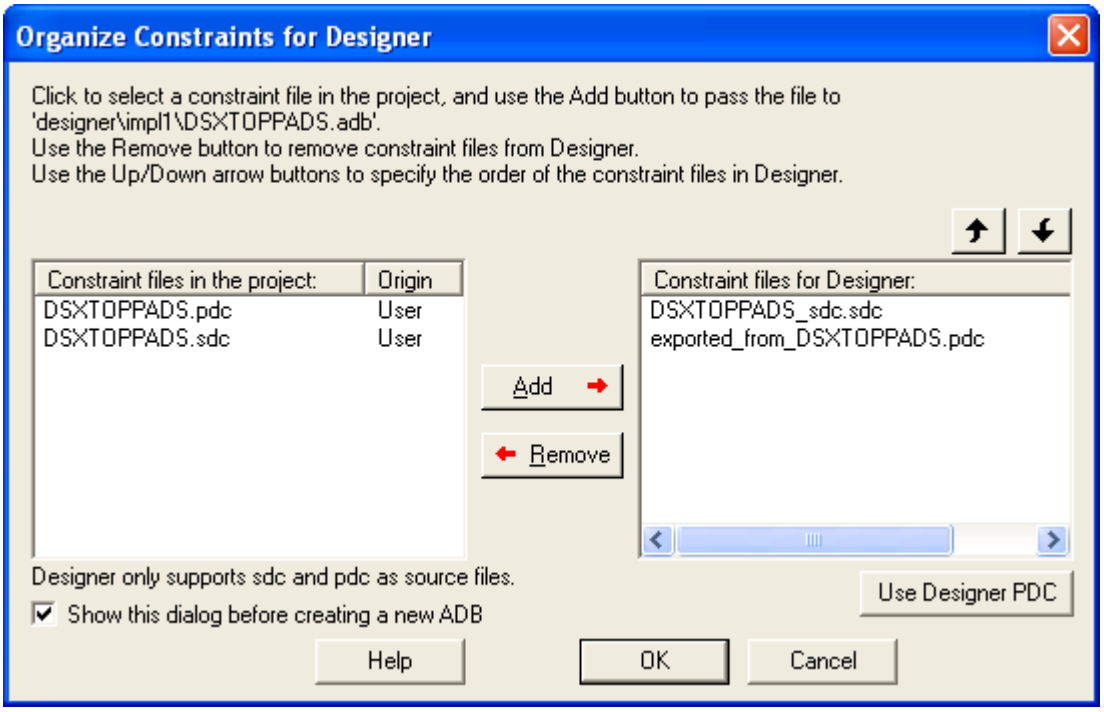



Figure 4 · Organize Constraints for Designer with Use Designer PDC Button

# Adding, Modifying, and Deleting Ports

You can add ports to, delete ports from, and modify ports in the I/O Attribute Editor.

### *To add ports:*

1. From the **I/O Attribute Editor** menu, choose **Add port**. The **Add New Port** dialog box appears (as shown below).

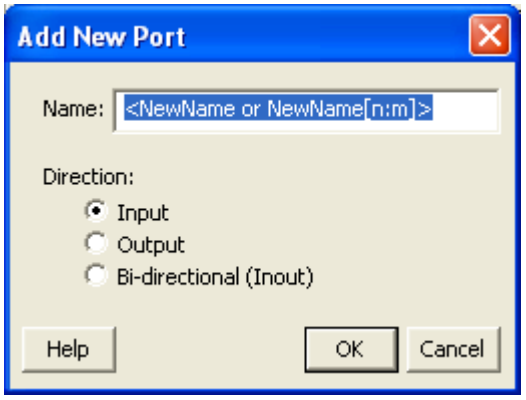



Figure 5 · Add New Port Dialog Box

2. Specify the name of the port you wish to add. You can specify a bus port by including the bus width as part of the name using brackets [ ], such as mybus[3:0].

3. Select the direction of the port.

You can also add a port using the right-click menu.

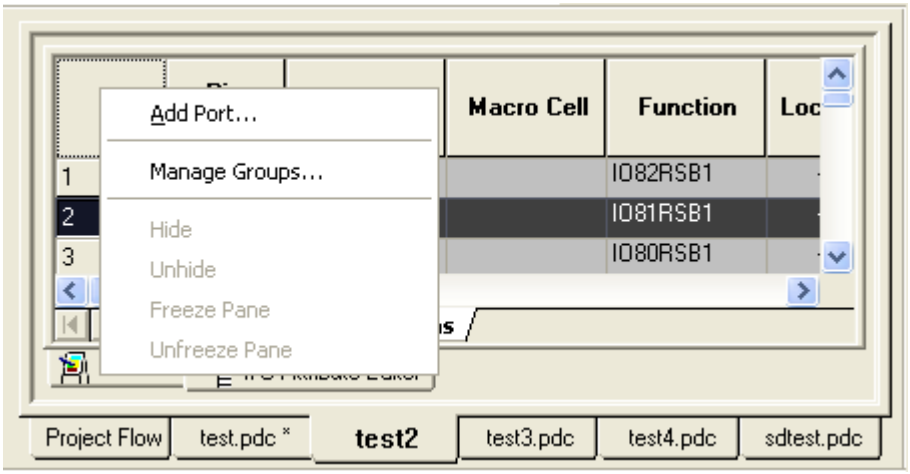
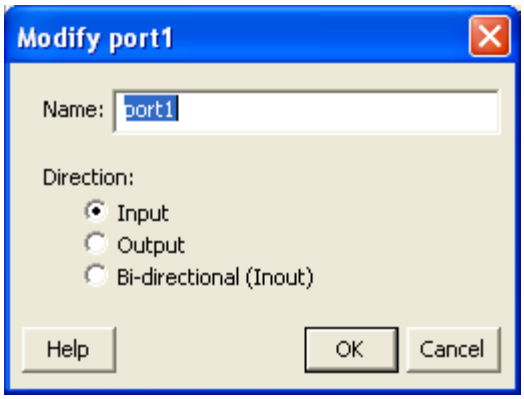



Figure 6 · I/O Attribute Editor Right-click Menu

### *To modify a port in the I/O Attribute Editor:*

1. Select the port in the Grid, right-click and choose **Modify Port**. The Modify Port dialog box appears, as shown in the figure below.




Modify Port Dialog Box

2. Enter the new name for the port, change its direction, and click **OK**.

To remove a port from the I/O Attribute Editor, select the port in the Grid, right-click and choose **Delete Port**.

Note: If you opened a PDC file associated with a module and the ports are loaded from the module, you cannot add, modify, nor delete the ports because they are brought over from the module.

# Saving/Closing a PDC File

To save the current PDC file, from the **File** menu, choose **Save** <*PDC filename*>**.pdc**. The saved file is added to your Libero IDE project. The new constraints will override any existing constraints in this file. Any comments you added

are not saved with this file. Therefore, you may want to save your changes into another file using the Save As menu command.

To save the current PDC file with a different name, choose **Save** <*PDC filename*>**.pdc As**. The **Save As** dialog box appears. Enter a new name for the PDC file.

To close the current PDC file without saving, from the **File** menu, choose **Close**. Click **No** when prompted to save.

You can also close a PDC file by right-clicking the name of the tab in the I/O Attribute Editor and choosing **Close**, as shown in the figure below.

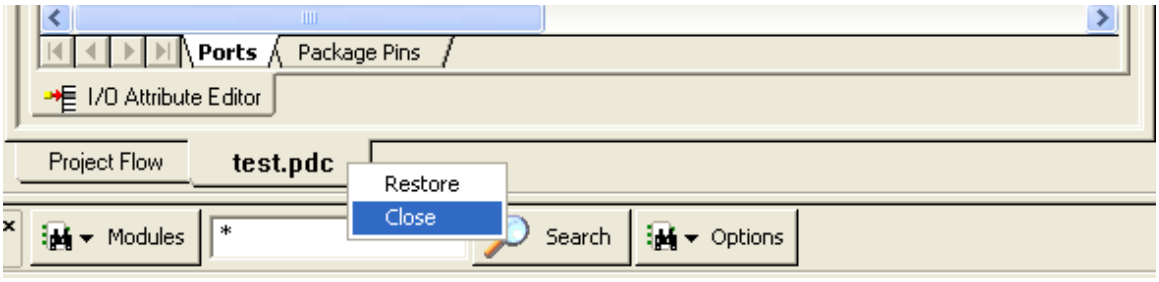



Figure 7 · Close a PDC File

# Deleting a PDC File from a Libero IDE Project

***To delete a PDC file from a project:***

1. In the **Design Explorer**, click the **Files** tab.
2. Select the PDC file that you want to delete. Right-click the file name and select **Delete from Project** or **Delete from Disk and Project**, or click the **Delete** key to delete it from the project.

# Assigning pins in Package Pins View

I/O Attribute Editor (v6.2 and higher) includes a Package Pins view in addition to its Ports view. Click the **Package Pins** tab to display your I/O attributes by package pin number. This view makes it much easier to assign address/data ports to adjacent pins. Additionally, it enables you to assign VREF pins (which you cannot do in Ports view) and to sort on banks.

## Package Pins View

The Package Pins View displays all columns shown in the Ports view plus the following additional columns:

- Function
- Dedicated
- VREF
- User Reserved

**Function** is the functionality of the I/O (for example, GND or ground). See the datasheet for your device for details about each function.

**Dedicated** determines whether the pin is reserved for some special functionality, such as UJTAG / Analog Block / XTL pads inputs.

**VREF** (Voltage referenced), if checked, assigns the selected pin as a VREF. This column only appears for devices that support VREF (IGLOOe, Fusion, ProASIC3L A3PE3000L, ProASIC3E, and Axcelerator). A device supports VREF if one or more of its I/O banks support VREF. IGLOO (excluding IGLOOe) and ProASIC3 (excluding ProASIC3L A3PE3000L and ProASIC3E) devices are not supported.

**User Reserved**, if checked, reserves the pin for use in another design. When a pin is reserved, you cannot assign it to a port. To unreserve the pin, deselect the **User Reserved** check box.

# Editing I/O Attributes

You edit I/O attributes using the I/O Attribute Editor. It displays all assigned and unassigned I/O macros and their attributes in tabular format.

Use the I/O Attribute Editor to view, sort, select, and edit common and device-specific I/O attributes.

You can view the I/O attributes by port or by package pin. Click the **Ports** tab to view I/O attributes by port name. Click the **Package Pins** tab to view I/O attributes by pin number.

Each row corresponds to an I/O macro (port) or a pin in the design, depending on the view displayed. The column headings specify the names of the I/O attributes in your design. The first four column headings are standard for all families so they will not change. However, the other column headings will change depending on the family you are designing for. For some I/O attributes, you will choose from a drop-down menu; for others, you might enter a value.

### *To edit I/O attributes:*

1. Select an I/O standard for each I/O macro in your device.
2. Select I/O attributes that are available for your selected I/O standard.

For descriptions of individual I/O attributes and support by family, refer to the I/O Attributes Reference section of the *Design Constraints Guide*.

## See Also

Editing Multiple Rows

Formatting Rows and Columns

Specifying an I/O Standard

Common I/O Attributes (All Families)

I/O Attributes by Family

# Editing Multiple Rows

***To edit multiple rows:***

1. Select the rows to edit. To select consecutive rows, click the first row, press and hold down the **SHIFT** key, and then click the last row. To select rows that are not consecutive, press and hold down the **CTRL** key, and then click each row to select. Continue to hold down the **SHIFT** or **CTRL** key.
2. While still holding down the **SHIFT** or **CTRL** key, click in the cell containing the value you want to change. Release the **SHIFT** or **CTRL** key, and then release the mouse button.

The change occurs in all selected rows.

Note: Note: You can also select an entire column, which enables you to edit all rows in that column.

## See Also

Editing I/O Attributes

Sorting Attributes

Formatting Rows and Columns

Common I/O attributes (All Families)

I/O Attributes by Family

# Sorting Attributes

You can sort rows by column in either ascending or descending order.

### *To sort I/O macros by attributes:*

- Double-click a column heading to sort the table rows in ascending order.
- Double-click the column again to sort the table rows in descending order.

When sorted, an arrowhead appears in the column header to indicate the sort order.

## See Also

Formatting Rows and Columns

Editing Multiple Rows

# Formatting Rows and Columns

When viewing and editing your input/output attributes, you can format the table to display only the attributes you want to see.

Note: Note: Clicking the top-left cell selects all rows in the I/O Attribute Editor.

### *To hide one or more rows or columns:*

1. Select the row(s) or column(s) you want to hide from view.
2. From the **I/O Attribute Editor> Format** menu, choose **Row > Hide** or **Column > Hide**, or right-click the row or column header and choose **Hide** from the right-click menu.

### *To show a hidden row or column:*

1. Select a range of rows or columns that span one or more hidden rows or columns.
2. From the **I/O Attribute Editor>Format** menu, choose **Row > Unhide** or **Column > Unhide**, or right-click the row or column header and choose **Unhide** from the right-click menu.

Note: Note: Unhide also works for a selected column that has a hidden column to its immediate left or right (or both).

You can "freeze" (or lock) one or more columns so they remain visible on the screen as you scroll horizontally.

### *To freeze or lock one or more columns:*

1. Select the column to the right of the last column to freeze.
2. From the **I/O Attribute Editor>Format** menu, choose **Column > Freeze Pane**, or right-click the column and choose **Freeze Pane** from the right-click menu.

To unfreeze one or more frozen columns, from the **I/O Attribute Editor>Format** menu, choose **Column > Unfreeze Pane**, or right-click any column header and choose **Unfreeze Pane** from the right-click menu. All frozen columns are unfrozen.

You can also resize all the columns and rows at once so their entire contents are visible.

Note: Note: You must unfreeze the current locked group before you can freeze another group.

### *To display a column's entire contents within it:*

1. Select the column(s) you want to display.
2. From the **I/O Attribute Editor>Format** menu, choose **Column >AutoFit**. The width of the column either expands or contracts to fit only the cell heading and cell contents.

## See Also

Sorting Attributes

# Manage Groups

You can group your I/Os by functionality as well as sort the ports by group ID.

You can add new groups and edit existing groups from the I/O Attribute Editor by right-clicking in the **Group** column and choosing **Manage Groups** as shown below.

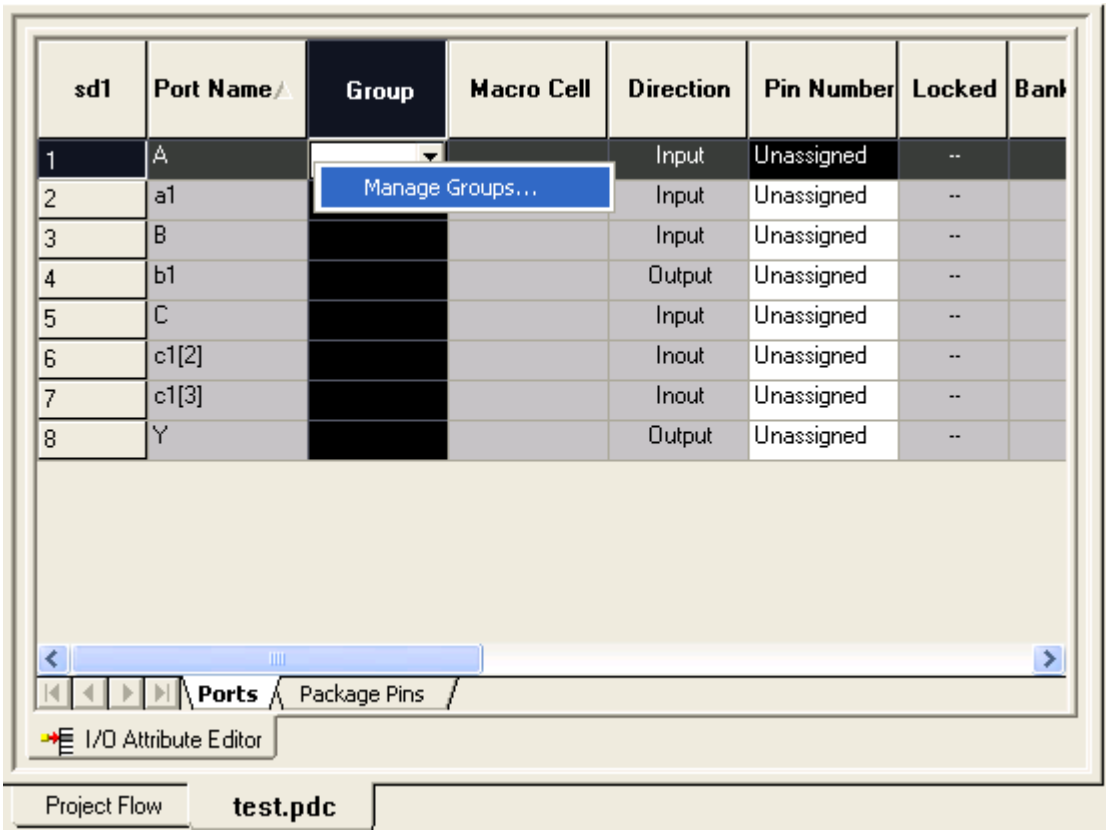



Figure 8 · Manage Groups Command on Right-click Menu

The Manage Groups dialog box appears with a list of existing groups and their descriptions.

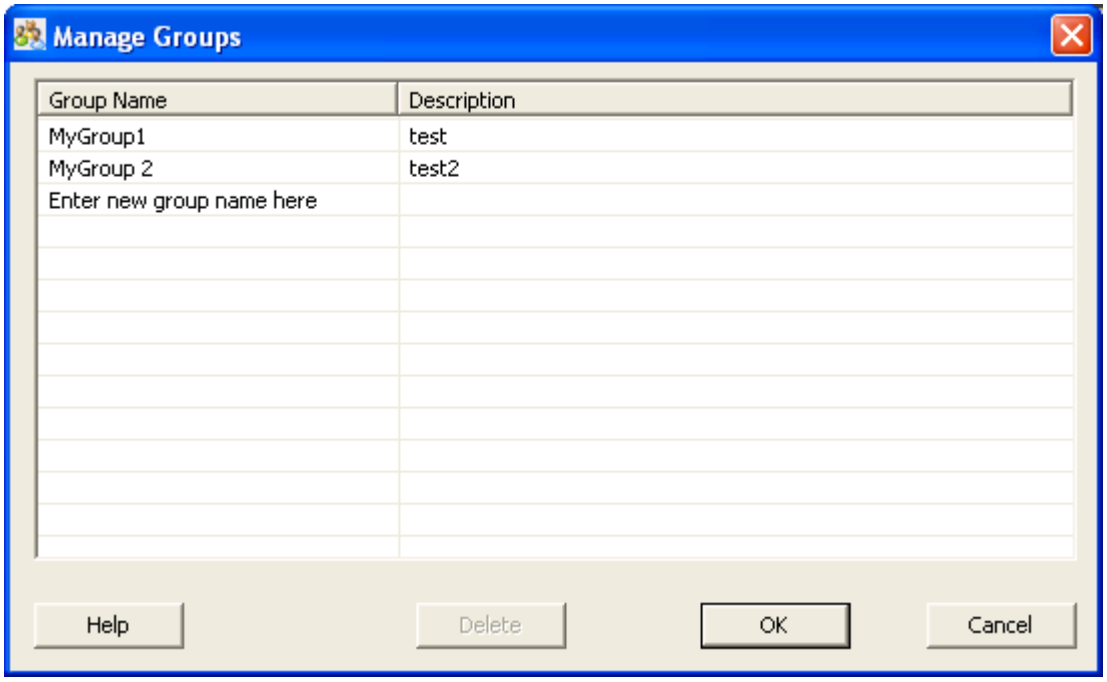



Figure 9 · Manage Groups Dialog Box

You can edit the names and descriptions of any existing group.

### To create a new group:

1. From the **I/O Attribute Editor** menu, choose **Tools>Manage Groups.** The Manage Groups dialog box appears.
2. In the **Manage Groups** dialog box, click in a blank row in the **Group Name** column, and type a name for your new group.
3. Click in the second column and type a description of the new group.
4. Click **OK**.

The new group appears in the drop-down list of each field in the Group column.

### To modify a group:

1. From the **I/O Attribute Editor** menu, choose **Tools>Manage Groups.** The Manage Groups dialog box appears.
2. In the **Manage Groups** dialog box, select the group name or description, and then click once to make it editable. The text to modify appears highlighted and there is an outline around the field as shown below:

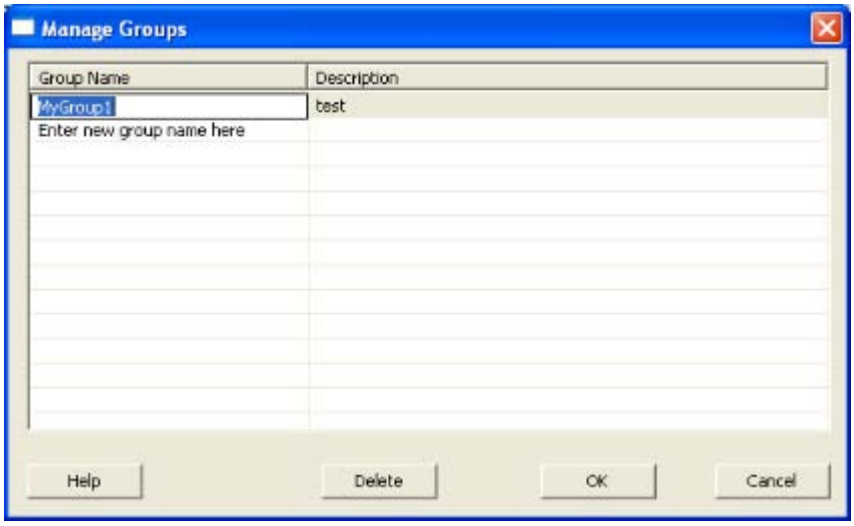



Figure 10 · Editing a Name in the Manage Groups Dialog Box

3. Edit the text.
4. Click **OK**.

### *To delete a group:*

1. From the **I/O Attribute Editor** menu, choose **Tools>Manage Groups.** The Manage Groups dialog box appears.
2. In the **Manage Groups** dialog box, select the row with the group to delete.
3. Click **Delete**.
4. Click **OK**.

Adding, modifying, and deleting groups can be undone using the **Undo** command.

# Manually Assigning Technologies to I/O Banks

The procedure for manually assigning technologies to I/O banks differs depending on whether you are designing for IGLOO, Fusion, ProASIC3, or Axcelerator devices.

### *To assign technologies to I/O banks in IGLOOe, Fusion, ProASIC3L, ProASIC3E, and Axcelerator devices:*

1. Select an I/O bank in either ChipPlanner or PinEditor.
2. From the **Edit** menu, choose **I/O Bank Settings**.
3. In the **I/O Bank Settings** dialog box, select the technologies, and click **Apply**.
   Selecting a standard selects all compatible standards and grays out incompatible ones. For example, selecting LVTTL also selects PCI, PCIX, and LVPECL, since they all have the same VCCI. Further, selecting GTLP (3.3 V) disables SSTL3 as an option because the VREFs of the two are not the same. Once you click **Apply**, the I/O bank is assigned the selected standards. Any I/O of the selected types can now be assigned to that I/O bank. Any previously assigned I/Os in the bank that are no longer compatible with the standards applied are unassigned.
4. Click **More Attributes** to set the low-power mode and input delay. (These attributes are supported in Axcelerator devices only.)
5. Assign I/O standards to other banks by selecting the banks from the list and assigning standards. Any banks not assigned I/O standards use the default standard selected in the Device Selection Wizard.
6. Leave the **Use default pins for VREFs** option selected to set default VREF pins and unset non-default VREF pins. If you unselect this option when setting a new VREF technology, no VREF pins are set. If you unselect this option when default VREF pins are already set, it unsets them.

   If the **Use default pins for VREFs** option is selected when you click **OK** or **Apply**, the software: 1) determines if setting default VREF pins causes any I/O macros to become unassigned, and if so, displays a warning message enabling you to cancel this operation, 2) determines if unsetting non-default VREF pins causes any I/O macros to become unassigned, and if so, displays a warning message enabling you to cancel this operation, and 3) sets default VREF pins and unsets non-default VREF pins.
7. Click **OK**. Using PinEditor, proceed to assign I/Os with the same standards to the appropriate banks.

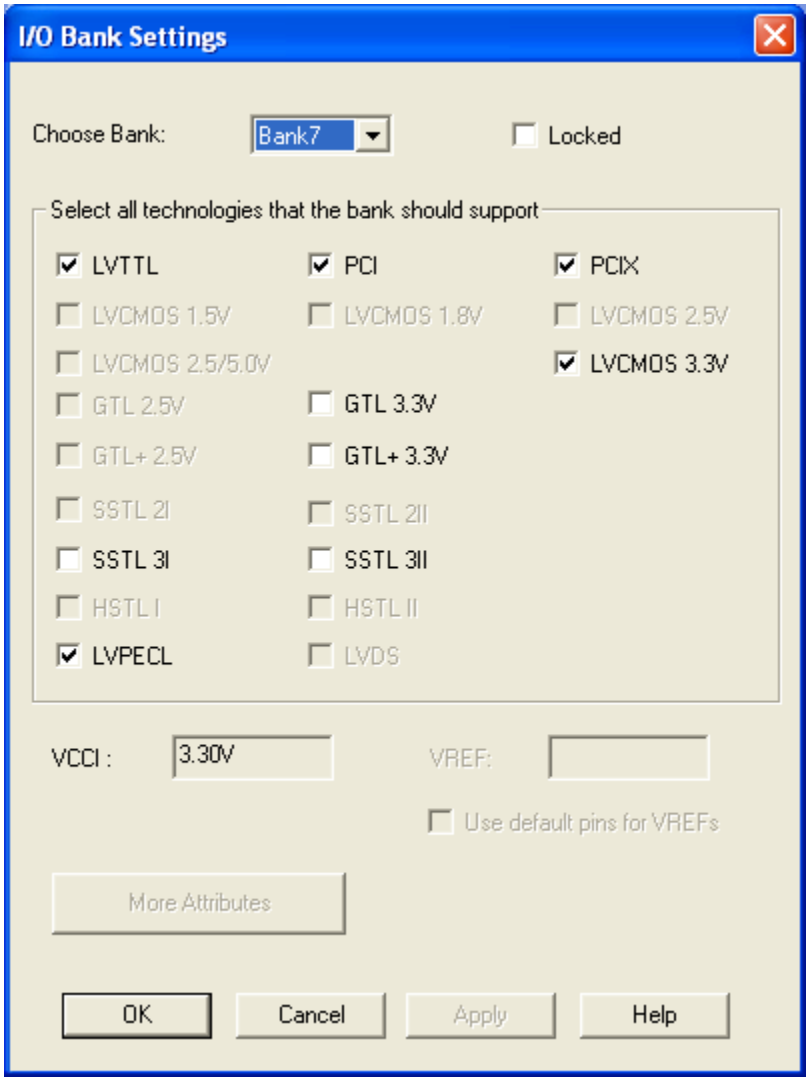



Figure 11 · I/O Bank Settings Dialog Box for IGLOOe, Fusion, ProASIC3L, ProASIC3E, and Axcelerator Devices

If VREF pins can be assigned, you must assign at least one VREF pin before running Layout. See "Assigning VREF Pins" in this guide for more information.

Note: If you use I/O standards that need reference voltage, make sure to assign VREF pins. Actel strongly recommends you use the defaults. VREF pins appear in red in ChipPlanner and are labeled VREF in PinEditor.

### *To set the low-power mode and input delay (for Axcelerator devices only):*

1. Click **More Attributes** in the **I/O Bank Settings** dialog box.
2. Drag the slider bar to the desired delay. The delay is bank specific.
3. Click **View All Delays** to see all the delay values (Best, Worst, Typical, Rise-Rise, Fall-Fall) for the input delay selected. You must select a technology to see the input delays.

4. Click **OK**.

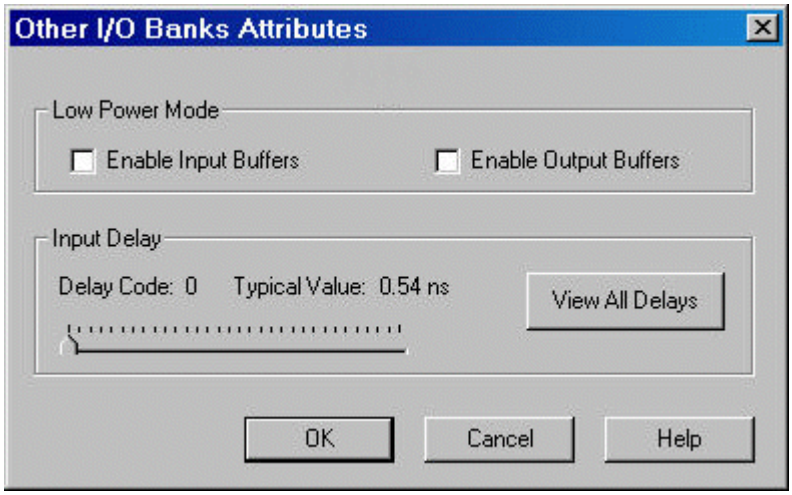



Figure 12 · Other I/O Bank Attributes Dialog Box

Note: To assign technologies to I/O banks in ProASIC3 and IGLOO devices:

1. Select an I/O bank in either ChipPlanner or PinEditor.
2. From the **Edit** menu, choose **I/O Bank Settings**.
3. In the **I/O Bank Settings** dialog box, select the technologies, and click **Apply**.
   Selecting a standard selects all compatible standards and grays out incompatible ones. For example, selecting LVTTL also selects PCI, PCIX, and LVPECL, since they all have the same VCCI. Note that LVDS is available only for banks 1 and 3. Once you click **Apply**, the I/O bank is assigned the selected standards. Any I/O of the selected types can now be assigned to that I/O bank. Any previously assigned I/Os in the bank that are no longer compatible with the standards applied are unassigned.
4. Assign I/O standards to other banks by selecting the banks from the list and assigning standards. Any banks not assigned I/O standards use the default standard selected in the Device Selection Wizard.
5. Click **OK**. Using PinEditor, proceed to assign I/Os with the same standards to the appropriate banks.

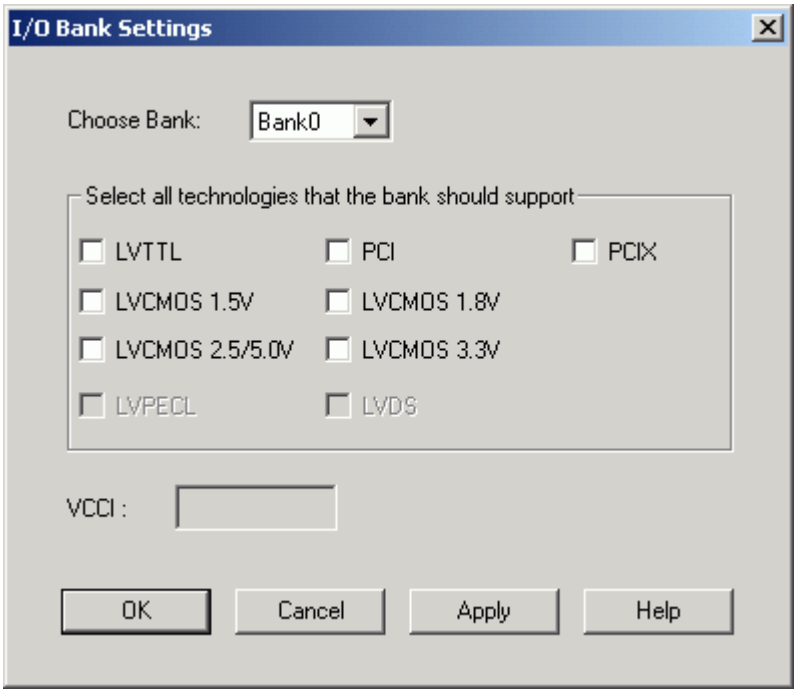



Figure 13 · I/O Bank Settings Dialog Box for IGLOO and ProASIC3 Devices

Note: You cannot assign VREF pins in ProASIC3 and IGLOO devices. You can assign VREF pins only to IGLOOe, Fusion, ProASIC3L (A3PE3000L die only), and ProASIC3E devices.

## See Also

Specifying Technologies for an I/O Bank

Automatically Assigning Technologies to I/O Banks

Assigning Pins in ProASIC3E

Assigning VREF Pins

Displaying VREF Pins

# Automatically Assigning Technologies to I/O Banks

The I/O Bank Assigner (IOBA) tool runs automatically when you run Layout. You can also use this tool from within the MultiView Navigator or from within Libero Project Manager. The I/O Bank Assigner tool automatically assign technologies and VREF pins (if required) to every I/O bank that does not currently have any technologies assigned to it. This tool is available when at least one I/O bank is unassigned.

Each time you run the I/O Bank Assigner, it unassigns all technologies from all I/O banks and then re-assigns them when it finds a feasible solution. To prevent I/O Bank Assigner from unassigning and re-assigning I/O technologies each time you run it, lock the I/O banks by selecting **Locked** in the I/O Bank Settings dialog box or by importing the `set_iobanks` PDC command with its `-fixed` argument set to "yes".

### *To automatically assign technologies to I/O banks:*

- In Project Manager, from the **I/O Attribute Editor** menu, choose **Tools>Auto-Assign I/O Banks**.
- In MultiView Navigator, from the **Tools** menu, choose **Auto-Assign I/O Banks**. You can also click the I/O Bank Assigner's toolbar button shown below.

Messages appear in the Output window informing you when the automatic I/O bank assignment begins and ends. If the assignment is successful, "I/O Bank Assigner completed successfully" appears in the Output window.

If the assignment is not successful, an error message appears in the Output window.

Tip: Tip: Click an underlined "Error" or "Info" message to display more information.

Note: Note: All I/O technologies assigned to I/O banks by the I/O Bank Assigner in Layout are unlocked.

To undo the I/O bank assignments, choose **Undo** from the **Edit** menu. Undo removes the I/O technologies assigned by the I/O Bank Assigner. It does not remove the I/O technologies previously assigned.

To redo the changes undone by the Undo command, choose **Redo** from the **Edit** menu.

If you need to clear I/O bank assignments made before using the Undo command, you can manually unassign or re-assign I/O technologies to banks. To do so, choose **I/O Bank Settings** from the **Edit** menu to display the **I/O Bank Settings** dialog box.

## See Also

About I/O Banks

Specifying Technologies for an I/O Bank

Manually Assigning Technologies to Banks

Using the Prelayout Checker

# Reserving Pins for Device Migration

With this feature, you can begin a design with a larger device that you intend to implement later with a smaller device. Because there might be some pins on the smaller device that are not bonded, you want to make sure that the pin assignments created on the larger device are compatible with the pins on the smaller device. This feature reserves the pins on the larger device that are not bonded on the smaller device.

Pins in the current device that are not bonded in the target device will be marked as "reserved."

You can explicitly reserve a pin in PinEditor or I/O Attribute Editor (Package Pins view). You can also reserve a pin by importing a PDC constraint file with the `reserve` PDC command.

### *To explicitly reserve a pin in PinEditor:*

- Select the pin to reserve, right-click it, and choose **Reserve Pin** from the right-click menu. (See screen below.) Repeat for each pin to reserve.

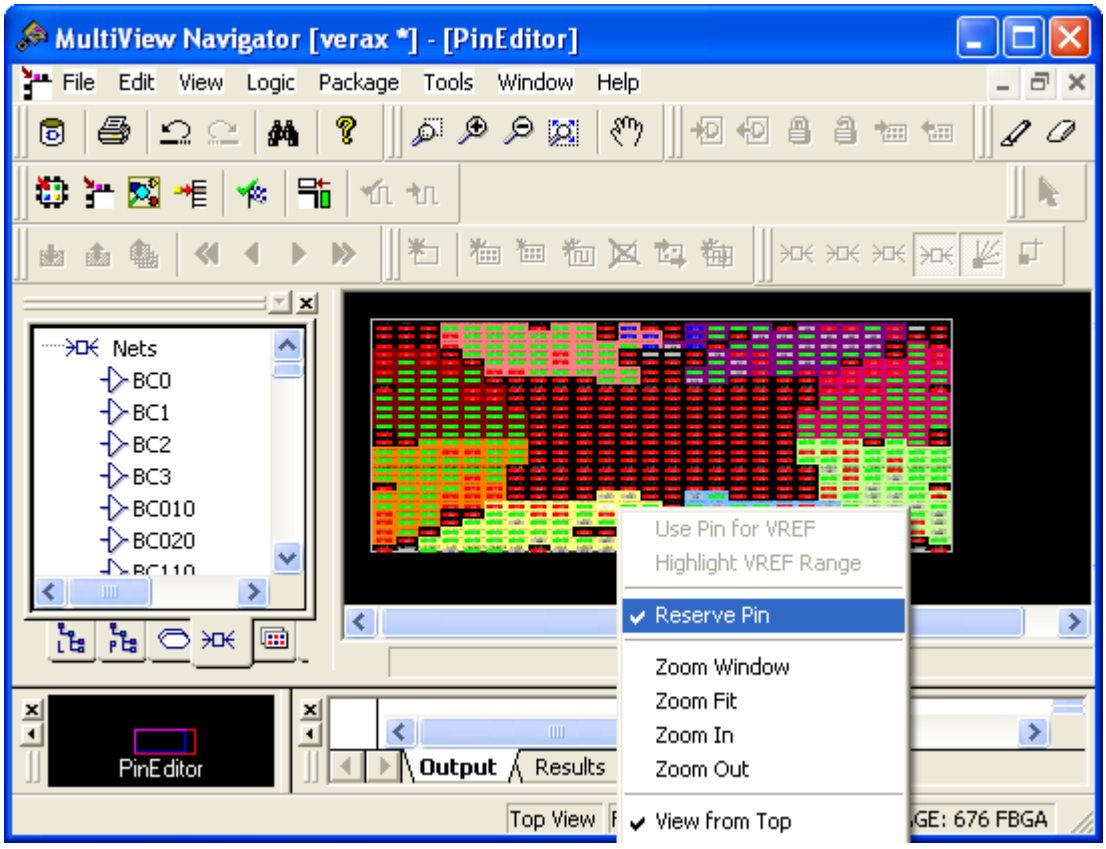



Figure 14 · Figure 15 · Reserve Pins from Right-click Menu in PinEditor

Note: To unreserve a reserved pin from the right-click menu in PinEditor, select the pin to unreserve, right-click it, and choose Reserve Pin to remove the checkmark.

### *To explicitly reserve a pin in I/O Attribute Editor:*

- In **Package Pins** view, select the **User Reserved** check box associated with the pin to reserve. (See screen below.) Repeat for each pin to reserve.

### *To automatically reserve pins that are not bonded in a destination device for migration, follow these steps:*

1. In PinEditor, from the **Edit** menu, choose **Reserve Pins for Migration**. The **Reserve Pins for Migration** dialog box appears. The current device for which the pins will be reserved appears in the **Reserve pins in the current device** text box.
2. From the **"that are not bonded in the target device"** drop-down list, select the target device to which you will be migrating your design.
3. Unselect the **Keep explicitly-reserved pins** check box if you do not want to save the pins that are currently explicitly reserved.

Choose **Undo Reserve Package Pin** from the **Edit** menu to unreserve the last pin you reserved.

### *To reserve pins with a PDC file:*

1. Open the PDC file to edit.
2. Use the `reserve` command to specify the names of the pins to reserve.

### *To unreserve pins with a PDC file:*

1. Open the PDC file to edit.
2. Use the `unreserve` command to specify the names of the pins to unreserve.

Note: Note: SX-A devices do not support the reserved pins feature.

## See Also

`reserve`
`unreserve`

# Specifying an I/O Standard

Use the I/O Standard column to select an I/O specification for each pin.

If required to match the I/O standard, other I/O attributes, such as I/O threshold, slew, and loading, are automatically set to their default settings; you cannot edit these defaults.

You can change the I/O standards only for a generic I/O buffer to any of the legal I/O standards.

### *To specify an I/O standard:*

1. Click the **I/O Standard** cell in the desired macro row.
2. Type or select a supported I/O standard from the drop-down list.

For devices that support I/O banks (for example, Axcelerator devices), the list is restricted to legal choices only. When an I/O is assigned, the I/O standards available for that I/O are limited to what the I/O bank location can support.

Note: Changing an I/O standard may also unassign existing I/Os. In addition, when a macro is assigned an I/O standard, the I/O bank is automatically assigned the voltages VCCI and VREF, if necessary. Unassigning this macro will undo these assignments as well.

## See Also

I/O Attributes by Family

# I/O Attributes

# I/O Attributes by Family or Device

Other than the four  supported by all families, the following table includes the attributes that each Actel family supports. The following table displays the attributes supported for each family.

| Attribute | Family | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | IGLOO | SmartFusion and Fusion | ProASIC3 | ProASIC PLUS | ProASIC | Axcelerator | SX-A | eX |
| Bank Name | X | X | X | | | X | | |
| I/O Standard | X | X | X | | | X | X | X |
| I/O Threshold | X, IGLOO PLUS only | | | | | | X | X |
| Output Drive | X | X | X | | | X | | |
| Slew | X | X | X | | | X | X | X |
| Power Up State | | | | | | | X | X |
| Resistor Pull | X | X | X | | | X | | |
| Schmitt Trigger | X, IGLOOe and IGLOO PLUS only | X | X, ProASIC3e and ProASIC3L only | | | | | |
| Input Delay | X, IGLOOe and IGLOO PLUS only | X | X, ProASIC3e and ProASIC3L only | | | X | | |
| Skew | X | X | X | | | | | |

| Attribute | Family | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **IGLOO** | **SmartFusion and Fusion** | **ProASIC3** | **ProASIC PLUS** | **ProASIC** | **Axcelerator** | **SX-A** | **eX** |
| Output Load | X | X | X | X | X | X | X | X |
| Use Register | X | X | X | | | X | | |
| Hot Swappable | X | X | X | | | X | | X |
| Hold State | X, IGLOO PLUS only | | | | | | | |
| User_Reserved | X | X | X, ProASIC3e and ProASIC3L only | | | X | | |

Refer to the appropriate datasheet for information about I/O standards for different families.

Note: Note: For Fusion devices, not all attributes apply to all banks for a given I/O standard. Refer to the Fusion datasheet for details.

# Bank Name

## Purpose

Displays the name of the bank to which the I/O macro has been assigned. You cannot change the bank name.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

# Direction

## Purpose

Indicates whether the pin is accepting a signal (input), sending a signal (output), or both sending and receiving a signal (Inout).

| Families | Supported |
|---|---|
| IGLOO | Yes |

| Families | Supported |
|---|---|
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

# Group

## Purpose

Indicates whether the port currently belongs to a group.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No |

| Families | Supported |
|---|---|
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

Use this attribute to assign a port to a group or unassign a port from a group.

# Hold State

## Purpose

Preserves the previous state of the I/O. By default, all the I/Os become tristated when the device goes into Flash*Freeze mode. . (A tristatable I/O is an I/O with three output states: high, low, and high impedance.) You can override this default using the hold_state attribute. When you set the hold_state to True, the I/O remains in the same state in which it was functioning before the device went into Flash*Freeze mode.

| Families | Supported |
|---|---|
| IGLOO | IGLOO PLUS only |
| SmartFusion | No |
| Fusion | No |
| ProASIC3 | No |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | No |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

# Hot Swappable

The I/O standard specified and the selected voltage determine this **read-only** attribute.

## Purpose

Indicates whether the I/O pin is hot swappable.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | A3P030 only |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | Yes |
| SX | No |
| RTSX-S | No |
| eX | Yes |
| MX | No |

## Values

If you see either a checkmark or ON (all standards except PCI and PCIX), it means that a clamp diode is NOT included to allow proper hot-swap behavior. If you do not see a checkmark or you see "OFF" (PCI and PCIX only), it means that a clamp diode is included as required by those specifications, but the I/O is NOT hot swappable.

# Input Delay

## Purpose

Indicates whether the input path delay elements are to be programmed. If they will be programmed, this option adds the specified input delay to the input path.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes, excluding ProASIC3 devices |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

## Values

Use this attribute to turn the input delay on or off.

For ProASIC3E devices, you specify the input delay per pin. You will see the actual delay only in Timer or in the SDF file.

Note: The actual input delay is a function of the operating conditions and is automatically computed by the delay extractor when a timing report is generated.

For Axcelerator devices, you specify the input delay per bank. You then set its input delay with the slider in the More I/O Bank AttributesMore I/O Bank Attributes dialog box. Possible values are 0 to 31.

# I/O Standard

## Purpose

Use the I/O standard attribute to assign an I/O standard to an I/O macro.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No * |
| ProASIC | Yes |
| Axcelerator | Yes |
| SX-A | Yes |
| SX | No |
| RTSX-S | Yes |
| eX | No |
| MX | No |

Note: Note: * Supports LVPECL but only on dedicated LVPECL I/Os.

Note: Note: Voltage referenced I/O inputs require an input referenced voltage (VREF). You must assign VREF pins to IGLOOe, ProASIC3E, and Axcelerator devices before running Layout.

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator families support multiple I/O standards (with different I/O voltages) in a single die. You can use I/O Attribute Editor to set I/O standards and attributes, or alternatively you can export and import this information using a PDC file.

Not all devices support all I/O standards. The following table shows you which I/O standards are supported by each device.

| I/O Standard | IGLOO | SmartFusion /Fusion | ProASIC3 | Axcelerator | RTSX-S | SX-A |
|---|---|---|---|---|---|---|
| CMOS | | | | | X | |
| CUSTOM | | | | | X | X |
| GTL+ | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| GTL 3.3 V | IGLOOe only | X | ProASIC3E and ProASIC3L only | | | |
| GTL 2.5 V | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| HSTL Class I | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| HSTL Class II | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| LVCMOS 3.3 V | IGLOOe only | X | ProASIC3E and ProASIC3L only | | | |
| LVCMOS 2.5 V | X | X | X | | | |
| LVCMOS 2.5 V/5.0V | IGLOOe only | X | X | X | | |

| I/O Standard | IGLOO | SmartFusion /Fusion | ProASIC3 | Axcelerator | RTSX-S | SX-A |
|---|---|---|---|---|---|---|
| LVCMOS 1.8 V | X | X | X | | | |
| LVCMOS 1.5 V | X | X | X | X | | |
| LVCMOS 1.2 V | X | X | ProASIC3 (A3PL), IGLOOe V2 only, IGLOO V2, and IGLOO PLUS only | X | | |
| LVDS | IGLOO and IGLOO PLUS only | | ProASIC3L only | | | |
| LVPECL | X | X | X | X | X | X |
| LVTTL/TTL | X | X | X | X | X | X |
| PCI | X | X | X | X | X | X |
| PCI-X 3.3 V | X | X | X | X | | |
| SSTL2 Class I and II | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| SSTL3 Class I and II | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |

Note: Note:

*Supported only on dedicated LVPECL I/Os.

Note: Note: For a list of I/O standards for all other families, refer to the datasheet for your specific device.

## Descriptions

Following are brief descriptions of the I/O standard attributes in the table above:

### CMOS (Complementary Metal-Oxide-Semiconductor)

An advanced integrated circuit (IC) manufacturing process technology for logic and memory, characterized by high integration, low cost, low power, and high performance. CMOS logic uses a combination of p-type and n-type metal-oxide-semiconductor field effect transistors (MOSFETs) to implement logic gates and other digital circuits found in computers, telecommunications, and signal processing equipment.

### CUSTOM

An option in the I/O Attribute Editor that enables you to customize individual I/O settings such as the I/O threshold, output slew rates, and capacitive loadings on an individual I/O basis. For example, PCI mode output can be set to low-slew rate. For more information, go to the Actel web site and check the datasheet for your device.

### GTL 2.5 V (Gunning Transceiver Logic 2.5 Volts)

A low-power standard (JESD 8.3) for electrical signals used in CMOS circuits that allows for low electromagnetic interference at high speeds of transfer. It has a voltage swing between 0.4 volts and 1.2 volts, and typically operates at speeds of between 20 and 40MHz. The VCCI must be connected to 2.5 volts.

### GTL 3.3 V (Gunning Transceiver Logic 3.3 Volts)

Same as GTL 2.5 V above, except the VCCI must be connected to 3.3 volts.

### GTL+ (Gunning Transceiver Logic Plus)

An enhanced version of GTL that has defined slew rates and higher voltage levels. It requires a differential amplifier input buffer and an open-drain output buffer. Even though output is open-drain, the VCCI must be connected to either 2.5 volts or 3.3 volts for IGLOO, Fusion, ProASIC3, and Axcelerator families.

### HSTL Class I and II (High-Speed Transceiver Logic)

A general-purpose, high-speed 1.5 V bus standard (EIA/JESD 8-6) for signalling between integrated circuits. The signalling range is 0 V to 1.5 V, and signals can be either single-ended or differential. HSTL requires a differential amplifier input buffer and a push-pull output buffer. It has four classes, of which Actel supports Class I and II. These classes are defined by standard EIA/JESD 8-6 from the Electronic Industries Alliance (EIA):

- Class I (unterminated or symmetrically parallel terminated)
- Class II (series terminated)
- Class III (asymmetrically parallel terminated)
- Class IV (asymmetrically double parallel terminated

### LVCMOS 3.3 V (Low-Voltage CMOS for 3.3 Volts)

An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 3.3 V applications.

### LVCMOS 2.5 V (Low-Voltage CMOS for 2.5 Volts)

An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 2.5 V applications.

### LVCMOS 2.5 V/5.5V (Low-Voltage CMOS for 2.5 and 5.0 Volts)

An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 2.5 V and 5.0V applications.

### LVCMOS 1.8 V (Low-Voltage CMOS for 1.8 Volts)

An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.8 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer.

### LVCMOS 1.5 V (Low-Voltage CMOS for 1.5 volts)

An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.5 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer.

### LVCMOS 1.2 V (Low-Voltage CMOS for 1.2 volts)

Note: Note: An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.2 V applications.

Note: Note: 1.2 voltage is supported for ProASIC3 (A3PL), IGLOOe V2 only, IGLOO V2, and IGLOO PLUS.

### LVDS (Low-Voltage Differential Signal)

A moderate-speed differential signalling system, in which the transmitter generates two different voltages which are compared at the receiver. It requires that one data bit be carried through two signal lines; therefore, you need two pins per input or output. It also requires an external resistor termination. The voltage swing between these two signal lines is approximately 350mV (millivolts). Axcelerator devices contain dedicated circuitry supporting a high-speed LVDS standard that has its own user specification.

### LVPECL (Low-Voltage Positive Emitter Coupled Logic)

PECL is another differential I/O standard. It requires that one data bit is carried through two signal lines; therefore, two pins are needed per input or output. It also requires an external resistor termination. The voltage swing between these two signal lines is approximately 850mV. When the power supply is +3.3 V, it is commonly referred to as low-voltage PECL (LVPECL).

### LVTTL/TTL (Low-Voltage Transitor-Transistor Level)

A general purpose standard (EIA/JESDSA) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer.

### PCI (Peripheral Component Interface)

A computer bus for attaching peripheral devices to a computer motherboard in a local bus. This standard supports both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5V-compliant for most families, excluding ProASIC3 families.

### PCI-X (Peripheral Component Interface Extended)

An enhanced version of the PCI specification that can support higher average bandwidth; it increases the speed that data can move within a computer from 66 MHz to 133 MHz. PCI-X is backward-compatible, which means that devices can operate at conventional PCI frequencies (33 MHz and 66 MHz). PCI-X is also more fault-tolerant than PCI.

### SSTL2 Class I and II (Stub Series Terminated Logic 2.5 V)

A general-purpose 2.5 V memory bus standard (JESD 8-9) for driving transmission lines. This standard was designed specifically for driving the DDR (double-data-rate) SDRAM modules used in computer memory. It requires a differential amplifier input buffer and a push-pull output buffer. It has two classes, of which Actel supports both.

### SSTL3 Class I and II (Stub Series Terminated Logic for 3.3 V)

A general-purpose 3.3 V memory bus standard (JESD 8-8) for driving transmission lines.

# I/O Threshold (or Output Level)

## Purpose

Indicates the compatible threshold level for inputs and outputs.

| Families | Supported |
|---|---|
| IGLOO | No |
| SmartFusion | No |
| Fusion | No |
| ProASIC3 | No |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | No |
| SX-A | Yes |
| SX | Yes |
| RTSX-S | Yes |
| eX | Yes |
| MX | Yes |

* For SX, there is an Output Level option, which is the same as the threshold option. See Output level for more information.

## Values

Use this attribute to set the compatible threshold level for inputs and outputs. The values you can choose from depend on which device you selected. The default I/O threshold displayed is based upon the I/O standard. If you want to set the I/O threshold independently of the I/O specification, you must select CUSTOM in the I/O standard cell.

# Locked

## Purpose

Indicates whether you can change the current pin assignment during layout.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | Yes |
| ProASIC | Yes |
| Axcelerator | Yes |
| SX-A | Yes (Fixed) |
| SX | Yes (Fixed) |
| RTSX-S | Yes (Fixed) |
| eX | Yes (Fixed) |
| MX | Yes (Fixed) |

## Values

Use this attribute to lock or unlock the pin assignment. Selecting the check box locks the pin assignment. Clearing the check box unlocks the pin assignment. If locked, you cannot change the pin assignment. If not locked, you can.

The term "fixed" for SX-A, SX, RTSX-S, eX,and MX devices means "locked."

# Macro Cell

## Purpose

Indicates the type of I/O macro. This value is read only and is applicable only to the I/O Attribute Editor tool (that is, you cannot use it in GCF or PDC files).

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | Yes |
| ProASIC | Yes |
| Axcelerator | Yes |
| SX-A | Yes |
| SX | Yes |
| RTSX-S | Yes |
| eX | Yes |
| MX | Yes |

# Output Drive

## Purpose

Every I/O standard has an output drive preset; however, for some I/O standards, you can choose which one to use. The higher the drive, the faster the I/O. The faster the I/O, the more power consumed by the I/O.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

## Values

Use this attribute to set the strength of the output buffer to between 2 and 24 mA, weakest to strongest, depending on your device family. The LVTTL output buffer has four programmable settings of its drive strength. Other I/O standards have full strength.

The list of I/O standards for which you can change the output drive and the list of values you can assign for each I/O standard is family-specific. Refer to the datasheet for your device for more information.

# Output Load or Loading (pf)

Note: In MVN tools, the column heading for this attribute is "Output load." In non-MVN tools, the column heading for this attribute is "Loading (pf)."

## Purpose

Indicates the output-capacitance value based on the I/O standard selected in the I/O Standard cell. This option is not available in software.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | Yes |
| ProASIC | Yes |
| Axcelerator | Yes |
| SX-A | Yes |
| SX | Yes |
| RTSX-S | Yes |
| eX | Yes |
| MX | Yes |

## Values

You can enter a capacitative load as an integral number of picofarads. The default value varies by device family. If necessary, you can change the output capacitance default setting to improve timing definition and analysis. Both the capacitive loading on the board and the Vil/Vih trip points of driven devices affect output-propagation delay.

Timer, Timing-Driven Layout, Timing Report, and Back-Annotation automatically uses the modified delay model for delay calculations.

# Output Level

## Purpose

Use the Output Level attribute to assign an I/O output level to an I/O pin.

The I/O pin functions as an input, output, tristate, or bidirectional buffer. Based on certain configurations, input and output levels are compatible with standard TTL, LVTTL, 3.3 V PCI or 5.0V PCI specifications. Unused I/O pins are automatically tristated by the Designer software.

| Families | Supported |
|---|---|
| IGLOO | No |
| SmartFusion | No |
| Fusion | No |
| ProASIC3 | No |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | No |
| SX-A | No |
| SX | Yes |
| RTSX-S | No |
| eX | No |
| MX | Yes |

## Values

LVTTLCMOS, or PCI.

## Default value

LVTTL

# Pin Number

## Purpose

Use this attribute to change a pin assignment by choosing one of the legal values from the drop-down list. If the pin has been assigned, the pin number appears in this column. If it hasn't been assigned, "Unassigned" appears in this column.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | Yes |
| ProASIC | Yes |
| Axcelerator | Yes |
| SX-A | Yes |
| SX | Yes |
| RTSX-S | Yes |
| eX | Yes |
| MX | Yes |

# Port Name

## Purpose

Indicates the port name of the I/O macro. This value is read only.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | Yes |
| ProASIC | Yes |
| Axcelerator | Yes |
| SX-A | Yes |
| SX | Yes |
| RTSX-S | Yes |
| eX | Yes |
| MX | Yes |

# Power-up State

## Purpose

Indicates the power-up state of the pin. All I/Os are equipped with pull-up and pull-down resistors, which are enabled during power-up. These resistors are disabled just before VCCA reaches 2.5 V, and then the I/Os behave according to the design. For eX and SX-A, this configurable I/O state does not eliminate the risk of an I/O driving a temporary unknown state near the end of the power-up sequence when VCCI is powered up before VCCA. For RTSX-S, the outputs will drive according to the design, when the resistors become disabled regardless of the power-up sequence.

| Families | Supported |
|---|---|
| IGLOO | No |
| SmartFusion | No |
| Fusion | No |
| ProASIC3 | No |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | No |
| SX-A | Yes |
| SX | No |
| RTSX-S | Yes |
| eX | Yes |
| MX | No |

## Values

Use this attribute to set the power-up state. Your choices are None, High, and Low. The default value is None. The only exception to this is an I/O that exists in the netlist as a port, is not connected to the core, and is configured as an Output Buffer. In that case, the default setting will be Low.

# Resistor Pull

## Purpose

Allows inclusion of a weak resistor for either pull-up or pull-down of the input buffer.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

## Values

Use this attribute to set the resistor pull. Your choices are None, Up (pull-up), or Down (pull-down). The default value is None except when an I/O exists in the netlist as a port, is not connected to the core, and is configured as an output buffer. In that case, the default setting is for a weak pull-down.

# Schmitt Trigger

## Purpose

A schmitt trigger is a buffer used to convert a slow or noisy input signal into a clean one before passing it to the FPGA. This is a simple, low-cost solution for a user working with low slew-rate signals. Using schmitt-trigger buffers guarantees a fast, noise-free, input signal to the FPGA.

Actel recommends that you use a schmitt trigger to buffer a signal if input slew rates fall below the values outlined in the specification for SX-A and RTSX-S devices. Depending on the application, different schmitt-trigger buffers can be used to fulfill the requirements.

Schmitt-trigger buffers are categorized in three configurations:

- Fixed threshold voltages with non-inverted outputs
- Fixed threshold voltages and inverted outputs
- Variable threshold voltages with non-inverted outputs

With the aid of schmitt-trigger buffers, low slew-rate applications can also be handled with ease. Implementation of these buffers is simple, not expensive, and flexible in that different configurations are possible depending on the application. The characteristics of schmitt-trigger buffers (e.g. threshold voltage) can be fixed or user-adjustable if required.

| Families | Supported |
|---|---|
| IGLOO | Yes, IGLOOe and IGLOO PLUS only |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes, with one exception: this attribute is not supported in ProASIC3L except in A3PE3000L |
| ProASIC PLUS | Yes* |
| ProASIC | No |
| Axcelerator | No |
| SX-A | No |

| Families | Supported |
|---|---|
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

*Although ProASIC PLUS supports the schmitt-trigger attribute,you cannot edit this attribute with the MultiView Navigator tools. Instead, it has to be instantiated in the schematic or the netlist.

## Values

A schmitt trigger has two possible states: on or off. The trigger for this circuit to change states is the input voltage level. That is, the output state depends on the input level, and will change only as the input crosses a pre-defined threshold.

For more information, please see the "Using Schmitt Triggers for Low Slew-Rate Input" Application Note on the Actel web site.

# Skew

## Purpose

Indicates whether there is a fixed additional delay between the enable/disable time for a tristatable I/O. (A tristatable I/O is an I/O with three output states: high, low, and high impedance.) 2 ns delay on rising edge, 0 ns delay on falling edge.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | No |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

## Values

You can set the skew for a clock to either on or off.

Note: A Tri State or "tristatable" logic gate has three output states: high, low, and high impedance. In a high impedance state, the output acts like a resistor with infinite resistance, which means the output is disconnected from the rest of the circuit.

# Slew

The slew rate is the amount of rise or fall time an input signal takes to get from logic low to logic high or vice versa. It is commonly defined to be the propagation delay between 10% and 90% of the signal's voltage swing.

## Purpose

Indicates the slew rate for output buffers. Generally, available slew rates are high and low.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | Yes |
| SX | Yes |
| RTSX-S | Yes |
| eX | Yes |
| MX | No |

## Values

You can set the slew rate for the output buffer to either **high** or **low**. The output buffer has a programmable slew rate for both high-to-low and low-to-high transitions. The low slew rate is incompatible with 3.3 V PCI requirements.

For ProASIC3 families, you can edit the slew for designs using LVTTL, all LVCMOS, or PCIX I/O standards. The other I/O standards have a preset slew value. For the Axcelerator family, you can edit the slew only for designs using the LVTTL I/O standard. For those devices that support additional slew values, Actel recommends that you use the high and low values

and let the software map to the appropriate absolute slew value. The default slew displayed in the I/O Attribute Editor is based on the selected I/O standard. For example, PCI mode sets the default output slew rate to High.

**One way to eliminate problems with low slew rate is with external schmitt triggers.**

In some applications, you may require a very fast (i.e. high slew rate) signal, which approaches an ideal switching transition. You can accomplish this by either reducing the track resistance and/or capacitance on the board or increasing the drive capability of the input signal. Both of these options are generally time consuming and costly. Furthermore, the closer the input signal approaches an ideal one, the greater the likelihood of unwanted effects such as increased peak current, capacitive coupling, and ground bounce. In many cases, you may want to incorporate a finite amount of slew rate into your signal to reduce these effects. On the other hand, if an input signal becomes too slow (i.e. low slew rate), then noise around the FPGA's input voltage threshold can cause multiple state changes. During the transition time, both input buffer transistors could potentially turn on at the same time, which could result in the output of the buffer to oscillate unpredictably. In this situation, the input buffer could still pass signals. However, these short, unpredictable oscillations would likely cause the device to malfunction. Actel has performed reliability tests on RTSX-S devices and the reliability of the device is guaranteed for signals with slew rates up to 500µs. This test has not been performed on the SX-A family. For more information, see the *RTSX-S TR/TF Experiment* report on the Actel web site.

# Use Register

## Purpose

The input and output registers for each individual I/O can be activated by selecting the check box associated with an I/O. The I/O registers are NOT selected by default.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

## See Also

I/O Register Combining Rules

# Explicitly Reserved

## Purpose

You can explicitly reserve a pin in one of the following ways:

- In the I/O Attribute Editor (Package Pins view), select the **User Reserved** check box associated with the pin to reserve.
- Select a pin in PinEditor, right-click it, and choose Reserve Pin from the right-click menu.
- User the reserve command in a PDC constraint file.

| Families | Supported |
|---|---|
| IGLOO | Yes |
| SmartFusion | Yes |
| Fusion | Yes |
| ProASIC3 | Yes |
| ProASIC PLUS | No |
| ProASIC | No |
| Axcelerator | Yes |
| SX-A | No |
| SX | No |
| RTSX-S | No |
| eX | No |
| MX | No |

# Using I/O Attribute Editor with SmartDesign

You can use the I/O Attribute Editor to assign I/Os in SmartDesign, and SmartDesign will automatically generate the PDC file and pass it to Designer.

When you open an existing PDC file that was generated by SmartDesign, the SmartDesign associated with that PDC file automatically opens.

Note:

# Enabling the I/O Attribute Editor in SmartDesign

***To enable the I/O Attribute Editor in SmartDesign:***

1. From the **SmartDesign** menu, choose **Show I/O Attribute Editor View**. The I/O Attribute Editor opens in front of the SmartDesign Canvas (as shown below).

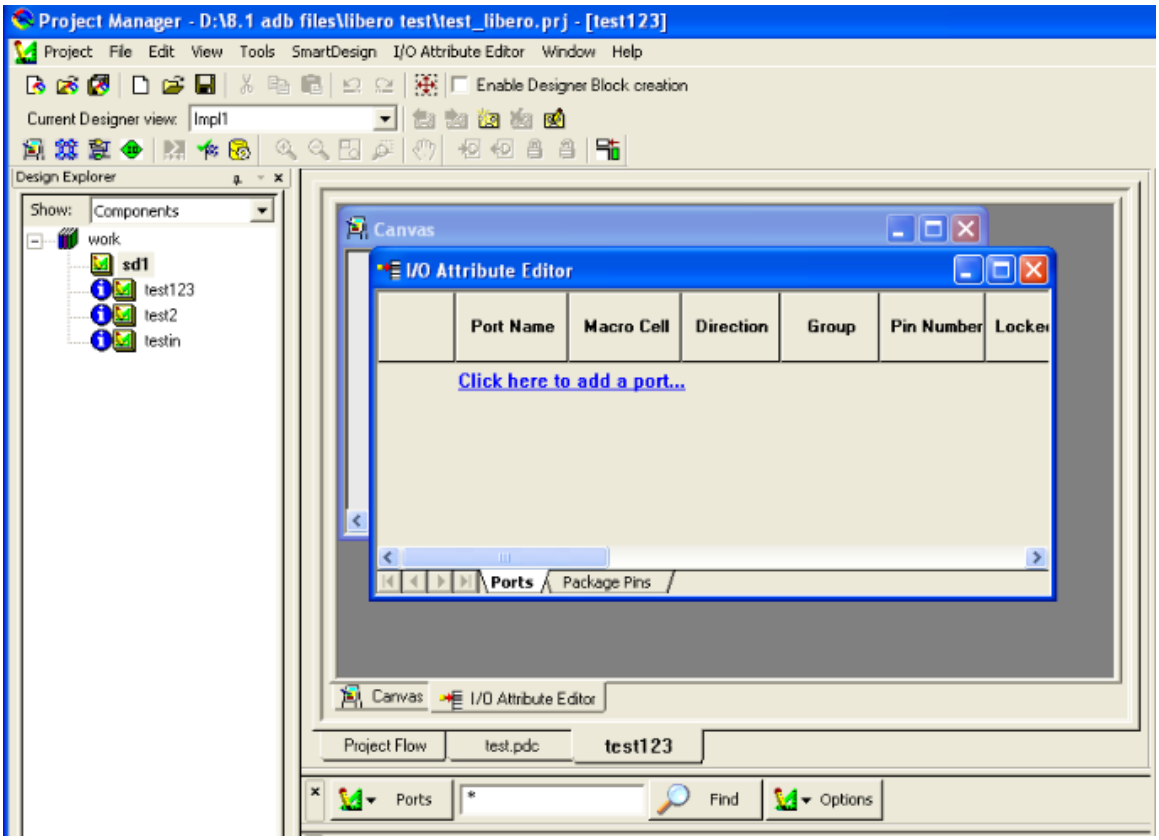



Figure 16 · I/O Attribute Editor in SmartDesign

2. Click the **Canvas** tab, and then select the component to modify.
3. Click the **I/O Attribute Editor** tab to modify the I/O attributes of the selected component.

# Using the I/O Attribute Editor with the Canvas

When you open a file in SmartDesign, the port list is populated.

If a port is added, modified, or deleted from the SmartDesign Canvas or Grid, it is automatically updated in the I/O Attribute Editor.

The I/O constraints are automatically passed from SmartDesign to Designer through a PDC file.

# I/O Attribute Editor Menu

This menu is available from the I/O Attribute Editor in Libero IDE.

| Command | Icon | Shortcut | Function |
|---|---|---|---|
| Add Port | | | Displays the Add New Port dialog box, in which you can add a new port to the design |
| Modify Port | | | Displays the Modify Port dialog box, in which you can change the name and direction of the port |
| Delete | | | Deletes the selected port |
| Unassign from Location | | CTRL + SHIFT + K | Unassigns the selected port from its current location |
| Unassign All from Location | | | Unassigns all ports from their locations |
| Lock |  | CTRL + L | Locks the selected port |
| Unlock |  | CTRL + SHIFT + L | Unlocks the selected locked port |
| Lock All | | | Locks all I/Os in your design |
| Unlock All | | | Unlocks all I/Os in your design |
| Import I/O Assignments | | | Displays the Open dialog box, in which you select the file to import |

## I/O Attribute Editor > Tools

| Command | Icon | Shortcut | Function |
|---|---|---|---|
| I/O Bank Settings | | CTRL + I | Displays the I/O Bank Settings dialog box, in which you can assign technologies and VREF pins to your I/O |

| Command | Icon | Shortcut | Function |
|---|---|---|---|
| | | | banks |
| Auto-Assign I/O Banks |  | | Assigns a voltage to every I/O bank that does not have a voltage assigned to it and if required, a VREF pin |
| Reserve Pins for Device Migration | | CTRL + M | Displays the Reserve Pins for Migration dialog box. This dialog box enables you to automatically reserve pins that are not bonded in the destination device that you select. |
| Manage Groups | | | Creates and manages the groups in the I/O Attribute Editor. You can also use this command to assign ports to a group. |
| Remove All Constraints | | | Removes all constraints from all ports. You can use the Undo command to put them back. |

## I/O Attribute Editor > Format

| Command | Icon | Shortcut | Function |
|---|---|---|---|
| Row > Hide | | | Hide the selected row(s) |
| Row > Unhide | | | Show all hidden rows between the selected rows |
| Column > Hide | | | Hide the selected column(s) |
| Column > Unhide | | | Show all hidden columns between the selected columns |
| Column > Freeze Pane | | | Freeze the selected columns(s) |
| Column > Unfreeze Pane | | | Unfreeze all columns(s) |
| Column > Autofit | | | Sets the width of columns within the table to accommodate all the text for any given row in those columns, including the column headings |

# Add New Port Dialog Box

To access this dialog box, from the **I/O Attribute Editor** menu, choose **Add Port**. You can also right-click a row in the **Ports** tab of the **I/O Attribute Editor**, and choose **Add New Port** to display this dialog box.

Use this dialog box to add a new port to your design.

## Name

Enter a name for the new port.

## Direction

Select one of the following options:

### Input

Select this option if the port is to receive a signal.

### Output

Select this option if the port is to send a signal.

### Bi-directional (Inout)

Select this option if the port will both send and receive a signal.

# Modify Port Dialog Box

To access this dialog box, from the **I/O Attribute Editor** menu, choose **Modify Port**. You can also right-click a row in the **Ports** tab of the **I/O Attribute Editor**, and choose **Modify Port** to display this dialog box.

Use this dialog box to modify the name or direction of an existing port in your design.

## Name

Enter a new name for the port.

## Direction

Select one of the following options:

### Input

Select this option if the port is to receive a signal.

### Output

Select this option if the port is to send a signal.

### Bi-directional (Inout)

Select this option if the port will both send and receive a signal.

# I/O Bank Settings Dialog Box (IGLOO and ProASIC3 only)

To access this dialog, from the **Edit** menu, choose **I/O Bank Settings**.

Use this dialog box to assign I/O technologies to I/O banks in IGLOO (excluding IGLOOe) and ProASIC3 (excluding ProASIC3L and ProASIC3E) devices.

## Choose Bank

Choose a bank from the drop-down list. If you do not assign I/O standards to a bank, that bank uses the default standard selected in the Device Selection Wizard.

## Locked

Select this option to lock all I/O banks, so the I/O Bank Assigner cannot unassign and re-assign the technologies in your design.

## Select All Technologies That the Bank Should Support

Selecting an I/O standard selects all compatible standards and grays out incompatible ones. For example, selecting LVTTL also selects PCI, PCIX, and LVPECL, since they all have the same VCCI. Further, selecting GTLP (3.3V) disables SSTL3 as an option because the VREFs of the two are not the same.

## VCCI

Each I/O bank has a common supply voltage, VCCI, for the I/Os within that bank.

Click **Apply** to assign the selected I/O standards to the selected bank. Any previously assigned I/Os in the bank that are no longer compatible with the standards applied are unassigned.

## See Also

Manually Assigning Technologies to I/O Banks

Assigning VREF Pins

# I/O Bank Settings Dialog Box

To access this dialog, from the **Edit** menu, choose **I/O Bank Settings**.

Use this dialog box to assign I/O technologies to I/O banks in IGLOOe, Fusion, ProASIC3L, ProASIC3E, and Axcelerator devices.

## Choose Bank

Choose a bank from the drop-down list. If you do not assign I/O standards to a bank, that bank uses the default standard selected in the Device Selection Wizard.

## Locked

Select this option to lock all I/O banks, so the I/O Bank Assigner cannot unassign and re-assign the technologies in your design.

## Select All Technologies That the Bank Should Support

Selecting an I/O standard selects all compatible standards and grays out incompatible ones. For example, selecting LVTTL also selects PCI, PCIX, and LVPECL, since they all have the same VCCI. Further, selecting GTLP (3.3V) disables SSTL3 as an option because the VREFs of the two are not the same.

## VCCI

Each I/O bank has a common supply voltage, VCCI, for the I/Os within that bank. (Technologies not allowed for the selected VCCI appear grayed out.)

## VREF

A voltage referenced I/O input (VREF) requires an input referenced voltage. You must assign VREF pins to IGLOOe, Fusion, ProASIC3L (A3PE3000L only), ProASIC3E, and Axcelerator devices before running Layout.

Note: You cannot assign VREF pins in IGLOO or ProASIC3 low-cost devices.

## Use Default Pins for VREFs

Select this check box to set default VREF pins and unset non-default VREF pins. If you unselect this option when setting a new VREF technology, no VREF pins are set. If you unselect this option when default VREF pins are already set, it unsets them.

Click **More Attributes**to set the low-power mode and input delay. (These attributes are not supported in IGLOOe, Fusion, ProASIC3E, or RTAXS devices.)

Click **Apply** to assign the selected I/O standards to the selected bank. Any previously assigned I/Os in the bank that are no longer compatible with the standards applied are unassigned.

## See Also

Manually Assigning Technologies to I/O Banks

Assigning Pins in IGLOOe, Fusion, and ProASIC3E

Assigning VREF Pins

# I/O Bank Settings for the SmartDesign Microcontroller Subsystem (MSS)

To access the I/O Bank settings in your MSS design you must click the **I/O Editor tab** in the MSS configurator, and from the **I/O Editor Menu** choose **I/O Bank Settings**.

You can use the I/O Bank Settings dialog box to change the VCCI of the banks where the MSS I/Os are placed.

Note: Note: These settings cannot be changed in Designer software.

You have four options:

- 1.50V
- 1.80V
- 2.50V
- 3.30V

East MSS I/Os refer to Bank2.

West MSS I/Os refer to Bank4.

When changing the VCCI the MSS I/Os placed on this bank will change the IoTech to match the new VCCI; this is done automatically.

The IoTech is changed as follows:

- 3.30V: MSS I/Os placed on this bank are changed to LVTTL.
- 2.50V: MSS I/Os placed on this bank are changed to LVCMOS 2.5V.
- 1.80V: MSS I/Os placed on this bank are changed to LVCMOS 1.8V.
- 1.50V: MSS I/Os placed on this bank are changed to LVCMOS 1.5V.

# More I/O Bank Attributes Dialog Box

To access this dialog, from the **Edit** menu, choose **I/O Bank Settings**, and then click **More Attributes.**

Use this dialog box to configure these low-power mode settings:

- Enable/disable input/output buffers
- Input delay for a bank

Although designed for high performance, the Axcelerator architecture also allows you to place the device into a low-power (LP) mode via a dedicated LP pin. Asserting the LP pin, which is grounded in normal operation, activates LP mode on all the I/O banks. When LP mode is activated, I/O banks are disabled (inputs disabled, outputs tristated), and PLLs are placed in a power-down mode. All internal register states are maintained in this mode. Furthermore, you can configure individual I/O banks to opt out of the LP mode, giving you access to critical signals while the rest of the chip is in LP mode.

Using the following options in the **More Attributes** dialog box, you can individually configure each I/O bank in an Axcelerator device when in low-power mode:

## Low-Power Mode

- **Enable Input Buffers**- Select to enable input buffers. If this option is selected, all used input buffers within this bank will remain enabled whether or not the LP pin is asserted.
- **Enable Output Buffers** - Select to enable output buffers. If this option is selected, all used output buffers within this bank will remain enabled whether or not the LP pin is asserted.

## Input Delay

Drag the slider bar to your desired delay. The delay is bank-specific. The delay code and typical value appear. Click **View All Delays** to see all the delay values (Best, Worst, Typical, Rise-Rise, Fall-Fall) for the input delay selected. You must select a technology to see the input delays.

Note: The Low-Power mode option is not supported in the IGLOO (all devices), ProASIC3 nano, ProASIC3L, Fusion and RTAX-S families.

For more information, refer to the datasheet for your device. Datasheets are available from the Actel web site.

# Entering Constraints

You can enter design constraints in the following ways:

- **Importing constraint files**: You can import GCF, PDC, SDC, DCF, or PIN constraint files. The type of file you use depends on which type of device you are designing.

- GCF (ProASIC and ProASIC^PLUS families)
- PDC (IGLOO, ProASIC3, SmartFusion, Fusion, and Axcelerator families)
- SDC (IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, RTAX-S, eX, and SX-A families)
- DCF (earlier Antifuse families such as eX, SX-A, and SX)
- PIN (only valid for earlier Antifuse families such as eX, SX-A, and SX)

- **Using constraint editor tools**: Designer's constraint editors are graphical user interface (GUI) tools for creating and modifying physical, logical, and timing constraints. Using these tools enables you to enter constraints without having to understand GCF, PDC, or other file syntax. Which constraint editor you use depends on which type of device you are designing.

For **IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^PLUS, ProASIC, and Axcelerator**, use the tools within the MultiView Navigator:

- ChipPlanner - Sets location and region assignments
- PinEditor in MVN - Sets the pin location constraints
- I/O Attribute Editor - Sets I/O attributes
- SmartTime Constraints EditorSmartTime Constraints Editor - Enables you to view and edit timing constraints

For all other families, you will use the following tools:

- ChipEditor - Sets location and region assignments
- PinEditor (non MVN)- Sets I/O attributes and pin location constraints
- Timer - Sets timing constraints

## See Also

Constraint Support by Family

Constraint Entry

Constraint File Format by Family

Designer Naming Conventions

# Importing Constraint Files

You can import a constraint file as either a source file or an auxiliary file. For details on how to import constraints files, refer to Importing Files in the Libero IDE User's Guide.

## Source File

Import constraints file as source files if they were created with external tools that will be tracked (audited). This helps to coordinate the design changes better. For details on how to import source files, refer to Importing Source files in the *Designer User's Guide*.

The following table shows different constraints format files that can be imported as source files for specific families.

Table 4 · File Types You Can Imported as Source Files

| Source Files | File Type Extension | Family |
|---|---|---|
| ProASIC Constraint File | *.gcf | ProASIC PLUS and ProASIC.<br>GCF **timing** constraints are not supported for ProASIC PLUS; use SDC timing constraints instead. |
| Physical Design Constraint File | *.pdc | IGLOO, ProASIC3, SmartFusion, Fusion. Axcelerator |
| Synopsys Constraint File | *.sdc | IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC PLUS, ProASIC*, Axcelerator, eX, SX-A<br>(*) Supported for analysis only. |
| PIN file | *.pin | eX, SX-A, SX, DX, MX, ACT3, ACT2/1200XL, ACT1 |

## Auxiliary File

When you import a constraint file as an auxiliary file, it is not audited and is treated more as one-time data-entry or data-change events, similar to entering data using one of the interactive editors. For details on how to import auxiliary files, refer to Importing Auxiliary files section in the *Designer User's Guide*.

The following table shows different constraints format files that can be imported as auxiliary files for specific families.

Table 5 · File Types You Can Imported as Auxiliary Files

| Auxiliary Files | File Type Extension | Family |
|---|---|---|

| Auxiliary Files | File Type Extension | Family |
|---|---|---|
| PIN | *.pin | eX, SX-A, SX, DX, MX, ACT3, ACT2/1200XL, ACT1 |
| SDC | *.sdc | IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC PLUS, Axcelerator, eX, SX-A |
| Physical Design Constraint** | *.pdc | IGLOO, Fusion, ProASIC3, and Axcelerator |
| Design Constraint File | *.dcf | eX, SX-A, SX, DX, MX, ACT3, ACT2/1200XL, ACT1 |
| Switching Activity Intermediate File/Format | *.saif | IGLOO, Fusion, ProASIC3, ProASIC PLUS, Axcelerator, ProASIC |
| Value Change Dump file | *.vcd | IGLOO, Fusion, ProASIC3, ProASIC PLUS, Axcelerator, ProASIC |

(*) When you import SDC as an auxiliary file, you can specify only one file in the **File > Import Auxiliary Files** dialog box.

(**) Not all PDC commands are supported when a PDC file is imported as an auxiliary file; some must be imported as source files. When importing a PDC file as an auxiliary file, the new or modified PDC constraints are merged with the existing constraints. The software resolves any conflicts between new and existing physical constraints and displays the appropriate message. Most PDC commands can be imported as auxiliary files. PDC commands that are not supported when the PDC file is imported as an auxiliary file are noted in their respective help topics.

You can either overwrite or retain your existing timing and physical constraints. For details on how to preserve the existing timing constraints, refer to . For details on how to preserve the existing physical constraints, refer to .

## See Also

Importing source files

Importing auxiliary files

Keep Existing Timing Constraints

Keep Existing Physical Constraints

# Using GUI Tools

# MultiView Navigator (MVN)

The MultiView Navigator is an interface in the Designer software that enables you to view and edit physical constraints for the IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC Plus, ProASIC, Axcelerator, SX-A, and eX families. It includes the following tools:

- NetlistViewer - generates a schematic view of your design.
- PinEditor - displays a view of the I/O macros assigned to the pins in your design.
- I/O Attribute Editor - displays a table of the I/O attributes in your design.
- ChipPlanner - displays a view of the I/O and logic macros in your design.

Note: MultiView Navigator works with SmartTime and SmartPower.

### See Also

Overview

About NetlistViewer in MultiView Navigator

About PinEditor in MultiView Navigator

About I/O Attribute Editor

About ChipPlanner

# Non-MVN Tools

The following GUI tools in the Designer software enable you to view and edit physical constraints for the SX, MX, 3200DX, ACT3, ACT2, and ACT1 families:

- ChipEditor - a graphical application for viewing and assigning I/O and logic macros. This tool is particularly useful when you need maximum control over your design placement.
- PinEditor - a graphical application for assigning I/O ports to package pins.
- NetlistViewer - a graphical application for displaying the contents of a design as a schematic. Use this tool to view nets, ports, and instances and to trace signals.

Note: Note: These tools work with Timer.

### See Also

About ChipEditor

About PinEditor

About NetlistViewer (non-MVN)

# About SmartTime Constraints Editor

SmartTime Constraints Editor is an interface in the Designer software that enables you to view and edit timing constraints. Use this editor to view, edit, and create timing constraints used by the SmartTime timing analysis and timing-driven optimization tools. The editor includes powerful visual dialogs that guide you toward capturing your timing requirements and timing exceptions quickly and correctly. The editor is also closely connected to the analysis view of SmartTime (SmartTime Timing Analyzer) that enables you to quickly analyze the impact of constraint changes.

# Exporting Constraint Files

The following table shows a complete list of constraint files that you can export along with the supported family.

| File | File Extension | Families |
|---|---|---|
| ProASIC Constraints file | .gcf | ProASIC |
| ProASIC PLUS Constraints file | .gcf | ProASIC PLUS (Timing constraints in GCF are not supported) |
| Criticality | *.crt | 3200DX, 1200XL, MX, ACT3, ACT2, ACT1 |
| PIN | *.pin | eX, SX-A, SX, 3200DX, 1200XL, MX, ACT3, ACT2, ACT1 |
| SDC | *.sdc | IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC PLUS, Axcelerator, eX, SX-A |
| Physical Design Constraint | *.pdc | IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator |
| Design Constraint file | *.dcf | eX, SX-A, SX, 3200DX, 1200XL, MX, ACT3, ACT2, ACT1 |

For details on how to export a constraint file, refer to .

## See Also

Exporting Files

# Constraints by Name: Timing

# Create Clock

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | DCF | Timer/SmartTime |
|---|---|---|---|
| IGLOO | X | | X |
| SmartFusion | X | | X |
| Fusion | X | | X |
| ProASIC3 | X | | X |
| ProASIC PLUS | X | | X |
| ProASIC | * | | X |
| Axcelerator | X | | X |
| eX | X | X | X |
| SX-A | X | X | X |
| SX | | X | X |
| MX | | X | X |
| 3200DX | | X | X |
| ACT3 | | X | X |
| ACT2/1200XL | | X | X |
| ACT1 | | X | X |

(*) Supported for analysis only.

## Purpose

Use this constraint to create a clock constraint at a specific source and define its waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by the defined clock source. The clock information is also used to compute the slacks in the specified clock domain, display setup and hold violations, and drive optimization tools such as place-and-route.

## Tools /How to Enter

You can use one or more of the following methods to enter clock constraints:

- SDC - create_clock
- DCF – global_clocks
- Timer - Clocks tab
- SmartTime - Specifying Clock Constraint

### See Also

Constraint Entry

`create_clock` (SDC)
Clock Definition

Clocks Tab (Timer)

Specifying Clock Constraint

# Create Generated Clock

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | Timer/SmartTime |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | X | X |
| ProASIC | X* | X* |
| Axcelerator | X | X |
| eX | X | X |
| SX-A | X | X |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

(*) Supported for analysis only

## Purpose

Use this constraint to create an internally generated clock constraint, such as clock dividers and PLL. The generated clock is defined in terms of multiplication and/or division factors with respect to a reference clock pin. When the reference clock pin changes, the generated clock is updated automatically.

## Tools /How to Enter

You can use one or more of the following methods to enter clock constraints:

- SDC – create_generated_clock
- SmartTime - Specifying Generated Clock Constraint

### See Also

Constraint Entry

`create_generated_clock` (SDC)

Specifying Generated Clock Constraint

# Remove Clock Uncertainty

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | Timer/SmartTime |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | X | X |
| ProASIC | X | X |
| Axcelerator | X* | X |
| eX | X | |
| SX-A | X | |

| Families | SDC | Timer/SmartTime |
|---|---|---|
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

(*) Supported for analysis only.

## Purpose

Use this constraint to remove the timing uncertainty between two clock waveforms within SmartTime.

You can remove clock uncertainty constraints in an SDC file, which you can either create yourself or generate with Synthesis tools, at the same time you import the netlist. Alternatively, you can remove clock uncertainty using the GUI tools in the Designer software.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to remove clock uncertainty:

- SDC – remove_clock_uncertainty
- SmartTime - Specifying Clock-to-Clock Uncertainty Constraint

### See Also

Constraint Entry

_Ref-737667696set_clock_uncertainty(SDC)

*SmartTime User's Guide*: Specifying Clock-to-Clock Uncertainty Constraint

# Set Clock Latency

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | Timer/SmartTime |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | X | X |
| ProASIC | X | X |
| Axcelerator | X* | X |
| eX | X | |
| SX-A | X | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

(*) Supported for analysis only.

## Purpose

Use this constraint to define the delay between an external clock source and the definition pin of a clock within SmartTime.

You can set clock latency constraints in an SDC file, which you can either create yourself or generate with Synthesis tools, at the same time you import the netlist. Alternatively, you can set clock latency using the GUI tools in the Designer software when you implement your design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to set clock latency:

- SDC – set_clock_latency
- SmartTime - Specifying Clock Source Latency

### See Also

Constraint Entry

set_clock_latency (SDC)

*SmartTime User's Guide*: Specifying Clock Source Latency

# Set Clock Uncertainty Constraint

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | Timer/SmartTime |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | X | X |
| ProASIC | X | X |
| Axcelerator | X* | X |

| Families | SDC | Timer/SmartTime |
|---|---|---|
| eX | X | |
| SX-A | X | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

(*) Supported for analysis only.

## Purpose

Use this constraint to define the timing uncertainty between two clock waveforms or maximum skew within SmartTime.

You can set clock uncertainty constraints in an SDC file, which you can either create yourself or generate with Synthesis tools, at the same time you import the netlist. Alternatively, you can set clock uncertainty using the GUI tools in the Designer software when you implement your design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to set clock uncertainty:

- SDC – set_clock_uncertainty
- SmartTime - Specifying Clock-to-Clock Uncertainty Constraint

### See Also

Constraint Entry

set_clock_uncertainty(SDC)

# Set Disable Timing Constraint

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | Timer/SmartTime |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | | |
| ProASIC | | |
| Axcelerator | X (including RTAX-S) | X |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

(*) Supported for analysis only.

## Purpose

Use this constraint disable the timing arc in the specified ports on a path.

You can disable the timing arc in an SDC file, which you can either create yourself or generate with Synthesis tools, at the same time you import the netlist. Alternatively, you can disable the timing arc using the GUI tools in the Designer software when you implement your design. <<is this true?>>

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to set maximum delay exception constraints:

- SDC – set_disable_timing
- SmartTime – Specifying Disable Timing Constraint

### See Also

Constraint Entry

set_disable_timing(SDC)

# Set False Path

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | DCF | Timer/SmartTime |
|---|---|---|---|
| IGLOO | X | | X |
| SmartFusion | X | | X |
| Fusion | X | | X |
| ProASIC3 | X | | X |
| ProASIC PLUS | X | | X |
| ProASIC | X* | | X |
| Axcelerator | X | | X |
| eX | X*** | X | X |
| SX-A | X*** | X | X |
| SX | | X | X |
| MX | | X | X |
| 3200DX | | X | X |
| ACT3 | | X | X |
| ACT2/1200XL | | X | X |
| ACT1 | | X | X |

(*) Supported for analysis only.

(***) Only the -through option is supported for layout.

## Purpose

Use this constraint to identify paths in the design that should be disregarded during timing analysis and timing optimization.

By definition, false paths are paths that cannot be sensitized under any input vector pair. Therefore, including false paths in timing calculation may lead to unrealistic results. For accurate static timing analysis, it is important to identify the false paths.

You can set false paths constraints in an SDC file, which you can either create yourself or generate with Synthesis tools, at the same time you import the netlist. Alternatively, you can set false paths using the GUI tools in the Designer software when you implement your design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to set false paths:

- SDC – set_false_path
- DCF -**global_stops**
- Timer - Breaks Tab
- SmartTime - Specifying False Path Constraint

### See Also

Constraint Entry

`set_false_path` (SDC)
`global_stops`(DCF)
Breaks Tab

Specifying False Path Constraint

# Set Input Delay

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | Timer/SmartTime |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | X | X |
| ProASIC | X* | X |
| Axcelerator | X | X |
| eX | X* | X |
| SX-A | X* | X |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

(*) Supported for analysis only.

(**) Supported for layout only.

## Purpose

Use this constraint to define the arrival time relative to a clock.

## Tools /How to Enter

You can use one or more of the following methods to set input delay constraint:

- SDC – set_input_delay
- SmartTime - Specifying Input Delay Constraint

### See Also

Constraint Entry

`set_input_delay` (SDC)

*SmartTime User's Guide*: Specifying Input Delay Constraint

# Set Load on Output Port

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | DCF | I/O Attribute Editor | PinEditor |
|---|---|---|---|---|
| IGLOO | X | | X | |
| SmartFusion | X | | X | |
| Fusion | X | | X | |
| ProASIC3 | X | | X | |
| ProASIC PLUS | | | X | |
| ProASIC | | | X | |
| Axcelerator | X | | X | |
| eX | X | X | | X |
| SX–A | X | X | | X |
| SX | | X | | X |
| MX | | X | | X |
| 3200DX | | X | | X |
| ACT3 | | X | | X |
| ACT2/1200XL | | X | | X |
| ACT1 | | X | | X |

## Purpose

Use this constraint to set the capacitance to a specified value on a specified port.

Delay on a given path depends on the load at the output pin of the device. For an accurate static timing analysis of a given design, it is important to set the load on the port which can be taken into account for delay calculations.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to set the load on a port:

- SDC – set_load
- DCF – pin_loads
- I/O Attribute Editor – Editing I/O Attributes
- PinEditor (non–MVN) – Specifying Capacitance
- SmartTime Constraints Editor GUI – Changing Output Port Capacitance

Note: Note: You can also set the output load using the pin_assign command in a Tcl script.

### See Also

Constraint Entry

`set_load` (SDC)
`pin_loads` (DCF)
`pin_assign`
*MultiView Navigator User's Guide*: Editing I/O Attributes

*PinEditor (Non–MVN) User's Guide*: Specifying Capacitance

# Set Maximum Delay

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | DCF | Timer/SmartTime |
|---|---|---|---|
| IGLOO | X | | X |
| SmartFusion | X | | X |
| Fusion | X | | X |
| ProASIC3 | X | | X |
| ProASIC PLUS | X | | X |
| ProASIC | X* | | X |
| Axcelerator | X | | X |
| eX | X** | X | X |
| SX-A | X** | X | X |
| SX | | X | X |
| MX | | X | X |
| 3200DX | | X | X |
| ACT3 | | X | X |
| ACT2/1200XL | | X | X |
| ACT1 | | X | X |

(*) Supported for analysis only.

(**) the -through option is not supported for layout.

## Purpose

Use this constraint to set the maximum delay exception between the specified ports on a path.

You can set maximum delay exception in an SDC file, which you can either create yourself or generate with Synthesis tools, at the same time you import the netlist. Alternatively, you can set maximum delay exceptions using the GUI tools in the Designer software when you implement your design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to set maximum delay exception constraints:

- SDC – set_max_delay
- DCF – pin_loads, max_delays
- Timer – Paths Tab
- SmartTime – Specifying Maximum Delay Constraint

### See Also

Constraint Entry

`set_max_delay` (SDC)
`pin_loads` (DCF)
`max_delays` (DCF)
*Timer User's Guide*: Paths Tab

# Set Minimum Delay

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | DCF | Timer/SmartTime |
|---|---|---|---|
| IGLOO | X | | X |
| SmartFusion | X | | X |
| Fusion | X | | X |
| ProASIC3 | X | | X |
| ProASIC PLUS | X | | X |
| ProASIC | X* | | X |
| Axcelerator | X | | X |
| eX | X** | X | X |
| SX-A | X** | X | X |
| SX | | X | X |
| MX | | X | X |
| 3200DX | | X | X |
| ACT3 | | X | X |
| ACT2/1200XL | | X | X |
| ACT1 | | X | X |

(*) Supported for analysis only.

(**) the -through option is not supported for layout.

## Purpose

Use this constraint to set the minimum delay exception between the specified ports on a path.

You can set minimum delay exception in an SDC file, which you can either create yourself or generate with Synthesis tools, at the same time you import the netlist. Alternatively, you can set minimum delay exceptions using the GUI tools in the Designer software when you implement your design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to set maximum delay exception constraints:

- SDC – set_min_delay
- SmartTime – Specifying minimum delay constraint

### See Also

Constraint Entry

set_min_delay (SDC)

# Set Multicycle Path

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | Timer/SmartTime |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | X | X |
| ProASIC | X* | X |
| Axcelerator | X | X |
| eX | X* | X |
| SX-A | X* | X |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

(*) Supported for analysis only.

## Purpose

Use this constraint to identify paths in the design that take multiple clock cycles.

You can set multicycle path constraints in an SDC file, which you can either create yourself or generate with Synthesis tools, at the same time you import the netlist. Alternatively, you can set multicycle paths using the GUI tools in the Designer software when you implement your design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to set multicycle paths constraints:

- SDC – set_multicycle_path
- SmartTime – Specifying Input Delay Constraint

### See Also

Constraint Entry

`set_multicycle_paths` `(SDC)`
*SmartTime User's Guide*: Specifying Input Delay Constraint

# Set Output Delay

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | SDC | Timer/SmartTime |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | X | X |
| ProASIC | X* | X |
| Axcelerator | X | X |
| eX | X* | X |
| SX-A | X* | X |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

(*) Supported for analysis only.

## Purpose

Use this constraint to set the output delay of an output relative to a clock.

## Tools /How to Enter

You can use one or more of the following methods to set output delay constraints:

- SDC – set_output_delay
- SmartTime – Specifying Output Delay Constraint

### See Also

Constraint Entry

`set_output_delay` (SDC)

*SmartTime User's Guide*: Specifying Output Delay Constraint

# Constraints by Name: Physical

# Assign I/O to Pin

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF | PIN | PinEditor |
|---|---|---|---|---|
| IGLOO | X | | | X |
| SmartFusion | X | | | X |
| Fusion | X | | | X |
| ProASIC3 | X | | | X |
| ProASIC PLUS | | X | | X |
| ProASIC | | X | | X |
| Axcelerator | X | | | X |
| eX | | | X | X |
| SX-A | | | X | X |
| SX | | | X | X |
| MX | | | X | X |
| 3200DX | | | X | X |
| ACT3 | | | X | X |
| ACT2/1200XL | | | X | X |
| ACT1 | | | X | X |

## Purpose

Use this constraint to set the location of a pin.

For IGLOO, ProASIC3 and Axcelerator, you can use the set_io command in a PDC file to assign I/Os to pins as well as set the attributes of an I/O. For ProASIC PLUS and ProASIC, you can use the set_io command in a GCF file to assign package pins to I/O ports or to locate I/O ports at a specified side or location of a device. For earlier families, you can use a PIN file to set the location of a pin.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to assign an I/O to a pin:

- PDC - set_io
- GCF - set_io
- PIN - PIN <pin_name>; PIN:<package_pin_number>
- PinEditor (MVN) - Assigning pins
- PinEditor (non-MVN) - Assigning pins

Note: Note: You can also set the location of a pin using the pin_assign command in a Tcl script.

### See Also

Constraint Entry

`set_io (PDC)`
`set_io (GCF)`
About PIN Files

`pin_assign`
*MultiView Navigator User's Guide*: Assigning Pins

*PinEditor (non-MVN) User's Guide*: Assigning Pins

# Assign I/O Macro to Location

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF | ChipPlanner | ChipEditor |
|---|---|---|---|---|
| IGLOO | X | | X | |
| SmartFusion | X | | X | |
| Fusion | X | | X | |
| ProASIC3 | X | | X | |
| ProASIC PLUS | | X | X | |
| ProASIC | | X | X | |
| Axcelerator | X | | X | |
| eX | | | | X |
| SX-A | | | | X |
| SX | | | | X |
| MX | | | | X |
| 3200DX | | | | X |
| ACT3 | | | | X |
| ACT2/1200XL | | | | X |
| ACT1 | | | | X |

## Purpose

Use this constraint to assign one or more I/O macros to a specific location. You can define the location using array co-ordinates.

By confining macros to one area, you can keep the nets connected to that area, resulting in better timing and better floorplanning. Sometimes placing some macros at specific locations can also result in meeting timing closures.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to assign a macro to a location:

- PDC - set_location
- GCF - set_location
- ChipPlanner -
- ChipEditor - Assigning Logic

### See Also

Constraint Entry

`set_location (PDC)`
`set_location (GCF)`
*MultiView Navigator User's Guide*: Assigning Logic to Locations

*ChipEditor User's Guide*: Assigning Logic

# Assign Macro to Region

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF | ChipPlanner |
|---|---|---|---|
| IGLOO | X | | X |
| SmartFusion | X | | X |
| Fusion | X | | X |
| ProASIC3 | X | | X |
| ProASIC PLUS | | X | X |
| ProASIC | | X | X |
| Axcelerator | X | | X |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to assign one or more macros to a specific region.

By confining macros to one area, you can keep the nets connected to that area, resulting in better timing and better floorplanning.

For IGLOO, ProASIC3, Fusion, SmartFusion and Axcelerator devices, you can use the define_region PDC command to create a region, and then use the assign_region PDC command to constrain a set of existing macros to that region.

For ProASIC PLUS and ProASIC, you can use the set_location GCF command to both create a region and constrain an existing set of macros to it at the same time. To define a region with the set_location command in a GCF file, you must specify the array coordinates for a rectangular area, for example, x1, y1, x2, y2.

You can also use the MultiView Navigator tool to create regions for any of the supported families.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to assign a macro to a region:

- PDC - assign_region
- GCF - set_location
- ChipPlanner - Assigning a macro to a region

### See Also

Constraint Entry

`assign_region` (PDC)
`set_location` (GCF)
*MultiView Navigator User's Guide*: Assigning a Macro to a Region

# Assign Net to Global Clock

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF |
|---|---|---|
| IGLOO | X | |
| SmartFusion | X | |
| Fusion | X | |
| ProASIC3 | X | |
| ProASIC PLUS | | X |
| ProASIC | | X |
| Axcelerator | | |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to assign high fan-out nets to global clock networks by promoting the net using an internal global macro.

If there are enough global clock routing resources available in a device, you can promote regular nets that have high fan-out to the dedicated fast global clock routing resources which can lead to better performance for your design. This is achieved by automatically inserting an internal global macro on a net which guides the place-and-route tool to promote that particular net to a global clock resource. This internal global macro is CLKINT for IGLOO, ProASIC3, SmartFusion and Fusion families, GLINT for ProASIC PLUS and ProASIC families, and either HCLKINT or CLKINT for Axcelerator families.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to assign a net to a global clock:

- PDC - `assign_global_clock`
- GCF - `set_global`

### See Also

Constraint Entry

`assign_global_clock` (PDC)
`set_global` (GCF)

# Assign Net to Local Clock

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF | ChipPlanner |
|---|---|---|---|
| IGLOO | X | | |
| SmartFusion | X | | |
| Fusion | X | | |
| ProASIC3 | X | | |
| ProASIC PLUS | | X | X |
| ProASIC | | X | X |
| Axcelerator | X | | |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to assign regular nets to local clock routing or to LocalClock regions. This results in the creation of a LocalClock region that spans the area of the local clock net.

If there are enough local clock resources but not enough global clock routing resources available in a device, you can assign regular nets that have high fan-out to the dedicated local clock routing resources which can lead to better performance for your design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to assign a net to a local clock:

- PDC -`assign_local_clock`
- GCF - `use_global`
- ChipPlanner - Creating LocalClock Regions

### See Also

Constraint Entry

`assign_local_clock` (PDC)
`use_global` (GCF)
*MultiView Navigator User's Guide*: Creating LocalClock Regions

# Assign Net to Quadrant Clock

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | ChipPlanner |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | | |
| ProASIC | | |
| Axcelerator | | |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to assign regular nets to quadrant clock routing. This results in the creation of a QuadrantClock region that spans the area of the quadrant clock net.

If there are enough quadrant clock resources but not enough global clock routing resources available in a device, you can promote regular nets that have high fan-out to the dedicated quadrant clock routing resources which can lead to better performance for your design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to assign a net to a local clock:

- PDC - `assign_quadrant_clock`
- ChipPlanner - Creating QuadrantClock Regions

### See Also

Constraint Entry

`assign_quadrant_clock` `(PDC)`

*MultiView Navigator User's Guide*: Creating QuadrantClock Regions

# Assign Net to Region

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF | ChipPlanner |
|---|---|---|---|
| IGLOO | X | | X |
| SmartFusion | X | | X |
| Fusion | X | | X |
| ProASIC3 | X | | X |
| ProASIC PLUS | | X | X |
| ProASIC | | X | X |
| Axcelerator | X | | X |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to place all the loads of a net into a given region. This constraint is useful for high fan-out or critical path nets or bus control logic.

Constraining nets to a region helps to control the connection delays from the net's driver to the logic instances it fans out to. You can adjust the size of the region to pack logic more closely together, hence, improving its net delays.

Suppose you have a global net with loads that span across the whole chip. When you constrain this net to a specific region, you force the loads of this global net into the given region. Forcing loads into a region frees up some areas that were previously used. You can then use these free areas for high-speed local clocks/spines.

Macros connected to a specific net can be assigned to a region in the device. The region can be defined using the define_region PDC command. With the set_net_region GCF command, you can use array coordinates to define the region into which you want to place all the connected instances, driver, and all the driven instances for the net(s).

When assigning a net to a region, all of the logic driven by that net will be assigned to that region.

**Using Regions for Critical Path and High Fan-out Nets**

You should assign high fan-out 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.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to assign a net to a region:

- PDC - `assign_net_macros`
- GCF -`set_net_region`
- ChipPlanner - Assigning a Net to a Region

### See Also

Constraint Entry

`assign_net_macros` (PDC)
`set_net_region` (GCF)
*MultiView Navigator User's Guide*:Assigning a Net to a Region

# Configure I/O Bank

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | ChipPlanner |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | | |
| ProASIC | | |
| Axcelerator | X | X |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to set the I/O supply voltage (VCCI) for I/O banks.

I/Os are organized into banks. The configuration of these banks determines the I/O standards supported. Since each I/O bank has its own user-assigned input reference voltage (VREF) and an input/output supply voltage, only I/Os with compatible standards can be assigned to the same bank.

For IGLOO, ProASIC3E, SmartFusion, Fusion and Axcelerator families, you can use the `set_iobank` PDC command to set the input/output supply voltage and the input reference voltage for an I/O bank. However, for ProASIC3 devices, you can use this command to set only the input/output supply voltage for an I/O bank.

For Axcelerator families, you can also use the `set_iobank` command to set the input delay value and enable or disable the low-power mode for input and output buffers.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to configure I/O banks:

- PDC - set_iobank
- ChipPlanner - Manually Assigning Technologies to I/O Banks

### See Also

Constraint Entry

`set_iobank`

*MultiView Navigator User's Guide*: Manually Assigning Technologies to I/O Banks

# Create Region

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF | ChipPlanner |
|---|---|---|---|
| IGLOO | X | | X |
| SmartFusion | X | | X |
| Fusion | X | | X |
| ProASIC3 | X | | X |
| ProASIC PLUS | | X | X |
| ProASIC | | X | X |
| Axcelerator | X | | X |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to create either a rectangular or rectilinear region on a device.

You can create a region within a device for setting specific physical constraints or for achieving better floorplanning. You can define regions with the array coordinates for that particular device.

For IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator families, you can use the `define_region` PDC command to create a rectangular or rectilinear region, and then use the `assign_region` PDC command to constrain a set of macros to that region.

For ProASIC PLUS and ProASIC, you can use the `set_location` GCF command to both create a region and constrain a set of macros to it at the same time. To define a region with the set_location command in a GCF file, you must specify the array coordinates for a rectangular area, for example, x1, y1, x2, y2.

You can also use the MultiView Navigator tool to create regions for any of the supported families.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to create a region constraint:

- PDC -`define_region`
- GCF -`set_location`
- ChipPlanner - Creating_regions

### See Also

Constraint Entry

`define_region` (PDC)
`set_location` (GCF)
*MultiView Navigator User's Guide*: Creating Regions

# Delete Regions

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | ChipPlanner |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | | X |
| ProASIC | | X |
| Axcelerator | X | X |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to remove the region(s) that you specify. You can use wildcards in the undefine_region PDC command to delete all user regions.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to delete all regions:

- PDC - undefine_region or reset_floorplan
- ChipPlanner - Editing Regions

### See Also

Constraint Entry

`undefine_region`

*MultiView Navigator User's Guide*: Editing Regions

# Move Block

## Families Supported

The following table shows which families support this constraint and which tools you can use to enter or modify it:

| Families | PDC | GCF | Compile Options |
|---|---|---|---|
| IGLOO | X | | |
| SmartFusion | X | | |
| Fusion | X | | |
| ProASIC3 | X | | |
| ProASIC PLUS | | | |
| ProASIC | | | |
| Axcelerator | X | | |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to move a Designer block from its original, locked placement by preserving the relative placement between the instances. You can move the block to the left, right, up, or down.

## Tools /How to Enter

You can use the following command to move a Designer block:

- PDC - move_block

### See Also

Set Block Options

Constraint Entry

# Move Region

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | ChipPlanner |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | | X |
| ProASIC | | X |
| Axcelerator | X | X |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to move the location of a previously defined region.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to move a region:

- PDC - move_region
- ChipPlanner - Editing Regions

### See Also

Constraint Entry

`move_region` (PDC)

*MultiView Navigator User's Guide*: Editing Regions

# Reserve Pins

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | I/O Attribute Editor | PinEditor |
|---|---|---|---|
| IGLOO | X | X | X |
| SmartFusion | X | X | X |
| Fusion | X | X | X |
| ProASIC3 | X | X | X |
| ProASIC PLUS | | | |
| ProASIC | | | |
| Axcelerator | X | X | X |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to reserve pins for use in a later design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to reserve one or more pins in your design:

- PDC - reserve
- PinEditor (MVN) - Reserving pins
- I/O Attribute Editor (MVN)- Assigning pins in Package Pins View

### See Also

unreserve

Constraint Entry

*MultiView Navigator User's Guide*: Assigning Pins

# Reset Attributes on an I/O to Default Settings

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | I/O Attribute Editor | ChipPlanner |
|---|---|---|---|
| IGLOO | X | X | X |
| SmartFusion | X | X | X |
| Fusion | X | X | X |
| ProASIC3 | X | X | X |
| ProASIC PLUS | | X | X |
| ProASIC | | X | X |
| Axcelerator | X | X | X |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to either reset an I/O to its default settings or to unassign an I/O.

Attributes for an I/O, such as I/O standard, I/O threshold, Output drive, and so on, can be restored to their default values. There are no I/O banks in ProASIC PLUS or ProASIC devices; however, you can unassign I/Os in these devices using the MultiView Navigator's ChipPlanner tool.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to restore I/O attributes:

- PDC - reset_io
- I/O Attribute Editor - Editing I/O Attributes
- ChipPlanner - Unassigning Pins

### See Also

Constraint Entry

`reset_io`

*MultiView Navigator User's Guide*: Editing I/O Attributes, Unassigning Pins

# Reset an I/O Bank to Default Settings

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | ChipPlanner | I/O Attribute Editor |
|---|---|---|---|
| IGLOO | X | X | X |
| SmartFusion | X | X | X |
| Fusion | X | X | X |
| ProASIC3 | X | X | X |
| ProASIC PLUS | | | |
| ProASIC | | | |
| Axcelerator | X | X | X |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to reset an I/O bank's technology to the default setting, which was specified in the Device Selection Wizard.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to reset an I/O bank to its default technology:

- PDC - reset_iobank
- I/O Attribute Editor - Editing I/O Attributes
- ChipPlanner - Assigning technologies to I/O banks

### See Also

Constraint Entry

`reset_iobank`

*MultiView Navigator User's Guide*: Assigning Technologies to I/O Banks, Editing I/O Attributes

# Reset Net's Criticality to Default Level

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC |
|---|---|
| IGLOO | |
| SmartFusion | |
| Fusion | |
| ProASIC3 | |
| ProASIC PLUS | |
| ProASIC | |
| Axcelerator | X |
| eX | |
| SX-A | |
| SX | |
| MX | |
| 3200DX | |
| ACT3 | |
| ACT2/1200XL | |
| ACT1 | |

## Purpose

Use this constraint to reset a net's criticality to its default value, which is 5.

Net criticality is a guide for the place-and-route tool to keep instances connected to a net as close as possible, at the cost of other (less critical) nets. Net criticality can vary from 1 to 10 with 1 being the least critical and 10 being the most.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to reset net criticality:

- PDC - reset_net_critical

### See Also

Constraint Entry

`reset_net_critical`
`set_net_critical`

# Set Block Options

## Families Supported

The following table shows which families support this constraint and which tools you can use to enter or modify it:

| Families | PDC | GCF | Compile Options |
|---|---|---|---|
| IGLOO | X | | |
| SmartFusion | X | | |
| Fusion | X | | |
| ProASIC3 | X | | |
| ProASIC PLUS | | | |
| ProASIC | | | |
| Axcelerator | X | | |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to override the compile option for placement or routing conflicts for an instance of a Designer block.

## Tools /How to Enter

You can use the following command to preserve instances:

- PDC - set_block_options

### See Also

Move Block

Constraint Entry

# Set Net's Criticality

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF |
|---|---|---|
| IGLOO | | |
| SmartFusion | | |
| Fusion | | |
| ProASIC3 | | |
| ProASIC PLUS | | X |
| ProASIC | | X |
| Axcelerator | X | |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to set the level at which the place-and-route tool must keep instances connected to a net.

Net criticality is a guide for the place-and-route tool to keep instances connected to a net as close as possible at the cost of other (less critical) nets. Net criticality can vary from 1 to 10 with 1 being the least critical and 10 being the most. You can set a net's criticality to any number between 1 and 10 to help place-and-route tool prioritize its timing driven placement.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to set net criticality:

- PDC - set_net_critical
- GCF - set_critical

### See Also

Constraint Entry

set_net_critical
set_critical

# Set Port Block

## Families Supported

The following table shows which families support this constraint and which tools you can use to enter or modify it:

| Families | PDC | GCF | Compile Options |
|---|---|---|---|
| IGLOO | X | | |
| SmartFusion | X | | |
| Fusion | X | | |
| ProASIC3 | X | | |
| ProASIC PLUS | | | |
| ProASIC | | | |
| Axcelerator | X | | |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to set properties on a port in the block flow.

## Tools /How to Enter

You can use the following command to preserve instances:

- PDC - set_port_block

### See Also

Constraint Entry

# Unassign I/O Macro from Location

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | ChipPlanner | ChipEditor |
|---|---|---|---|
| IGLOO | X | X | |
| SmartFusion | X | X | |
| Fusion | X | X | |
| ProASIC3 | X | X | |
| ProASIC PLUS | | X | |
| ProASIC | | X | |
| Axcelerator | X | X | |
| eX | | | X |
| SX-A | | | X |
| SX | | | X |
| MX | | | X |
| 3200DX | | | X |
| ACT3 | | | X |
| ACT2/1200XL | | | X |
| ACT1 | | | X |

## Purpose

Use this constraint to unassign a macro or a group of macros from a specific location in the device.

Macros assigned to specific locations with the set_location PDC command can be unassigned from that location using -no switch with the set_location PDC command

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to unassign a macro from a location:

- PDC - set_location and set_multitile_location
- ChipPlanner - Assigning logic to locations
- ChipEditor - Assigning Logic

### See Also

`set_location`
`set_multitile_location`
*MultiView Navigator User's Guide*: Assigning Logic to Locations

*ChipEditor User's Guide*: Assigning Logic

# Unassign Macro from Region

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | ChipPlanner |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | | X |
| ProASIC | | X |
| Axcelerator | X | X |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to unassign one or more macros from a specific region in the device.

Macros assigned to a specific region using the assign_region command can be unassigned from that region using the unassign_macro_from_region command

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to unassign a macro from a region:

- PDC - unassign_macro_from_region
- ChipPlanner - Unassigning a Macro from a Region

### See Also

Constraint Entry

`unassign_macro_from_region`

*MultiView Navigator User's Guide*: Unassigning a Macro from a Region

# Unassign Macro(s) Driven by Net from Region

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | ChipPlanner |
|---|---|---|
| IGLOO | X | X |
| SmartFusion | X | X |
| Fusion | X | X |
| ProASIC3 | X | X |
| ProASIC PLUS | | X |
| ProASIC | | X |
| Axcelerator | X | X |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to unassign macros that are connected to a specific net from an assigned region.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to unassign macros on a net from a region:

- PDC - unassign_net_macros
- ChipPlanner - Unassigning a macro from a region

### See Also

Constraint Entry

`unassign_net_macros`

*MultiView Navigator User's Guide*: Unassigning a Macro from a Region

# Unreserve Pins

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | I/O Attribute Editor | PinEditor |
|---|---|---|---|
| IGLOO | **X** | **X** | **X** |
| SmartFusion | X | X | X |
| Fusion | X | **X** | **X** |
| ProASIC3 | **X** | **X** | **X** |
| ProASIC PLUS | | | |
| ProASIC | | | |
| Axcelerator | **X** | **X** | **X** |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to unreserve pins that were previously reserved. Once pins are unreserved, you can use them again in a design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to unreserve one or more pins in your design:

- PDC - unreserve
- PinEditor (MVN) - Reserving pins
- I/O Attribute Editor (MVN)- Assigning pins in Package Pins ViewAssigning pins in Package Pins view

### See Also

`reserve`

Constraint Entry

*`MultiView Navigator User's Guide`*: `Assigning Pins`

# Constraints by Name: Netlist Optimization

# Delete Buffer Tree

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF |
|---|---|---|
| IGLOO | X | |
| SmartFusion | X | |
| Fusion | X | |
| ProASIC3 | X | |
| ProASIC PLUS | | X |
| ProASIC | | X |
| Axcelerator | | |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to remove all buffers and inverters from a given net. In the IGLOO and ProASIC3 architectures, inverters are considered buffers because all tile inputs can be inverted. This rule is also true for all Flash architectures but not for Antifuse architectures.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to delete a buffer tree:

- PDC - delete_buffer_tree
- GCF - optimize

### See Also

Constraint Entry

_Ref2029020432Netlist Optimization Constraints
dont_touch_buffer_tree (PDC)

# Demote Global Net to Regular Net

## Families Supported

The following table shows which families support this constraint and which tools you can use to enter or modify it:

| Families | PDC | GCF | Compile Options |
|---|---|---|---|
| IGLOO | X | | X |
| SmartFusion | X | | X |
| Fusion | X | | X |
| ProASIC3 | X | | X |
| ProASIC PLUS | | X | |
| ProASIC | | X | |
| Axcelerator | | | |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint either to free up a dedicated clock routing resource by demoting a global net to a regular net or to prevent a clock net from automatically being promoted to a global net.

If there are multiple clocks in a design and not enough clock routing resources, you can demote a global net to a regular net for a clock that does not drive logic through the critical path in a design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to demote a clock net to a regular net:

- PDC - unassign_global_clock
- GCF - set_noglobal
- Compile Options - -demote_globals <value>

### See Also

Constraint Entry

Netlist Optimization Constraints

`unassign_global_clock` (PDC)
`set_noglobal` (GCF)
*Designer User's Guide*: Setting Compile Options, -demote_globals <value>

# Promote Regular Net to Global Net

## Families Supported

The following table shows which families support this constraint and which tools you can use to enter or modify it:

| Families | PDC | GCF | Compile Options |
|---|---|---|---|
| IGLOO | X | | X |
| SmartFusion | X | | X |
| Fusion | X | | X |
| ProASIC3 | X | | X |
| ProASIC PLUS | | X | |
| ProASIC | | X | |
| Axcelerator | | | |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to increase the performance of your design.

If there are enough clock routing resources available in a device, you can promote regular nets that have high fan-out to the dedicated fast clock routing resources which can lead to better performance for your design.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to promote a regular net to a global clock net:

- PDC - assign_global_clock
- GCF file - set_global (backwards compatible for ProASIC families only)
- Compile Options - -promote_globals <value>

### See Also

Constraint Entry

Netlist Optimization Constraints

`assign_global_clock` (PDC)
`set_global` (GCF)
*Designer User's Guide*: Setting Compile Options, -promote_globals <value>

# Restore Buffer Tree

## Families Supported

The following table shows which families support this constraint and which file formats and tools you can use to enter or modify it:

| Families | PDC | GCF |
|---|---|---|
| IGLOO | X | |
| SmartFusion | X | |
| Fusion | X | |
| ProASIC3 | X | |
| ProASIC PLUS | | X |
| ProASIC | | X |
| Axcelerator | | |
| eX | | |
| SX-A | | |
| SX | | |
| MX | | |
| 3200DX | | |
| ACT3 | | |
| ACT2/1200XL | | |
| ACT1 | | |

## Purpose

Use this constraint to undo the operation of a previously specified delete_buffer_tree command. This constraint is useful in the import and merge flow when users want to keep the previous database constraint but want to overwrite just one delete_buffer_tree command.

## Tools /How to Enter

You can use one or more of the following commands or GUI tools to restore a buffer tree:

- PDC - dont_touch_buffer_tree
- GCF - dont_optimize

### See Also

Constraint Entry

`Netlist Optimization Constraints`
`delete_buffer_tree` (PDC)

# Set Preserve

## Families Supported

The following table shows which families support this constraint and which tools you can use to enter or modify it:

| Families | PDC | GCF | Compile Options |
|---|---|---|---|
| IGLOO | X | | |
| SmartFusion | X | | |
| Fusion | X | | |
| ProASIC3 | X | | |
| ProASIC PLUS | | | |
| ProASIC | | | |
| Axcelerator | X | | |
| eX | | | |
| SX-A | | | |
| SX | | | |
| MX | | | |
| 3200DX | | | |
| ACT3 | | | |
| ACT2/1200XL | | | |
| ACT1 | | | |

## Purpose

Use this constraint to preserve instances before compiling them so they will not be combined during compile.

## Tools /How to Enter

You can use the following command to preserve instances:

- PDC - set_preserve

### See Also

Constraint Entry

Netlist Optimization Constraints

set_preserve (PDC)

# Constraints by File Format - SDC Command Reference

# About Synopsys Design Constraints (SDC) Files

Synopsys Design Constraints (SDC) is a Tcl-based format used by Synopsys tools to specify the design intent, including the timing and area constraints for a design. Actel tools use a subset of the SDC format to capture supported timing constraints. You can import or export an SDC file from the Designer software. Any timing constraint that you can enter using Designer tools, can also be specified in an SDC file.

Use the SDC-based flow to share timing constraint information between Actel tools and third-party EDA tools.

| Command | Action |
|---|---|
| create_clock | Creates a clock and defines its characteristics |
| create_generated_clock | Creates an internally generated clock and defines its characteristics |
| remove_clock_uncertainty | Removes a clock-to-clock uncertainty from the current timing scenario. |
| set_clock_latency | Defines the delay between an external clock source and the definition pin of a clock within SmartTime |
| set_clock_uncertainty | Defines the timing uncertainty between two clock waveforms or maximum skew |
| set_false_path | Identifies paths that are to be considered false and excluded from the timing analysis |
| set_input_delay | Defines the arrival time of an input relative to a clock |
| set_load | Sets the load to a specified value on a specified port |
| set_max_delay | Specifies the maximum delay for the timing paths |
| set_min_delay | Specifies the minimum delay for the timing paths |
| set_multicycle_path | Defines a path that takes multiple clock cycles |
| set_output_delay | Defines the output delay of an output relative to a clock |

## See Also

Constraint Entry

SDC Syntax Conventions

Importing Constraint Files

# SDC Syntax Conventions

The following table shows the typographical conventions that are used for the SDC command syntax.

| Syntax Notation | Description |
|---|---|
| command -argument | Commands and arguments appear in `Courier New` typeface. |
| *variable* | Variables appear in blue, italic *Courier New* typeface. You must substitute an appropriate value for the variable. |
| [-argument *value*] | Optional arguments begin and end with a square bracket. |

Note: Note: SDC commands and arguments are case sensitive.

## Example

The following example shows syntax for the create_clock command and a sample command:

```
create_clock -period period_value [-waveform edge_list] source
```

```
create_clock –period 7 –waveform {2 4}{CLK1}
```

## Wildcard Characters

You can use the following wildcard characters in names used in the SDC commands:

| Wildcard | What it does |
|---|---|
| \ | Interprets the next character literally |
| * | Matches any string |

Note: Note: The matching function requires that you add a backslash (\) before each slash in the pin names in case the slash does not denote the hierarchy in your design.

## Special Characters ([ ], { }, and \)

Square brackets ([ ]) are part of the command syntax to access ports, pins and clocks. In cases where these netlist objects names themselves contain square brackets (for example, buses), you must either enclose the names with curly brackets ({}) or precede the open and closed square brackets ([ ]) characters with a backslash (\). If you do not do this, the tool displays an error message.

For example:

```
create_clock -period 3 clk\[0\]
set_max_delay 1.5 -from [get_pins ff1\[5\]:CLK] -to [get_clocks {clk[0]}]
```

Although not necessary, Actel recommends the use of curly brackets around the names, as shown in the following example:

```
set_false_path –from {data1} –to [get_pins {reg1:D}]
```

In any case, the use of the curly bracket is mandatory when you have to provide more than one name.

For example:

```
set_false_path –from {data3 data4} –to [get_pins {reg2:D reg5:D}]
```

## Entering Arguments on Separate Lines

If a command needs to be split on multiple lines, each line except the last must end with a backslash (\) character as shown in the following example:

```
set_multicycle_path 2 –from \
[get_pins {reg1*}] \
-to {reg2:D}
```

### See Also

About SDC Files

# create_clock

Creates a clock and defines its characteristics.

```
create_clock -name name -period period_value [-waveform edge_list]  source
```

## Arguments

`-name name`

Specifies the name of the clock constraint. This parameter is required for virtual clocks when no clock source is provided.

`-period period_value`

Specifies the clock period in nanoseconds. The value you specify is the minimum time over which the clock waveform repeats. The period_value must be greater than zero.

`-waveform edge_list`

Specifies the rise and fall times of the clock waveform in ns over a complete clock period. There must be exactly two transitions in the list, a rising transition followed by a falling transition. You can define a clock starting with a falling edge by providing an edge list where fall time is less than rise time. If you do not specify -waveform option, the tool creates a default waveform, with a rising edge at instant 0.0 ns and a falling edge at instant (period_value/2)ns.

*source*

Specifies the source of the clock constraint. The source can be ports or pins in the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing one. Only one source is accepted. Wildcards are accepted as long as the resolution shows one port or pin.

## Supported Families

IGLOO, ProASIC3, ProASIC[PLUS], Axcelerator, ProASIC (for analysis), eX, SX-A

## Description

Creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

The clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

## Exceptions

- None

## Examples

The following example creates two clocks on ports CK1 and CK2 with a period of 6, a rising edge at 0, and a falling edge at 3:

```
create_clock -name {my_user_clock} -period 6 CK1
create_clock -name {my_other_user_clock} –period 6 –waveform {0 3} {CK2}
```

The following example creates a clock on port CK3 with a period of 7, a rising edge at 2, and a falling edge at 4:

```
create_clock -period 7 -waveform {2 4} [get_ports {CK3}]
```

## Actel Implementation Specifics

- The -waveform in SDC accepts waveforms with multiple edges within a period. In Actel design implementation, only two waveforms are accepted.
- SDC accepts defining a clock on many sources using a single command. In Actel design implementation, only one source is accepted.
- The source argument in SDC create_clock command is optional. This is in conjunction with the -name argument in SDC to support the concept of virtual clocks. In Actel implementation, source is a mandatory argument as -name and virtual clocks concept is not supported.
- The -domain argument in the SDC create_clock command is not supported.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

Clock Definition

Create Clock

Create a New Clock Constraint

# create_generated_clock

Creates an internally generated clock and defines its characteristics.

```
create_generated_clock -name {name -source reference_pin [-divide_by divide_factor] [-
multiply_by multiply_factor] [-invert] source
```

## Arguments

`-name` *name*

Specifies the name of the clock constraint. This parameter is required for virtual clocks when no clock source is provided.

`-source` *reference_pin*

Specifies the reference pin in the design from which the clock waveform is to be derived.

`-divide_by`*divide_factor*

Specifies the frequency division factor. For instance if the *divide_factor* is equal to 2, the generated clock period is twice the reference clock period.

`-multiply_by` *multiply_factor*

Specifies the frequency multiplication factor. For instance if the *multiply_factor* is equal to 2, the generated clock period is half the reference clock period.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`source`

Specifies the source of the clock constraint on internal pins of the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing clock. Only one source is accepted. Wildcards are accepted as long as the resolution shows one pin.

## Supported Families

IGLOO, ProASIC3,SmartFusion, Fusion, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, eX, SX-A

## Description

Creates a generated clock in the current design at a declared source by defining its frequency with respect to the frequency at the reference pin. The static timing analysis tool uses this information to compute and propagate its waveform across the clock network to the clock pins of all sequential elements driven by this source.

The generated clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

## Exceptions

- None

## Examples

The following example creates a generated clock on pin U1/reg1:Q with a period twice as long as the period at the reference port CLK.

```
create_generated_clock -name {my_user_clock} –divide_by 2 –source [get_ports {CLK}]
 U1/reg1:Q
```

The following example creates a generated clock at the primary output of myPLL with a period ¾ of the period at the reference pin clk.

```
create_generated_clock –divide_by 3 –multiply_by 4  -source clk [get_pins {myPLL:CLK1}]
```

## Actel Implementation Specifics

- SDC accepts either –multiply_by or –divide_by option. In Actel design implementation, both are accepted to accurately model the PLL behavior.
- SDC accepts defining a generated clock on many sources using a single command. In Actel design implementation, only one source is accepted.
- The -duty_cycle ,-edges and –edge_shift options in the SDC create_generated_clock command are not supported in Actel design implementation.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

Create Generated Clock Constraint (SDC)

# remove_clock_uncertainty

Removes a clock-to-clock uncertainty from the current timing scenario.

```
remove_clock_uncertainty -from | -rise_from | -fall_from from_clock_list -to | -rise_to| -
fall_to to_clock_list -setup {value} -hold {value}
remove_clock_uncertainty -id constraint_ID
```

## Arguments

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

*`from_clock_list`*

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to` , or `-fall_to` arguments for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. You can specify only one of the `-to`, `-rise_to` , or `-fall_to` arguments for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to` , or `-fall_to` arguments for the constraint to be valid.

*`to_clock_list`*

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-id` *`constraint_ID`*

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either the exact parameters to set the constraint or its constraint ID.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, RTAX-S, eX (for analysis), SX-A (for analysis)

## Description

Removes a clock-to-clock uncertainty from the specified clock in the current scenario. If the specified arguments do not match clocks with an uncertainty constraint in the current scenario, or if the specified ID does not refer to a clock-to-clock uncertainty constraint, this command fails.

Do not specify both the exact arguments and the ID.

## Exceptions

- None

## Examples

```
remove_clock_uncertainty -from Clk1 -to Clk2
remove_clock_uncertainty -from Clk1 -fall_to { Clk2 Clk3 } -setup
remove_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
remove_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 } -setup
```

```
remove_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
remove_clock_uncertainty -id $clockId
```

## See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

set_clock_uncertainty

# set_clock_latency

Defines the delay between an external clock source and the definition pin of a clock within SmartTime.

```
set_clock_latency -source [-rise][-fall][-early][-late] delay clock
```

## Arguments

`-source`

Specifies a clock source latency on a clock pin.

`-rise`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-fall`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`-late`

Optional. Specifies that the latency is late bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`-early`

Optional. Specifies that the latency is early bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

*`delay`*

Specifies the latency value for the constraint.

*`clock`*

Specifies the clock to which the constraint is applied. This clock must be constrained.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC[PLUS], Axcelerator, ProASIC (for analysis), eX, SX-A

## Description

Clock source latency defines the delay between an external clock source and the definition pin of a clock within SmartTime. It behaves much like an input delay constraint. You can specify both an "early" delay and a"late" delay for this latency, providing an uncertainty which SmartTime propagates through its calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.

## Exceptions

- None

## Examples

The following example sets an early clock source latency of 0.4 on the rising edge of main_clock. It also sets a clock source latency of 1.2, for both the early and late values of the falling edge of main_clock. The late value for the clock source latency for the falling edge of main_clock remains undefined.

```
set_clock_latency –source –rise –early 0.4 { main_clock  }
set_clock_latency –source –fall    1.2 { main_clock }
```

## Actel Implementation Specifics

- SDC accepts a list of clocks to -set_clock_latency. In Actel design implementation, only one clock pin can have its source latency specified per command.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# set_clock_uncertainty

Defines the timing uncertainty between two clock waveforms or maximum skew.

```
set_clock_uncertainty  uncertainty (-from | -rise_from | -fall_from) from_clock_list (-to |
-rise_to | -fall_to) to_clock_list [-setup | -hold]
```

## Arguments

*uncertainty*

Specifies the time in nanoseconds that represents the amount of variation between two clock edges. The value must be a positive floating point number.

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid. This option is the default.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. You can specify only one of the `-from`, `-rise_from`, or `-fall_from` arguments for the constraint to be valid.

*from_clock_list*

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to` , or `-fall_to` arguments for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. You can specify only one of the `-to`, `-rise_to` , or `-fall_to` arguments for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. You can specify only one of the `-to`, `-rise_to` , or `-fall_to` arguments for the constraint to be valid.

*`to_clock_list`*

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If you do not specify either option (`-setup` or `-hold` ) or if you specify both options, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If you do not specify either option (`-setup` or `-hold` ) or if you specify both options, the uncertainty applies to both setup and hold checks.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC[PLUS], Axcelerator, ProASIC (for analysis), eX, SX-A

## Description

Clock uncertainty defines the timing between an two clock waveforms or maximum clock skew.

Both setup and hold checks must account for clock skew. However, for setup check, SmartTime looks for the smallest skew. This skew is computed by using the maximum insertion delay to the launching sequential component and the shortest insertion delay to the receiving component.

For hold check, SmartTime looks for the largest skew. This skew is computed by using the shortest insertion delay to the launching sequential component and the largest insertion delay to the receiving component. SmartTime makes this distinction automatically.

## Exceptions

- None

## Examples

The following example defines two clocks and sets the uncertainty constraints, which analyzes the inter-clock domain between clk1 and clk2.

```
create_clock –period 10 clk1
create_generated_clock –name clk2 -source clk1 -multiply_by 2 sclk1
set_clock_uncertainty 0.4 -rise_from clk1 -rise_to clk2
```

## Actel Implementation Specifics

- SDC accepts a list of clocks to -set_clock_uncertainty.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

create_clock (SDC)

create_generated_clock (SDC)

remove_clock_uncertainty

# set_disable_timing

Disables timing arcs within the specified cell and returns the ID of the created constraint if the command succeeded.

```
set_disable_timing [-from from_port] [-to to_port] cell_name
```

## Arguments

`-from from_port`

Specifies the starting port.

`-to to_port`

Specifies the ending port.

*`cell_name`*

Specifies the name of the cell in which timing arcs will be disabled.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, and RTAX-S

## Description

This command disables the timing arcs in the specified cell, and returns the ID of the created constraint if the command succeeded. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

## Examples

The following example disables the arc between a2:A and a2:Y.

```
set_disable_timing -from port1 -to port2 cellname
```

This command ensures that the arc is disabled within a cell instead of between cells.

## Actel Implementation Specifics

- None

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# set_false_path

Identifies paths that are considered false and excluded from the timing analysis.

```
set_false_path [-from from_list] [-through through_list] [-to to_list]
```

## Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC[PLUS], Axcelerator, ProASIC (for analysis), eX (-through option), SX-A (-through option)

## Description

The set_false_path command identifies specific timing paths as being false. The false timing paths are paths that do not propagate logic level changes. This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins, and the path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints. If more than one object is specified within one -through option, the path can pass through any objects.

## Examples

The following example specifies all paths from clock pins of the registers in clock domain clk1 to data pins of a specific register in clock domain clk2 as false paths:

```
set_false_path –from [get_clocks {clk1}] –to reg_2:D
```

The following example specifies all paths through the pin U0/U1:Y to be false:

```
set_false_path -through U0/U1:Y
```

## Actel Implementation Specifics

- SDC accepts multiple -through options in a single constraint to specify paths that traverse multiple points in the design. In Actel design implementation, only one –through option is accepted.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

Set False Path Constraint

# set_input_delay

Defines the arrival time of an input relative to a clock.

```
set_input_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] input_list
```

## Arguments

`delay_value`

Specifies the arrival time in nanoseconds that represents the amount of time for which the signal is available at the specified input after a clock edge.

`-clock clock_ref`

Specifies the clock reference to which the specified input delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

`-max`

Specifies that delay_value refers to the longest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-min`

Specifies that delay_value refers to the shortest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

`input_list`

Provides a list of input ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC[PLUS], ProASIC (for analysis),Axcelerator, eX (for analysis), SX-A (for analysis)

## Description

The set_input_delay command sets input path delays on input ports relative to a clock edge. This usually represents a combinational path delay from the clock pin of a register external to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds input delay to path delay for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be:

- a single port name used as source for a clock constraint
- a single pin name used as source for a clock constraint; for instance reg1:CLK. This name can be hierarchical (for instance toplevel/block1/reg2:CLK)
- an object accessor that will refer to one clock: [get_clocks {clk}]

## Examples

The following example sets an input delay of 1.2ns for port data1 relative to the rising edge of CLK1:

```
set_input_delay 1.2 -clock [get_clocks CLK1] [get_ports data1]
```

The following example sets a different maximum and minimum input delay for port IN1 relative to the falling edge of CLK2:

```
set_input_delay 1.0 -clock_fall -clock CLK2 –min {IN1}
set_input_delay 1.4 -clock_fall -clock CLK2 –max {IN1}
```

## Actel Implementation Specifics

- In SDC, the -clock is an optional argument that allows you to set input delay for combinational designs. Actel implementation currently requires this argument.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

Set Input Delay

# set_load

Sets the load to a specified value on a specified port.

```
set_load capacitance port_list
```

## Arguments

*capacitance*

Specifies the capacitance value that must be set on the specified ports.

*port_list*

Specifies a list of ports in the current design on which the capacitance is to be set.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, RTAX-S, eX, and SX-A

## Description

The load constraint enables the Designer software to account for external capacitance at a specified port. You cannot set load constraint on the nets. When you specify this constraint on the output ports, it impacts the delay calculation on the specified ports.

## Examples

The following examples show how to set output capacitance on different output ports:

```
set_load  35   out_p
set_load  40   {O1 02}
set_load  25   [get_ports out]
```

## Actel Implementation Specifics

- In SDC, you can use the set_load command to specify capacitance value on nets. Actel implementation only supports output ports.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

Set Load on Port

# set_max_delay (SDC)

Specifies the maximum delay for the timing paths.

```
set_max_delay delay_value [-from from_list] [-to to_list]
```

## Arguments

*delay_value*

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC$^{PLUS}$, ProASIC (for analysis), Axcelerator, eX (-through option), SX-A (-through option)

## Description

This command specifies the required maximum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The tool automatically derives the individual maximum delay targets from clock waveforms and port input or output delays. For more information, refer to the create_clock, set_input_delay, and set_output_delay commands.

The maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

## Examples

The following example sets a maximum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_max_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a maximum delay by constraining all paths to output ports whose names start by "out" with a delay less than 3.8 ns:

```
set_max_delay 3.8 -to [get_ports out*]
```

## Actel Implementation Specifics

- The –through option in the set_max_delay SDC command is not supported.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

Set Max Delay

# set_min_delay

Specifies the minimum delay for the timing paths.

```
set_min_delay delay_value [-from from_list] [-to to_list]
```

## Arguments

*delay_value*

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC$^{PLUS}$, ProASIC (for analysis), Axcelerator, eX (-through option), SX-A (-through option)

## Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The tool automatically derives the individual minimum delay targets from clock waveforms and port input or output delays. For more information, refer to the create_clock, set_input_delay, and set_output_delay commands.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

## Examples

The following example sets a minimum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by "out" with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

## Actel Implementation Specifics

- The –through option in the set_min_delay SDC command is not supported.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# set_multicycle_path

Defines a path that takes multiple clock cycles.

```
set_multicycle_path ncycles [-setup] [-hold] [-from from_list] [–through through_list] [-to to_list]
```

## Arguments

*ncycles*

Specifies an integer value that represents a number of cycles the data path must have for setup or hold check. The value is relative to the starting point or ending point clock, before data is required at the ending point.

`-setup`

Optional. Applies the cycle value for the setup check only. This option does not affect the hold check. The default hold check will be applied unless you have specified another set_multicycle_path command for the hold value.

`-hold`

Optional. Applies the cycle value for the hold check only. This option does not affect the setup check.

Note: If you do not specify "-setup" or "-hold", the cycle value is applied to the setup check and the default hold check is performed (*ncycles* -1).

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins or ports through which the multiple cycle paths must pass.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC$^{PLUS}$, ProASIC (for analysis), Axcelerator, eX (for analysis), SX-A (for analysis)

## Description

Setting multiple cycle paths constraint overrides the single cycle timing relationships between sequential elements by specifying the number of cycles that the data path must have for setup or hold checks. If you change the multiplier, it affects both the setup and hold checks.

False path information always takes precedence over multiple cycle path information. A specific maximum delay constraint overrides a general multiple cycle path constraint.

If you specify more than one object within one -through option, the path passes through any of the objects.

## Examples

The following example sets all paths between reg1 and reg2 to 3 cycles for setup check. Hold check is measured at the previous edge of the clock at reg2.

```
set_multicycle_path 3 -from [get_pins {reg1}] –to [get_pins {reg2}]
```

The following example specifies that four cycles are needed for setup check on all paths starting at the registers in the clock domain ck1. Hold check is further specified with two cycles instead of the three cycles that would have been applied otherwise.

```
set_multicycle_path 4 -setup -from [get_clocks {ck1}]
set_multicycle_path 2 -hold -from [get_clocks {ck1}]
```

## Actel Implementation Specifics

- SDC allows multiple priority management on the multiple cycle path constraint depending on the scope of the object accessors. In Actel design implementation, such priority management is not supported. All multiple cycle path constraints are handled with the same priority.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

Set Multicycle Path

# set_output_delay

Defines the output delay of an output relative to a clock.

```
set_output_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] output_list
```

## Arguments

`delay_value`

Specifies the amount of time before a clock edge for which the signal is required. This represents a combinational path delay to a register outside the current design plus the library setup time (for maximum output delay) or hold time (for minimum output delay).

`-clock clock_ref`

Specifies the clock reference to which the specified output delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

`-max`

Specifies that delay_value refers to the longest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

`-min`

Specifies that delay_value refers to the shortest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

*`output_list`*

Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC[PLUS], ProASIC (for analysis), Axcelerator, eX (for analysis), SX-A (for analysis)

## Description

The set_output_delay command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless you specify it. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds output delay to path delay for paths ending at primary outputs.

## Examples

The following example sets an output delay of 1.2ns for port OUT1 relative to the rising edge of CLK1:

```
set_output_delay 1.2 -clock [get_clocks CLK1] [get_ports OUT1]
```

The following example sets a different maximum and minimum output delay for port OUT1 relative to the falling edge of CLK2:

```
set_output_delay 1.0 -clock_fall -clock CLK2 –min {OUT1}
```

```
set_output_delay 1.4 -clock_fall -clock CLK2 –max {OUT1}
```

## Actel Implementation Specifics

- In SDC, the -clock is an optional argument that allows you to set the output delay for combinational designs. Actel implementation currently requires this option.

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

Set Output Delay

# Design Object Access Commands

Design object access commands are SDC commands. Most SDC constraint commands require one of these commands as command arguments.

Designer software supports the following SDC access commands:

| Design Object | Access Command |
|---|---|
| Cell | get_cells |
| Clock | get_clocks |
| Net | get_nets |
| Port | get_ports |
| Pin | get_pins |
| Input ports | all_inputs |
| Output ports | all_outputs |
| Registers | all_registers |

## See Also

About SDC Files

# all_inputs

Returns all the input or inout ports of the design.

```
all_inputs
```

## Arguments

- None

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC[PLUS], ProASIC, Axcelerator, eX, SX-A

## Exceptions

- None

## Example

```
set_max_delay -from [all_inputs] -to [get_clocks ck1]
```

## Actel Implementation Specifics

- None

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# all_registers

Returns either a collection of register cells or register pins, whichever you specify.

```
all_registers [-clock clock_name] [-cells] [-data_pins ]
 [-clock_pins] [-async_pins] [-output_pins]
```

## Arguments

`-clock clock_name`

Creates a collection of register cells or register pins in the specified clock domain.

`-cells`

Creates a collection of register cells. This is the default. This option cannot be used in combination with any other option.

`-data_pins`

Creates a collection of register data pins.

`-clock_pins`

Creates a collection of register clock pins.

`-async_pins`

Creates a collection of register asynchronous pins.

`-output_pins`

Creates a collection of register output pins.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC[PLUS], ProASIC, Axcelerator, eX, SX-A

## Description

This command creates either a collection of register cells (default) or register pins, whichever is specified. If you do not specify an option, this command creates a collection of register cells.

## Exceptions

- None

## Examples

```
set_max_delay 2 -from [all_registers] -to [get_ports {out}]
set_max_delay 3 –to [all_registers –async_pins]
set_false_path –from [all_registers –clock clk150]
set_multicycle_path –to [all_registers –clock c* -data_pins
 –clock_pins]
```

## Actel Implementation Specifics

- None

## See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# all_registers

Returns either a collection of register cells or register pins, whichever you specify.

```
all_registers [-clock clock_name] [-cells] [-data_pins ]
 [-clock_pins] [-async_pins] [-output_pins]
```

## Arguments

`-clock clock_name`

Creates a collection of register cells or register pins in the specified clock domain.

`-cells`

Creates a collection of register cells. This is the default. This option cannot be used in combination with any other option.

`-data_pins`

Creates a collection of register data pins.

`-clock_pins`

Creates a collection of register clock pins.

`-async_pins`

Creates a collection of register asynchronous pins.

`-output_pins`

Creates a collection of register output pins.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC[PLUS], ProASIC, Axcelerator, eX, SX-A

## Description

This command creates either a collection of register cells (default) or register pins, whichever is specified. If you do not specify an option, this command creates a collection of register cells.

## Exceptions

- None

## Examples

```
set_max_delay 2 -from [all_registers] -to [get_ports {out}]
set_max_delay 3 –to [all_registers –async_pins]
set_false_path –from [all_registers –clock clk150]
set_multicycle_path –to [all_registers –clock c* -data_pins
 –clock_pins]
```

## Actel Implementation Specifics

- None

## See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# get cells

Returns the cells (instances) specified by the pattern argument.

```
get_cells pattern
```

## Arguments

*pattern*

Specifies the pattern to match the instances to return. For example, "get_cells U18*" returns all instances starting with the characters "U18", where "*" is a wildcard that represents any character string.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, RTAX-S, eX, and SX-A

## Description

This command returns a collection of instances matching the pattern you specify. You can only use this command as part of a –from, -to, or –through argument for the following constraint exceptions: set_max delay, set_multicycle_path, and set_false_path design constraints.

## Exceptions

- None

## Examples

```
set_max_delay 2 -from [get_cells {reg*}] -to [get_ports {out}]
set_false_path –through [get_cells {Rblock/muxA}]
```

## Actel Implementation Specifics

- None

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# get_clocks

Returns the specified clock.

```
get_clocks pattern
```

## Arguments

*pattern*

Specifies the pattern to match to the Timer or SmartTime on which a clock constraint has been set.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, RTAX-S, eX, and SX-A

## Description

- If this command is used as a –from argument in maximum delay (set_max_path_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the clock pins of all the registers related to this clock are used as path start points.
- If this command is used as a –to argument in maximum delay (set_max_path_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the synchronous pins of all the registers related to this clock are used as path endpoints.

## Exceptions

- None

## Example

```
set_max_delay -from [get_ports datal] -to \
[get_clocks ck1]
```

## Actel Implementation Specifics

- None

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# get_pins

Returns the specified pins.

```
get_pins pattern
```

## Arguments

*pattern*

Specifies the pattern to match the pins.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, RTAX-S, eX, and SX-A

## Exceptions

- None

## Example

```
create_clock -period 10 [get_pins clock_gen/reg2:Q]
```

## Actel Implementation Specifics

- None

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# get_nets

Returns the named nets specified by the pattern argument.

```
get_nets pattern
```

## Arguments

*pattern*

Specifies the pattern to match the names of the nets to return. For example, "get_nets N_255*" returns all nets starting with the characters "N_255", where "*" is a wildcard that represents any character string.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, RTAX-S, eX, and SX-A

## Description

This command returns a collection of nets matching the pattern you specify. You can only use this command as source objects in create clock (create_clock) or create generated clock (create_generated_clock) constraints and as -through arguments in set false path (set_false_path), set minimum delay (set_min_delay), set maximum delay (set_max_delay), and set multicycle path (set_multicycle_path) constraints.

## Exceptions

- None

## Examples

```
set_max_delay 2 -from [get_ports RDATA1] -through [get_nets {net_chkp1 net_chkqi}]
set_false_path –through [get_nets {Tblk/rm/n*}]
create_clcok -name mainCLK -per 2.5 [get_nets {cknet}]
```

## Actel Implementation Specifics

- None

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# get_ports

Returns the specified ports.

```
get_ports pattern
```

## Argument

*pattern*

Specifies the pattern to match the ports. This is equivalent to the macros $in()[<pattern>] when used as –from argument and $out()[<pattern>] when used as –to argument or $ports()[<pattern>] when used as a –through argument.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, RTAX-S, eX, and SX-A

## Exceptions

- None

## Example

```
create_clock -period 10[get_ports CK1]
```

## Actel Implementation Specifics

- None

### See Also

Constraint Support by Family

Constraint Entry Table

SDC Syntax Conventions

# About Physical Design Constraint (PDC) Files

A PDC file is a Tcl script file specifying physical constraints. This file can be imported and exported from Designer. Any constraint that you can enter using the PinEditor in MVN or ChipPlanner tool, you can also use in a PDC file.

Note: Note: Only IGLOO, Fusion, ProASIC3, and Axcelerator devices support PDC files. Designer supports the following PDC commands.

| Command | Action |
|---|---|
| assign_global_clock | Assigns regular nets to global clock networks by promoting the net using a CLKINT macro |
| assign_local_clock | Assigns regular nets to local clock routing (Axcelerator) or to LocalClock regions (IGLOO, Fusion, and ProASIC3) |
| assign_net_macros | Assigns the macros connected to a net to a specified defined region |
| assign_quadrant_clock | Assigns regular nets to a specific quadrant clock region (IGLOO, Fusion, and ProASIC3) |
| assign_region | Assigns macros to a pre-specified region |
| define_region | Defines either a rectangular or rectilinear region |
| delete_buffer_tree | Removes all buffers and inverters from a given net for IGLOO, Fusion, and ProASIC3E devices |
| dont_touch_buffer_tree | Restores all buffers and inverters that were removed from a given net with the delete_buffer_tree command |
| move_block | Moves only the block core (COMB, SEQ) of the specified instance (I/Os or PLLs) to the specified location on the chip |
| move_region | Moves a region to new coordinates |
| reset_floorplan | Deletes all defined regions. Placed macros are not affected. |
| reset_io | Resets all attributes on a macro to the default values |

| Command | Action |
|---|---|
| reset_iobank | Resets an I/O banks technology to the default technology |
| reset_net_critical | Resets net criticality to default level |
| set_io | Sets the attributes of an I/O |
| set_iobank | Specifies the I/O bank's technology and sets the VREF pins for the specified banks |
| set_location | Places a given logic instance at a particular location |
| set_block_options | Overrides the compile option for either a specific block or an instance of a block |
| set_multitile_location | Assigns specified two-tile and four-tile macros to specified locations on the chip |
| set_port_block | Sets properties on a port in the Block flow |
| set_preserve | Preserves instances before compile so that instances are not combined |
| set_net_critical | Sets net criticality, which is issued to influence placement and routing in favor of performance |
| set_reserve | Reserves the specified pins in the design |
| set_unreserve | Resets the specified pins in the design that were previously reserved |
| unassign_global_clock | Assigns clock nets to regular nets |
| unassign_local_clock | Unassigns the specified user-defined net from a LocalClock or QuadrantClock region |
| unassign_macro_from_region | Unassigns macros from a specified region, if they are assigned to that region |
| unassign_net_macros | Unassigns macros connected to a specified net from a defined region |

| Command | Action |
|---|---|
| unassign_quadrant_clock | Unassigns the specified net from a QuadrantClock region |
| undefine_region | Removes the specified region |

Note: Note: PDC commands are case sensitive. However, their arguments are not.

## See Also

Constraint Entry

PDC Syntax Conventions

PDC Naming Conventions

Importing Constraint Files

# PDC Syntax Conventions

The following table shows the typographical conventions that are used for the PDC command syntax.

| Syntax Notation | Description |
|---|---|
| command<br>-argument | Commands and arguments appear in Courier New typeface. |
| variable | Variables appear in blue, italic Courier New typeface. You must substitute an appropriate value for the variable. |
| [-argument *value*]<br>[*variable*]+ | Optional arguments begin and end with a square bracket with one exception: if the square bracket is followed by a plus sign (+), then users must specify at least one argument. The plus sign (+) indicates that items within the square brackets can be repeated. Do not enter the plus sign character. |

Note: Note: PDC commands are case sensitive. However, their arguments are not.

## Examples

Syntax for the assign_local_clock (Axcelerator) command followed by a sample command:

```
assign_local_clock -type value -net netname [LocalClock_region ]+
```

```
assign_local_clock -type hclk -net  reset_n tile1a tile2a
```

Syntax for the set_io (Axcelerator) command followed by a sample command:

```
set_io portname [-iostd value][-register value][-out_drive value][-slew value][-res_pull
value][-out_load value][-pinname value][-fixed value][-in_delay value]
```

```
set_io ADDOUT2 \
-iostd PCI \
-register yes \
-out_drive 16 \
-slew high \
-out_load 10 \
-pinname T21 \
-fixed yes
```

## Wildcard Characters

You can use the following wildcard characters in names used in PDC commands:

| Wildcard | What It Does |
|---|---|
| \ | Interprets the next character literally |
| ? | Matches any single character |
| * | Matches any string |
| [] | Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range) |

Note: Note: The matching function requires that you add a slash (\) before each slash in the port, instance, or net name when using wildcards in a PDC command and when using wildcards in the Find feature of the MultiView Navigator. For example, if you have an instance named "A/B12" in the netlist, and you enter that name as "A\\/B*" in a PDC command, you will not be able to find it. In this case, you must specify the name as A\\\\/B*.

## Special Characters ([ ], { }, and \)

Sometimes square brackets are part of the command syntax. In these cases, you must either enclose the open and closed square brackets characters with curly brackets or precede the open and closed square brackets characters with a backslash (\). If you do not, you will get an error message.

For example:

```
set_iobank {mem_data_in[57]} -fixed no 7 2
or
set_iobank mem_data_in\[57\] -fixed no 7 2
```

## Entering Arguments on Separate Lines

To enter an argument on a separate line, you must enter a backslash (\) character at the end of the preceding line of the command as shown in the following example:

```
set_io ADDOUT2 \
-iostd PCI \
-register Yes \
-out_drive 16 \
-slew High \
-out_load 10 \
-pinname T21 \
-fixed yes
```

### See Also

About PDC Files

PDC Naming Conventions

# PDC Naming Conventions

Note: The names of ports, instances, and nets in an imported netlist are sometimes referred to as their original names.

## Rules for Displaying Original Names

Port names appear exactly as they are defined in a netlist. For example, a port named A/B appears as A/B in ChipPlanner, PinEditor, and I/O Attribute Editor in MultiView Navigator.

Instances and nets display the original names plus an escape character (\) before each backslash (/) and each slash (\) that is not a hierarchy separator. For example, the instance named `A/\B` is displayed as `A\/\\B`.

## Which Name Do I Use in PDC Commands?

The names of ports, instances, and nets in a netlist displayed in MultiView Navigator (MVN) for IGLOO, Fusion, ProASIC3 and Axcelerator devices are names taken directly from the imported netlist.

## Using PDC Commands

When writing PDC commands, follow these rules:

- Always use the macro name as it appears in the netlist. (See "Merged elements" in this topic for exceptions.)
- Names from a netlist: For port names, use the names exactly as they appear in the netlist. For instance and net names, add an escape character (\) before each backslash (\) and each slash (/) that is not a hierarchy separator.
- Names from MVN and compile report: Use names as they appear in MultiView Navigator or the compile report.
- For wildcard names, always add an extra backslash (\) before each backslash.
- Always apply the PDC syntax conventions to any name in a PDC command.

The following table provides examples of names as they appear in an imported netlist and the names as they should appear in a PDC file:

| Type of name and its location | Name in the imported netlist | Name to use in PDC file |
|---|---|---|
| Port name in netlist | A/:B1 | A/:B1 |
| Port name in MVN | A/:B1 | A/:B1 |
| Instance name in a netlist | A/:B1 A$(1) | A\\/:B1 A$(1) |
| Instance name in the netlist but using a wildcard character in a PDC file | A/:B1 | A\\\\/:B* |
| Instance name in MVN or a compile report | A\/:B1 | A\\/:B1 |

| Type of name and its location | Name in the imported netlist | Name to use in PDC file |
|---|---|---|
| Net name in a netlist | Net1/:net1 | Net1\\/:net1 |
| Net name in MVN or a compile report | Net1\/:net1 | Net1\\/:net1 |

When exporting PDC commands, the software always exports names using the PDC rules described in this topic.

## Case Sensitivity When Importing PDC Files

The following table shows the case sensitivity in the PDC file based on the source netlist.

| File Type | Case Sensitivity |
|---|---|
| Verilog | Names in the netlist are case sensitive. |
| Edif | Names in the netlist are always case sensitive because we use the Rename clause, which is case sensitive. |
| Vhdl | Names in the netlist are not case sensitive unless those names appear between slashes (\). |

For example, in VHDL, capital "A" and lowercase "a" are the same name, but \A\ and \a\ are two different names. However, in a Verilog netlist, an instance named "A10" will fail if spelled as "a10" in the set_location command:

```
set_location A10 (This command will succeed.)
set_location a10 (This command will fail.)
```

## Which Name to Use in the Case of Merged Elements (IGLOO, Fusion, and ProASIC3 Only)

The following table indicates which name to use in a PDC command when performing the specified operation:

| Operation | Name to Use |
|---|---|
| I/O connected to PLL with a hardwired connection | PLL instance name |
| I/O combined with FF or DDR | I/O instance name |
| Global promotion | |

### See Also

About PDC Files

PDC Syntax Conventions

# assign_global_clock

Assigns regular nets to global clock networks by promoting the net using a CLKINT macro.

```
assign_global_clock -net netname
```

## Arguments

`-net netname`

Specifies the name of the net to promote to a global clock network. The net is promoted using a CLKINT macro, which you can place on a chip-wide clock location.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion

## Exceptions

- The assign_global_clock command is not supported in auxiliary PDC files.

## Examples

```
assign_global_clock -net globalReset
```

### See Also

Assign Net to Global Clock

assign_local_clock (Axcelerator)

assign_local_clock (IGLOO, Fusion, and ProASIC3)

unassign_global_clock

PDC Syntax Conventions

PDC Naming Conventions

# assign_local_clock (IGLOO, ProASIC3, SmartFusion and Fusion)

Assigns regular nets to LocalClock regions.

```
assign_local_clock -net  netname-type clock_type clock_region
```

## Arguments

-net *netname*

Specifies the name of the net to assign to a LocalClock region.

-type *clock_type*

Specifies the type of region to which the net will be assigned:

| Value | Description |
|---|---|
| chip | Specifies a LocalClock region driven by a clock rib located on the middle of the chip |
| quadrant | Specifies one of the following:<br>• A QuadrantClock region<br>• A LocalClock region driven by a clock rib located on the top or bottom of the chip |

*clock_region*

Specifies a LocalClock region.

LocalClock regions are defined as one of the following:

- A single spine defined as T# (Top spine) or B# (Bottom spine)
- A multi-spine rectangle defined as [T | B]#:[T | B]#

Spines are numbered from left to right starting at 1. The maximum spine number is a function of the die selected by the user. Please refer to the examples in this help topic.

Local clock uses clock spine resources remaining after global assignment from Input Netlist and PDC constraints. There are six chip-wide and twelve quadrant-wide clock resources per device. You may reserve portions of a clock network (chip-wide or quadrant-wide) for local clocks by assigning clock nets to regions. If there are not enough clock resources to honor all local clock assignments, the Layout command will fail.

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Note: Note: You must import the PDC file along with the netlist as a source file because Compile needs to delete buffers and legalize the netlist. Shared instances between local clocks are supported. Compile will insert buffers to legalize the netlist.

## Exceptions

- If the net is a clock net, it is demoted to a regular net. You will see an unassign_global_net command in the PDC file if the net is demoted to a regular net by the compiler and the assignment to local clock failed.

## Examples

This example assigns the net named localReset to the chip-wide spine T1:

```
assign_local_clock -net localReset -type chip T1
```

This example assigns the net named localReset to the quadrant spines T1, T2, T3, T4, and T5, which span more than one quadrant:

```
assign_local_clock -net localReset -type quadrant T1:T5
```

This example assigns the net named localReset to the chip-wide spines T1, T2, T3, T4, T5, T6, B1, B2, B3, B4, B4, and B6:

```
assign_local_clock -net localReset -type chip T1:B6
```

### See Also

Assign Net to Local Clock

unassign_local_clock

assign_quadrant_clock

PDC Syntax Conventions

PDC Naming Conventions

# assign_local_clock (Axcelerator)

Assigns regular nets to local clock routing.

```
assign_local_clock -type value -net netname[ LocalClock_region ]+
```

Note: Note: Along with the netlist, import PDC files as source files because the compiler needs to insert or delete buffers as well as legalize the netlist.

## Arguments

`-type value`

Specifies the type of clock to which the net will be assigned. You can enter one of the following values:

| Value | Description |
|---|---|
| hclk | Indicates a hardwired clock. |
| rclk | Indicates a routed clock. |

Note: Note: Nets currently assigned to hclk or rclk are not demoted. Therefore, nets currently assigned to routed clocks or hardwired clocks cannot be assigned to local clocks. Also, hardwired clock (hclk) networks can only drive clock pins.

`-net netname`

Specifies the name of the net to assign to a LocalClock region. You must specify a net name that currently exists in the design.

`LocalClock_region`

You must specify at least one LocalClock region. You can define the LocalClock region at either the tile level or at the row or column level within a tile.

You can define the LocalClock region at the tile level as follows: tile `<number><letter>`. The tiles are numbered 1,2,3, .. in the Y axis and A, B, C, … in the X axis. Tile1A is the lower-left tile. In addition, you can cascade LocalClock regions by specifying into which tiles to assign the user-specified net. Refer to the examples below.

You can also define the LocalClock region at the row or column level within a tile as follows: tile `<number><letter>.`row`<number> | `col`<number>`. Whether you select a row or a column depends on whether you are specify hclk or rclk for the LocalClock region. Refer to the examples below. When defining a row, ensure that the LocalClock region is composed of at least two consecutive rows.

LocalClock assignment uses resources remaining after global assignment from Input Netlist. Axcelerator devices can include a total of four routed clock and four hardwired clock networks.

## Supported Families

Axcelerator

## Exceptions

- The assign_local_clock command is not supported in auxiliary PDC files. If importing a PDC file that includes this command, you must import it as a source file.

## Examples

You can cascade tiles to create one LocalClock region as follows:

```
assign_local_clock -type hclk -net  reset_n tile1a tile2a
assign_local_clock -type rclk -net  reset_n tile1a tile2a
```

You can assign a net to one tile as follows:

```
assign_local_clock -type rclk -net  reset_n tile1a
assign_local_clock -type hclk -net  reset_n tile2c
```

You can assign a net to a column within a tile as follows:

```
assign_local_clock -type hclk -net  reset_n tile1a.col7  tile2a.col9
```

You can assign a net to a row within a tile as follows:

```
assign_local_clock -type rclk -net  reset_n tile1a.row4 tile1a.row5
```

### See Also

Assign Net to Local Clock

assign_global_clock

assign_local_clock (Fusion, IGLOO, and ProASIC3)

unassign_local_clock

PDC Syntax Conventions

PDC Naming Conventions

# assign_net_macros

Assigns to a user-defined region all the macros that are connected to a net.

```
assign_net_macros region_name [net1]+ [-include_driver value]
```

## Arguments

*region_name*

Specifies the name of the region to which you are assigning macros. The region must exist before you use this command. See define_region (rectangular) or define_region (rectilinear). Because the define_region command returns a region object, you can write a simple command such as assign_net_macros [define_region]+ [net]+

*net1*

You must specify at least one net name. Net names are AFL-level (Actel flattened netlist) names. These names match your netlist names most of the time. When they do not, you must export AFL and use the AFL names. Net names are case insensitive. Hierarchical net names from ADL are not allowed. You can use the following wildcard characters in net names:

| Wilcard | What It Does |
|---|---|
| \ | Interprets the next character as a non-special character |
| ? | Matches any single character |
| * | Matches any string |
| [] | Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range) |

*net1*

Specifies whether to add the driver of the net(s) to the region. You can enter one of the following values:

| Value | Description |
|---|---|
| Yes | Include the driver in the list of macros assigned to the region (default) . |
| No | Do not assign the driver to the region. |

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator

## Exceptions

- Placed macros (not connected to the net) that are inside the area occupied by the net region are automatically unplaced.
- Net region constraints are internally converted into constraints on macros. PDC export results as a series of assign_region <region_name> macro1 statements for all the connected macros.
- If the region does not have enough space for all of the macros, or if the region constraint is impossible, the constraint is rejected and a warning message appears in the Log window.
- For overlapping regions, the intersection must be at least as big as the overlapping macro count.
- If a macro on the net cannot legally be placed in the region, it is not placed and a warning message appears in the Log window.
- Net region constraints may result in a single macro being assigned to multiple regions. These net region constraints result in constraining the macro to the intersection of all the regions affected by the constraint.

## Examples

```
assign_net_macros cluster_region1 keyin1intZ0Z_62 -include_driver no
```

### See Also

Assign Net to Region

unassign_net_macros

Unassign macros on net from region

PDC Syntax Conventions

PDC Naming Conventions

# assign_quadrant_clock

Assigns regular nets to a specific quadrant clock region.

```
assign_quadrant_clock -net netname -quadrant quadrant_clock_region  [-fixed value]
```

## Arguments

`-net netname`

Specifies the name of the net to assign to a QuadrantClock region. You must specify a net name that currently exists in the design.

`-quadrant quadrant_clock_region`

Specifies the QuadrantClock region to which the net will be assigned. Each die has four quadrants. QuadrantClock regions are defined as one of the following:

- UL: Upper-Left quadrant
- UR: Upper-Right quadrant
- LL: Lower-Left quadrant
- LR: Lower-Right quadrant

For quadrant clock assignments, regular nets are automatically promoted to clock nets driven by an internal clock driver (CLKINT).

There are twelve quadrant-wide clock resources per device. You may reserve portions of a clock network for quadrant clocks by assigning clock nets to regions. If there are not enough clock resources to honor all local clock assignments, the Layout command will fail.

`-fixed value`

Specifies if the specified QuadrantClock region is locked. If you do not want the Global Assigner to remove it, then lock the region. You can enter one of the following values:

| Value | Description |
|---|---|
| yes | The QuadrantClock region is locked. |
| no | The QuadrantClock region is not locked. |

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

## Exceptions

- This command is not supported in auxiliary PDC files.  If importing a PDC file that includes this command, you must import it as a source file.

## Examples

This example assigns the net named FRAMEN_in to the quadrant clock in the upper-left (UL) quadrant of the chip:

```
assign_quadrant_clock -net FRAMEN_in -quadrant UL
```

This example assigns the net named STOPN_in to the quadrant clock in the lower-right (LR) quadrant of the chip:

```
assign_quadrant_clock -net STOPN_in -quadrant LR
```

This example assigns the net named n32 to the quadrant clock in the lower-right (LR) quadrant of the chip and locks it so that the Global Assigner cannot delete the quadrant region:

```
assign_quadrant_clock -net n32 -quadrant LR -fixed yes
```

### See Also

Assign Net to Quadrant Clock

`unassign_quadrant_clock`

PDC Syntax Conventions

PDC Naming Conventions

# assign_region

Constrains a set of macros to a specified region.

```
assign_region region_name [ macro_name]+
```

## Arguments

*region_name*

Specifies the region to which the macros are assigned. The macros are constrained to this region. Because the define_region command returns a region object, you can write a simpler command such as assign_region [define_region]+ [macro_name]+

*macro_name*

Specifies the macro(s) to assign to the region. You must specify at least one macro name. You can use the following wildcard characters in macro names:

| Wildcard | What It Does |
|---|---|
| \ | Interprets the next character as a non-special character |
| ? | Matches any single character |
| * | Matches any string |
| [] | Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range) |

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- The region must be created before you can assign macros to it.
- You can assign only hard macros or their instances to a region. You cannot assign a group name. A hard macro is a logic cell consisting of one or more silicon modules with locked relative placement.
- You can assign a collection of macros by providing a prefix to their names.

## Examples

In the following example, two macros are assigned to a region:

```
assign_region cluster_region1 des01/G_2722_0_and2 des01/data1_53/U0
```

In the following example, all macros whose names have the prefix des01/Counter_1 (or all macros whose names match the expression des01/Counter_1/*) are assigned to a region:

```
assign_region User_region2 des01/Counter_1
```

## See Also

Assign Macro to Region

`unassign_macro_from_region`

PDC Syntax Conventions

PDC Naming Conventions

# define_region

Defines either a rectangular region or a rectilinear region.

```
define_region [-name region_name ] -type region_type [x1 y1 x2 y2]+ [-color value] [-route
value] [-push_place  value]
```

## Arguments

`-name region_name`

Specifies the region name. The name must be unique. Do not use reserved names such as “bank0” and “bank<N>” for region names. If the region cannot be created, the name is empty. A default name is generated if a name is not specified in this argument.

`-type region_type`

Specifies the region type. The default is inclusive. The following table shows the acceptable values for this argument:

| Region Type Value | Description |
|---|---|
| Empty | Empty regions cannot contain macros. |
| Exclusive | Only contains macros assigned to the region. |
| Inclusive | Can contain macros both assigned and unassigned to the region. |

`x1 y1 x2 y2`

Specifies the series of coordinate pairs that constitute the region. These rectangles may or may not overlap. They are given as x1 y1 x2 y2 (where x1, y1 is the lower left and x2 y2 is the upper right corner in row/column coordinates). You must specify at least one set of coordinates.

`-color value`

Specifies the color of the region. The following table shows the recommended values for this argument:

| Color | Decimal Value |
|---|---|
| | 16776960 |
| | 65280 |
| | 16711680 |
| | 16760960 |
| | 255 |
| | 16711935 |

| Color | Decimal Value |
|---|---|
| | 65535 |
| | 33023 |
| | 8421631 |
| | 9568200 |
| | 8323199 |
| | 12632256 |

`-route value`

Specifies whether to direct the routing of all nets internal to a region to be constrained within that region. A net is internal to a region if its source and destination pins are assigned to the region. This option only applies to IGLOO, Fusion, and ProASIC3 families. You can enter one of the following values:

| Constrain Routing Value | Description |
|---|---|
| Yes | Constrain the routing of nets within the region as well as the placement. |
| No | Do not constrain the routing of nets within the region. Only constrain the placement. This is the default value. |

Note: Note: Local clocks and global clocks are excluded from the -route option. Also, interface nets are excluded from the –route option because they cross region boundaries.

An empty routing region is an empty placement region. If -route is "yes", then no routing is allowed inside the empty region. However, local clocks and globals can cross empty regions.

An exclusive routing region is an exclusive placement region (rectilinear area with assigned macros) along with the following additional constraints:

- For all nets internal to the region (the source and all destinations belong to the region), routing must be inside the region (that is, such nets cannot be assigned any routing resource which is outside the region or crosses the region boundaries).
- Nets without pins inside the region cannot be assigned any routing resource which is inside the region or crosses any region boundaries.

An inclusive routing region is an inclusive placement region (rectilinear area with assigned macros) along with the following additional constraints:

- For all nets internal to the region (the source and all destinations belong to the region), routing must be inside the region (that is, such nets cannot be assigned any routing resource which is outside the region or crosses the region boundaries).
- Nets not internal to the region can be assigned routing resources within the region.

`-push_place` *`value`*

Specifies whether to over-constrain placement for routability, contracting or expanding the size of a placement region, depending on the region's type. To use this option, you must also specify the route option (-route yes). This option only applies to IGLOO, Fusion, and ProASIC3 families. You can enter one of the following values:

| Over-constrain Placement Value | Description |
|---|---|
| Yes | Adjust the size of a placement region according to its type. |
| No | Do not adjust the size of a placement region. This is the default value. |

Specifying both `-route yes` and `-push_place yes` usually creates a tighter placement region (for example, a normal MxN Inclusive placement region would shrink to (M-2)x(N-2)). On the other hand, the prohibited region for external nets of Exclusive and Empty Region types would expand to (M+2)x(N+2).

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator and RTAX-S

## Description

Unlocked macros in empty or exclusive regions are unassigned from that region. You cannot create empty regions in areas that contain locked macros.

You can define a rectilinear region only in a PDC file; you cannot define a rectilinear region using the MultiView Navigator tool.

Use inclusive or exclusive region constraints if you intend to assign logic to a region. An inclusive region constraint with no macros assigned to it has no effect. An exclusive region constraint with no macros assigned to it is equivalent to an empty region.

## Exceptions

- If macros assigned to a region exceed the area's capacity, an error message appears in the Log window.

## Examples

The following example defines an empty rectangular region.

```
define_region -name cluster_region1 -type empty 100 46 102 46
```

The following example defines a rectilinear region with the name RecRegion. This region contains two rectangular areas.

```
define_region -name RecRegion -type Exclusive 0 40 3 42 0 77 7 79
```

The following examples define three regions with three different colors:

```
define_region -name UserRegion0 -color 128 50 19 60 25
define_region -name UserRegion1 -color 16711935 11 2 55 29
define_region -name UserRegion2 -color 8388736 61 6 69 19
```

### See Also

Create Region

assign_region

Creating Regions

PDC Syntax Conventions

PDC Naming Conventions

# delete_buffer_tree

Instructs the Compile command to remove all buffers and inverters from a given net. In the IGLOO and ProASIC3 architectures, inverters are considered buffers because all tile inputs can be inverted. This rule is also true for all Flash architectures but not for Antifuse architectures.

```
delete_buffer_tree [netname]+
```

## Arguments

*netname*

Specifies the names of the nets from which to remove buffer or inverter trees. This command takes a list of names. You must specify at least one net name. You can use the following wildcard characters in net names:

| Wildcard | What It Does |
|---|---|
| \ | Interprets the next character as a non-special character |
| ? | Matches any single character |
| * | Matches any string |
| [] | Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range) |

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

## Exceptions

- The delete_buffer_tree command is not supported in auxiliary PDC files.

## Examples

```
delete_buffer_tree net1
delete_buffer_tree netData\[*\]
```

### See Also

don't_touch_buffer_tree

PDC Syntax Conventions

PDC Naming Conventions

# dont_touch_buffer_tree

Undoes the delete_buffer_tree command. That is, it restores all buffers and inverters that were removed from a given net.

Note: Note: This command is not supported in auxiliary PDC files.

```
dont_touch_buffer_tree [netname]+
```

## Arguments

*netname*

Specifies the names of the nets from which to restore buffers or inverters. This command takes a list of names. You must specify at least one net name. Separate each net name with a space. You can use the following wildcard characters in net names:

| Wildcard | What It Does |
|---|---|
| \ | Interprets the next character as a non-special character |
| ? | Matches any single character |
| * | Matches any string |
| [] | Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range) |

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

## Exceptions

- None

## Example

```
dont_touch_buffer_tree net1 net2
dont_touch_buffer_tree netData\[*\]
```

### See Also

delete_buffer_tree

PDC Syntax Conventions

PDC Naming Conventions

# move_block

Moves a Designer block from its original, locked placement by preserving the relative placement between the instances. You can move the Designer block to the left, right, up, or down.

**Note: If possible, routing is preserved when you move the blocks for IGLOO, Fusion and ProASCI3 families.**

```
move_block -inst_name instance_name -up y -down y  -left x -right x  -non_logic value
```

## Arguments

`-inst_name instance_name`

Specifies the name of the instance to move. If you do not know the name of the instance, run a compile report or look at the names shown in the Block tab of the MultiView Navigator Hierarchy view.

`-up y`

Moves the block up the specified number of rows. The value must be a positive integer.

`-down y`

Moves the block down the specified number of rows. The value must be a positive integer.

`-left x`

Moves the block left the specified number of columns. The value must be a positive integer.

`-right x`

Moves the block right the specified number of columns. The value must be a positive integer.

`-non_logic value`

Specifies what to do with the non-logic part of the block, if one exists. The following table shows the acceptable values for this argument:

| Value | Description |
|---|---|
| move | Move the entire block. |
| keep | Move only the logic portion of the block (COMB/SEQ) and keep the rest locked in the same previous location, if there is no conflict with other blocks. |
| unplace | Move only the logic portion of the block (COMB/SEQ) and unplace the rest of it, such as I/Os and RAM. |

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Description

This command moves a Designer block from its original, locked position to a new position.

You can move the entire block or just the logic part of it. You must use the -non_logic argument to specify what to do with the non-logic part of the block. You can find placement information about the block in the Block report. From the **Tools** menu in the designer software, choose **Reports > Block > Interface** to display the report that shows the location of the blocks.

The -up, -down, -left, and -right arguments enable you to specify how to move the block from its original placement. You cannot rotate the block, but the relative placement of macros within the block will be preserved and the placement will be locked. However, routing will be lost. You can either use the ChipPlanner tool or run a Block report to determine the location of the block.

The -non_logic argument enables you to move a block that includes non-logic instances, such as RAM or I/Os that are difficult to move. Once you have moved a part of a block, you can unplace the remaining parts of the block and then place them manually as necessary.

Note: Note: If designing for the Axcelerator family, we recommend that you move the block to the left or right by increments of 10 to match the clusters, or if your design includes RAM, we recommend that you move the block up or down by increments of 7 to match the RAM clusters. For IGLOO, Fusion, and ProASIC3 families, we recommend that you move the block left or right by increments of 16 to match the RAM clusters and the spine columns. If your block is driven by a quadrant clock, be sure not to move the macros driven by this clock outside of the quadrant.

## Exceptions

- You must import this PDC command as a source file, not as an auxiliary file.
- You must use this PDC command if you want to preserve the relative placement and routing (if possible) of a block you are instantiating many times in your design. Only one instance will be preserved by default. To preserve other instances, you must move them using this command.

## Examples

The following example moves the entire block (instance name instA) 16 columns to the right and 16 rows up:

```
move_block -inst_name instA -right 16 -up 16 -non_logic move
```

The following example moves only the logic portion of the block and unplaces the rest by 16 columns to the right and 16 rows up.

```
move_block -inst_name instA –right 16 –up 16 –non_logic unplace
```

### See Also

set_block_options

# move_region

Moves the named region to the coordinates specified.

```
move_region region_name [x1 y1 x2 y2]+
```

## Arguments

*region_name*

Specifies the name of the region to move. This name must be unique.

*x1 y1 x2 y2*

Specifies the series of coordinate pairs representing the location in which to move the named region. These rectangles can overlap. They are given as x1 y1 x2 y2, where x1, y1 represents the lower-left corner of the rectangle and x2 y2 represents the upper-right corner. You must specify at least one set of coordinates.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- None

## Examples

This example moves the region named RecRegion to a new region which is made up of two rectangular areas:

```
move_region RecRegion 0 40 3 42 0 77 7 79
```

### See Also

Move region

PDC Syntax Conventions

PDC Naming Conventions

# reserve

Reserves the named pins in the current device package.

```
reserve -pinname "list of package pins"
```

## Arguments

`-pinname "list of package pins"`

Specifies the package pin name(s) to reserve. You can reserve one or more pins.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- None.

## Examples

```
reserve -pinname "F2"
reserve -pinname "F2 B4 B3"
reserve -pinname "124 17"
```

### See Also

unreserve

PDC Syntax Conventions

PDC Naming Conventions

# reset_floorplan

Deletes all regions.

```
reset_floorplan
```

## Arguments

- None

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- None

## Examples

reset_floorplan

### See Also

PDC Syntax Conventions

PDC Naming Conventions

# reset_io

Restores all attributes of an I/O macro to its default values. Also, if the port is assigned, it will become unassigned.

```
reset_io portname -attributes value
```

## Arguments

*portname*

Specifies the port name of the I/O macro to be reset. You can use the following wildcard characters in port names:

| Wildcard | Waht It Does |
|---|---|
| \ | Interprets the next character as a non-special character |
| ? | Matches any single character |
| * | Matches any string |
| [] | Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range) |

`-attributes` *value*

Preserve or not preserve the I/O attributes during incremental flow. The following table shows the acceptable values for this argument:

| Value | Description |
|---|---|
| yes | Unassigns all of the I/O attributes and resets them to their default values. |
| no | Unassigns only the port. |

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- None

## Examples

```
reset_io a
```

Resets the I/O macro “a” to the default I/O attributes and unassigns it.

```
reset_io b_*
```

Resets all I/O macros beginning with "b_" to the default I/O attributes and unassigns them.

```
reset_io b -attributes no
```

Only unassigns port b from its location.

## See Also

Reset attributes on an I/O to default settings

`set_io`

PDC Syntax Conventions

PDC Naming Conventions

# reset_iobank

Resets an I/O bank's technology to the default technology, which is specified using the Designer software in the Device Selection Wizard.

```
reset_iobank bankname
```

## Arguments

*bankname*

Specifies the I/O bank to be reset to the default technology. For example, for ProASIC3E and Axcelerator devices, I/O banks are numbered 0-7 (bank0, bank1,.. bank7).

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- Any pins that are assigned to the specified I/O bank but are incompatible with the default technology are unassigned.

## Examples

The following example resets I/O bank 4 to the default technology:

```
reset_iobank bank4
```

### See Also

Reset an I/O bank to the default settings

set_iobank

PDC Syntax Conventions

PDC Naming Conventions

# reset_net_critical

Resets the critical value to its default. Net criticality can vary from 1 to 10, with 1 being the least critical and 10 being the most. The default is 5. Criticality numbers are used in timing driven place-and-route.

Increasing a net's criticality forces place-and-route to keep instances connected to the net as close as possible, at the cost of other (less critical) nets.

```
reset_net_critical [netname]+
```

## Arguments

*netname*

Specifies the name of the net to be reset to the default critical value. You must specify at least one net name. You can use the following wildcard characters in net names:

| Wildcard | What It Does |
|---|---|
| \ | Interprets the next character as a non-special character |
| ? | Matches any single character |
| * | Matches any string |
| [] | Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range) |

## Supported Families

Axcelerator

## Exceptions

- None

## Examples

This example resets the net preset_a:

```
reset_net_critical preset_A
```

### See Also

Reset net's criticality to default level

set_net_critical

PDC Syntax Conventions

# set_block_options

Overrides the compile option for placement or routing conflicts for an instance of a Designer block.

```
set_block_options -inst_name instance_name -placement_conflicts value -routing_conflicts
value
```

## Arguments

`-inst_name instance_name`

Specifies the name of the instance of the block. If you do not know the name of the instance, run a compile report or look at the names shown in the Block tab of the MultiView Navigator Hierarchy view.

`-placement_conflicts value`

Specifies what to do when the designer software encounters a placement conflict. The following table shows the acceptable values for this argument:

| Value | Description |
|---|---|
| error | Compile errors out if any instance from a Designer block becomes unplaced or its routing is deleted. This is the default compile option. |
| resolve | If some instances get unplaced for any reason, the non-conflicting elements remaining are also unplaced. Basically, if there are any conflicts, nothing from the block is kept. |
| keep | If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved but not locked. Therefore, the placer can move them into another location if necessary. |
| lock | If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved and locked. |
| discard | Discards any placement from the block, even if there are no conflicts. |

`-routing_conflicts value`

Specifies what to do when the designer software encounters a routing conflict. The following table shows the acceptable values for this argument:

| Value | Description |
|---|---|
| error | Compile errors out if any route in any preserved net from a Designer block is deleted. |
| resolve | If a route is removed from a net for any reason, the routing for the non-conflicting nets is also deleted. Basically, if there are any conflicts, no routes from the block are kept. |

| Value | Description |
|---|---|
| keep | If a route is removed from a net for any reason, the routing for the non-conflicting nets is kept unlocked. Therefore, the router can re-route these nets. |
| lock | If routing is removed from a net for any reason, the routing for the non-conflicting nets is kept as locked, and the router will not change them. This is the default compile option. |
| discard | Discards any routing from the block, even if there are no conflicts. |

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Description

This command enables you to override the compile option for placement or routing conflicts for an instance of a block.

## Exceptions

- You must import this PDC command as a source file, not as an auxiliary file.
- If placement is discarded, the routing is automatically discarded too.

## Examples

This example makes the designer software display an error if any instance from a block becomes unplaced or the routing is deleted:

```
set_block_options -inst_name instA -placement_conflicts ERROR -routing_conflicts ERROR
```

### See Also

move_block

# set_io (IGLOOe, Fusion, ProASIC3L, and ProASIC3E)

Sets the attributes of an I/O for IGLOOe, Fusion, ProASIC3L, and ProASIC3E devices. You can use the set_io command to assign an I/O technology, the I/O attributes, place, or lock the I/O at a given pin location.

```
set_io portname [-pinname value][-fixed value][-iostd value][-out_drive value][-slew value][-
res_pull value][-schmitt_trigger value] [-in_delay value] [-skew value][-out_load value][-
register value]
```

## Arguments

Specifies the portname of the I/O macro to set.

`-pinname value`

Assigns the I/O macro to the specified pin.

`-fixed value`

Locks or unlocks the location of this I/O. Locked pins are not moved during layout. Therefore, locking this I/O ensures that the specified pin location is used during place-and-route. If this I/O is not currently assigned, then this argument has no effect. The following table shows the acceptable values for the -fixed attribute:

| Value | Description |
|---|---|
| yes | The location of this I/O is locked |
| no | The location of this I/O is unlocked |

`-iostd value`

Sets the I/O standard for this macro. Choosing a standard allows the software to set other attributes such as the slew rate and output loading. If the voltage standard used with the I/O is not compatible with other I/Os in the I/O bank, then assigning an I/O standard to a port will invalidate its location and automatically unassign the I/O. The following table shows the acceptable values for the -iostd attribute for IGLOOe, Fusion, ProASIC3L, and ProASIC3E devices:

| Value | Description |
|---|---|
| LVTTL | (Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer. |
| LVCMOS33 | (Low-Voltage CMOS for 3.3 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 3.3 V applications. |
| LVCMOS25 | (Low-Voltage CMOS for 2.5 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 2.5 V applications. |
| LVCMOS25_50 | (Low-Voltage CMOS for 2.5 and 5.0 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 2.5 V and 5.0V applications. |

| Value | Description |
|---|---|
| LVCMOS18 | (Low-Voltage CMOS for 1.8 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.8 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer. |
| LVCMOS15 | (Low-Voltage CMOS for 1.5 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.5 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer. |
| LVCMOS12 | (Low-Voltage CMOS for 1.2 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.2 V applications. This I/O standard is supported only in ProASIC3L and the IGLOO family of devices. |
| LVDS | A moderate-speed differential signaling system, in which the transmitter generates two different voltages which are compared at the receiver. It requires that one data bit be carried through two signal lines; therefore, you need two pins per input or output. It also requires an external resistor termination. The voltage swing between these two signal lines is approximately 350mV (millivolts). |
| LVPECL | PECL is another differential I/O standard. It requires that one data bit is carried through two signal lines; therefore, two pins are needed per input or output. It also requires an external resistor termination. The voltage swing between these two signal lines is approximately 850mV. When the power supply is +3.3 V, it is commonly referred to as low-voltage PECL (LVPECL). |
| PCI | (Peripheral Component Interface) Specifies support for both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5V-compliant for most families, excluding ProASIC3 families. |
| PCIX | (Peripheral Component Interface Extended) An enhanced version of the PCI specification that can support higher average bandwidth; it increases the speed that data can move within a computer from 66 MHz to 133 MHz. PCI-X is backward-compatible, which means that devices can operate at conventional PCI frequencies (33 MHz and 66 MHz). PCI-X is also more fault tolerant than PCI. |

| Value | Description |
|---|---|
| HSTLI | (High-Speed Transceiver Logic) A general-purpose, high-speed 1.5 V bus standard (EIA/JESD 8-6). It has four classes, of which Actel supports Class I and II for IGLOOe and ProASIC3E devices. It requires a differential amplifier input buffer and a push-pull output buffer. |
| HSTLII | (High-Speed Transceiver Logic) A general-purpose, high-speed 1.5 V bus standard (EIA/JESD 8-6). It has four classes, of which Actel supports Class I and II for IGLOOe and ProASIC3E devices. It requires a differential amplifier input buffer and a push-pull output buffer. |
| SSTL3I | (Stub Series Terminated Logic for 3.3 V) A general-purpose 3.3 V memory bus standard (JESD8-8). It has two classes, of which Actel supports both. It requires a differential amplifier input buffer and a push-pull output buffer. |
| SSTL3II | See SSTL3I above. |
| SSTL2I | (Stub Series Terminated Logic for 2.5 V) A general-purpose 2.5 V memory bus standard (JESD8-9). It has two classes, of which Actel supports both. It requires a differential amplifier input buffer and a push-pull output buffer. |
| SSTL2II | See SSTL2I above. |
| GTL25 | A low-power standard (JESD 8.3) for electrical signals used in CMOS circuits that allows for low electromagnetic interference at high speeds of transfer. It has a voltage swing between 0.4 volts and 1.2 volts, and typically operates at speeds of between 20 and 40MHz. The VCCI must be connected to 2.5 volts. |
| GTL33 | Same as GTL 2.5 V, except the VCCI must be connected to 3.3 volts. |
| GTLP25 | (Gunning Transceiver Logic Plus) A high-speed bus standard (JESD8.3). It requires a differential amplifier input buffer and an open-drain output buffer. Even though output is open-drain, IGLOO (excluding the IGLOO device), ProASIC3, Axcelerator families, support still needs the VCCI to be connected to 2.5 V or 3.3 V. |
| GTLP33 | See GTLP33 above. |

```
-out_drive value
```

Sets the strength of the output buffer to 2, 4, 6, 8, 12, 16, or 24 in mA, weakest to strongest. The list of I/O standards for which you can change the output drive and the list of values you can assign for each I/O standard is family-specific. Not all I/O standards have a selectable output drive strength. Also, each I/O standard has a different range of legal output drive strength values. The values you can choose from depend on which I/O standard you have specified for this command. See the "Slew and Out_drive Settings" table under "Exceptions" in this topic for possible values. Also, refer to the ProASIC3E and IGLOOe datasheets for more information. The following table shows the acceptable values for the -out_drive attribute:

| Value | Description |
|---|---|
| 2 | Sets the output drive strength to 2mA |
| 4 | Sets the output drive strength to 4mA |
| 6 | Sets the output drive strength to 6mA |
| 8 | Sets the output drive strength to 8mA |
| 12 | Sets the output drive strength to 12mA |
| 16 | Sets the output drive strength to 16mA |
| 24 | Sets the output drive strength to 24mA |

`-slew value`

Sets the output slew rate. Slew control affects only the falling edges for some families. For ProASIC3, IGLOO, Fusion, and Axcelerator families, slew control affects both rising and falling edges. Whether you can use the slew attribute depends on which I/O standard you have specified for this command.

Not all I/O standards have a selectable slew. For ProASIC3 devices, this attribute is only available for LVTTL, LVCMOS33, LVCMOS25_50, LVCMOS18, LVCMOS15, and PCIX outputs. For any of the I/O standards, the slew can be either high or low. The default is high. See the " Slew and Out_drive Settings" table under "Exceptions" in this topic. Also, refer to the ProASIC3E and IGLOOe datasheets for more information. The following table shows the acceptable values for the -slew attribute:

| Value | Description |
|---|---|
| high | Sets the I/O slew to high |
| low | Sets the I/O slew to low |

`-res_pull value`

Allows you to include a weak resistor for either pull-up or pull-down of the input buffer. Not all I/O standards have a selectable resistor pull option. The following table shows the acceptable values for the -res_pull attribute:

| Value | Description |
|---|---|
| up | Includes a weak resistor for pull-up of the input buffer |

| Value | Description |
|---|---|
| down | Includes a weak resistor for pull-down of the input buffer |
| none | Does not include a weak resistor |

`-schmitt_trigger` *value*

Specifies whether this I/O has an input schmitt trigger. The schmitt trigger introduces hysteresis on the I/O input. This allows very slow moving or noisy input signals to be used with the part without false or multiple I/O transitions taking place in the I/O. The following table shows the acceptable values for the -schmitt_trigger attribute:

| Value | Description |
|---|---|
| on | Turns the schmitt trigger on |
| off | Turns the schmitt trigger off |

`-in_delay` *value*

Specifies whether this I/O has an input delay. You can specify an input delay between 0 and 7. The input delay is not a delay value but rather a selection from 0 to 7. The actual value is a function of the operating conditions and is automatically computed by the delay extractor when a timing report is generated. The following table shows the acceptable values for the -in_delay attribute:

| Value | Description |
|---|---|
| off | This I/O does not have an input delay |
| 0 | Sets the input delay to 0 |
| 1 | Sets the input delay to 1 |
| 2 | Sets the input delay to 2 |
| 3 | Sets the input delay to 3 |
| 4 | Sets the input delay to 4 |
| 5 | Sets the input delay to 5 |
| 6 | Sets the input delay to 6 |
| 7 | Sets the input delay to 7 |

`-skew` *value*

Specifies whether there is a fixed additional delay between the enable/disable time for a tristatable I/O. (A tristatable I/O is an I/O with three output states: high, low, and high impedance.) The following table shows the acceptable values for the -skew attribute:

| Value | Description |
|---|---|
| on | Yes, there is a fixed additional delay |
| off | No, there is not a fixed additional delay |

`-out_load` *`value`*

Determines what Timer will use as the loading on the output pin. This attribute applies only to outputs. You can enter a capacitive load as an integral number of picofarads (pF). Specify an integer between 0 and 1023 pF.

`-register` *`value`*

Specifies whether the register will be combined into the I/O. If this option is yes, the combiner combines the register into the I/O module if possible. This option overrides the default setting in the Compile options. I/O registers are off by default. The following table shows the acceptable values for the -register attribute:

| Value | Description |
|---|---|
| yes | Register combining is allowed on this I/O |
| no | Register combining is not allowed on this I/O |

See I/O Register Combining Rules for more details.

## Supported Families

IGLOOe, ProASIC3L, ProASIC3E, SmartFusion and Fusion devices

## Exceptions

- If an argument is not specified, the value is not changed, as long as it is consistent with other settings. If setting an attribute invalidates the I/Os location, then the I/O is unassigned.
- You can specify an out_drive strength and slew rate only for certain I/O standards per family. Not all I/O standards have a selectable output drive strength or slew. The following table shows I/O standards for which you can specify a slew and out_drive setting:

| I/O Standard | Output | | | | | | | Slew |
|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 12 | 16 | 24 | |
| LVTTL | X | X | X | X | X | X | X | High Low |
| LVCMOS33 | X | X | X | X | X | X | X | High Low |

| I/O Standard | Output | | | | | | | Slew |
|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 12 | 16 | 24 | |
| LVCMOS25 | X | X | X | X | X | X | X | High Low |
| LVCMOS25_50 | X | X | X | X | X | X | X | High Low |
| LVCMOS18 | x | x | x | x | - | - | - | High Low |
| LVCMOS15 | x | x | - | - | - | - | - | High Low |
| LVCMOS12 | x | - | - | - | - | - | - | High Low |
| PCIX | - | - | - | - | - | - | - | High Low |

Note: Note: AGL030 and AGL015 do not support 2mA. They only support 1mA.

## Examples

```
set_io IO_in\[2\] -iostd LVPECL \
-slew low \
-skew off \
-schmitt_trigger off \
-in_delay 0 \
-register no \
-pinname 366 \
-fixed no
```

### See Also

Assign I/O to pin

reset_io

PDC Syntax Conventions

PDC Naming Conventions

# set_io (IGLOO PLUS)

Sets the attributes of an I/O for IGLOO PLUS devices. You can use the `set_io` command to assign an I/O technology, the I/O attributes, place, or lock the I/O at a given pin location.

```
set_io portname [-pinname value][-fixed value][-iostd value][-out_drive value][-slew value][-
res_pull value][-schmitt_trigger value] -out_load value][-register value] [-hold_state
value]
```

**Document the exported set_io option '-DIRECTION' in the the OLH? (mark these topics with red asterisk in the toc)**

## Arguments

*portname*

Specifies the portname of the I/O macro to set.

`-pinname` *value*

Assigns the I/O macro to the specified pin.

`-fixed` *value*

Locks or unlocks the location of this I/O. Locked pins are not moved during layout. Therefore, locking this I/O ensures that the specified pin location is used during place-and-route. If this I/O is not currently assigned, then this argument has no effect. The following table shows the acceptable values for the -fixed attribute:

| Value | Description |
|---|---|
| yes | The location of this I/O is locked |
| no | The location of this I/O is unlocked |

`-iostd` *value*

Sets the I/O standard for this macro. Choosing a standard allows the software to set other attributes such as the slew rate and output loading. If the voltage standard used with the I/O is not compatible with other I/Os in the I/O bank, then assigning an I/O standard to a port will invalidate its location and automatically unassign the I/O. The following table shows the acceptable values for the -iostd attribute for IGLOO PLUS devices:

| Value | Description |
|---|---|
| LVTTL | (Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer. |
| LVCMOS33 | (Low-Voltage CMOS for 3.3 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 3.3 V applications. |
| LVCMOS25 | (Low-Voltage CMOS for 2.5 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 2.5 V applications. |

| Value | Description |
|---|---|
| LVCMOS18 | (Low-Voltage CMOS for 1.8 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.8 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer. |
| LVCMOS15 | (Low-Voltage CMOS for 1.5 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.5 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer. |
| LVCMOS12 | (Low-Voltage CMOS for 1.2 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.2 V applications. This I/O standard is supported only in ProASIC3L and the IGLOO family of devices. |

`-out_drive value`

Sets the strength of the output buffer to 2, 4, 6, 8, 12, or 16 in mA, weakest to strongest. The list of I/O standards for which you can change the output drive and the list of values you can assign for each I/O standard is family-specific. Not all I/O standards have a selectable output drive strength. Also, each I/O standard has a different range of legal output drive strength values. The values you can choose from depend on which I/O standard you have specified for this command. See the "Slew and Out_drive Settings" table under "Exceptions" in this topic for possible values. Also, refer to the IGLOO PLUS datasheet for more information. The following table shows the acceptable values for the -out_drive attribute:

| Value | DescriptiT |
|---|---|
| 2 | Sets the output drive strength to 2mA |
| 4 | Sets the output drive strength to 4mA |
| 6 | Sets the output drive strength to 6mA |
| 8 | Sets the output drive strength to 8mA |
| 12 | Sets the output drive strength to 12mA |
| 16 | Sets the output drive strength to 16mA |

`-slew value`

Sets the output slew rate. Slew control affects only the falling edges for some families. For ProASIC3, IGLOO, Fusion, and Axcelerator families, slew control affects both rising and falling edges. Not all I/O standards have a selectable slew. Whether you can use the slew attribute depends on which I/O standard you have specified for this command.

For ProASIC3 devices, this attribute is only available for LVTTL, LVCMOS33, LVCMOS25_50, LVCMOS18, LVCMOS15, and PCIX outputs. For any of the I/O standards, the slew can be either high or low. The default is

high. See the "Slew and Out_drive Settings" table under "Exceptions" in this topic. Also, refer to the IGLOO PLUS datasheet for more information. The following table shows the acceptable values for the -slew attribute:

| Value | Description |
|---|---|
| high | Sets the I/O slew to high |
| low | Sets the I/O slew to low |

`-res_pull` *value*

Allows you to include a weak resistor for either pull-up or pull-down of the input buffer. Not all I/O standards have a selectable resistor pull option. The following table shows the acceptable values for the -res_pull attribute:

| Value | Description |
|---|---|
| up | Includes a weak resistor for pull-up of the input buffer |
| down | Includes a weak resistor for pull-down of the input buffer |
| none | Does not include a weak resistor |

`-schmitt_trigger` *value*

Specifies whether this I/O has an input schmitt trigger. The schmitt trigger introduces hysteresis on the I/O input. This allows very slow moving or noisy input signals to be used with the part without false or multiple I/O transitions taking place in the I/O. The following table shows the acceptable values for the -schmitt_trigger attribute:

| Value | Description |
|---|---|
| on | Turns the schmitt trigger on |
| off | Turns the schmitt trigger off |

`-out_load` *value*

Determines what Timer will use as the loading on the output pin. This attribute applies only to outputs. You can enter a capacitive load as an integral number of picofarads (*pF*). Specify an integer between 0 and 1023*pF*. The default is 5*pF* for all IGLOO devices.

`-register` *value*

Specifies whether the register will be combined into the I/O. If this option is yes, the combiner combines the register into the I/O module if possible. This option overrides the default setting in the Compile options. I/O registers are off by default. The following table shows the acceptable values for the -register attribute:

| Value | Description |
|---|---|
| yes | Register combining is allowed on this I/O |
| no | Register combining is not allowed on this I/O |

See I/O Register Combining Rules for more details.

`-hold_state` *value*

Preserves the previous state of the I/O. By default, all the I/Os become tristated when the device goes into Flash*Freeze mode. (A tristatable I/O is an I/O with three output states: high, low, and high impedance.) You can override this default using the hold_state attribute. When you set the hold_state to True, the I/O remains in the same state in which it was functioning before the device went into Flash*Freeze mode. The following table shows the acceptable values for the -skew attribute:

| Value | Description |
|---|---|
| on | Preserves the previous state of the I/O |
| off | Does not preserve the previous state of the I/O |

## Supported Families

IGLOO PLUS

## Exceptions

- If an argument is not specified, the value is not changed, as long as it is consistent with other settings. If setting an attribute invalidates the I/Os location, then the I/O is unassigned.
- You can specify an out_drive strength and slew rate only for certain I/O standards per family. Not all I/O standards have a selectable output drive strength or slew. The following table shows I/O standards for which you can specify a slew and out_drive setting:

| I/O Standard | Output | | | | | | Slew |
|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 12 | 16 | |
| LVTTL | X | X | X | X | X | X | High Low |
| LVCMOS33 | X | X | X | X | X | X | High Low |
| LVCMOS25 | X | X | X | X | X | X | High Low |
| LVCMOS18 | X | X | X | X | - | - | High Low |
| LVCMOS15 | X | X | - | - | - | - | High Low |
| LVCMOS12 | x | - | - | - | - | - | High Low |
| PCI | - | - | - | - | - | - | High Low |

| I/O Standard | Output | | | | | | Slew |
|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 12 | 16 | |
| PCIX | - | - | - | - | - | - | High Low |

## Examples

```
set_io IO_in\[2\] -iostd LVPECL \
   -slew low \
-schmitt_trigger off \
-register no \
-pinname 366 \
-fixed no
```

### See Also

Assign I/O to pin

reset_io

PDC Syntax Conventions

PDC Naming Conventions

# set_io (IGLOO and ProASIC3)

Sets the attributes of an I/O for IGLOO and ProASIC3 devices. You can use the `set_io` command to assign an I/O technology, the I/O attributes, assign, or lock the I/O at a given pin location.

```
set_io portname [-pinname value][-fixed value][-iostd value][-out_drive value][-slew value][-
res_pull value][-out_load value][-skew value][-register value]
```

## Arguments

*portname*

Specifies the portname of the I/O macro to set.

`-pinname` *value*

Assigns the I/O macro to the specified pin.

`-fixed` *value*

Locks or unlocks the location of this I/O. Locked pins are not moved during layout. Therefore, locking this I/O ensures that the specified pin location is used during place-and-route. If this I/O is not currently assigned, then this argument has no effect. The following table shows the acceptable values for the -fixed attribute:

| Value | Description |
|---|---|
| yes | The location of this I/O is locked |
| no | The location of this I/O is unlocked |

`-iostd` *value*

Sets the I/O standard for this macro. Choosing a standard allows the software to set other attributes such as the slew rate and output loading. If the voltage standard used with the I/O is not compatible with other I/Os in the I/O bank, then assigning an I/O standard to a port will invalidate its location and automatically unassign the I/O. The following table shows the acceptable values for the -iostd attribute for ProASIC3 devices:

| Value | Description |
|---|---|
| LVTTL | (Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer. |
| LVCMOS33 | (Low-Voltage CMOS for 3.3 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 3.3 V applications. |
| LVCMOS25_50 | (Low-Voltage CMOS for 2.5 and 5.0 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 2.5 V and 5.0V applications. |

| Value | Description |
|---|---|
| LVCMOS18 | (Low-Voltage CMOS for 1.8 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.8 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer. |
| LVCMOS15 | (Low-Voltage CMOS for 1.5 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.5 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer. |
| LVCMOS12 | (Low-Voltage CMOS for 1.2 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.2 V applications. This I/O standard is supported only in ProASIC3L (A3PE3000L) and the IGLOO family of devices. |
| LVDS | A moderate-speed differential signalling system, in which the transmitter generates two different voltages which are compared at the receiver. It requires that one data bit be carried through two signal lines; therefore, you need two pins per input or output. It also requires an external resistor termination. The voltage swing between these two signal lines is approximately 350mV (millivolts). |
| LVPECL | PECL is another differential I/O standard. It requires that one data bit is carried through two signal lines; therefore, two pins are needed per input or output. It also requires an external resistor termination. The voltage swing between these two signal lines is approximately 850mV. When the power supply is +3.3 V, it is commonly referred to as low-voltage PECL (LVPECL). |
| PCI | (Peripheral Component Interface) Specifies support for both 33 |

| Value | Description |
|---|---|
| | MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5V-compliant for most families, excluding ProASIC3 families. |
| PCIX | (Peripheral Component Interface Extended) An enhanced version of the PCI specification that can support higher average bandwidth; it increases the speed that data can move within a computer from 66 MHz to 133 MHz. PCIX is backward-compatible, which means that devices can operate at conventional PCI frequencies (33 MHz and 66 MHz). PCIX is also more fault tolerant than PCI. |

`-out_drive` *`value`*

Sets the strength of the output buffer to 2, 4, 6, 8, 12, or 16 in mA, weakest to strongest. The list of I/O standards for which you can change the output drive and the list of values you can assign for each I/O standard is family-specific. Not all I/O standards have a selectable output drive strength. Also, each I/O standard has a different range of legal output drive strength values. The values you can choose from depend on which I/O standard you have specified for this command. See the "Slew and Out_drive Settings" table under "Exceptions" in this topic for possible values. Also, refer to the IGLOO and ProASIC3 datasheets for more information.

Note: Note: Dies AGL015 and AGL030 only support the default output drive strength of 1mA. You must explicitly set the -output_drive attribute using either a PDC file or changing this setting in the I/O Attribute Editor of MultiView Navigator.

The following table shows the acceptable values for the -out_drive attribute:

| Value | Description |
|---|---|
| 1 | Sets the output drive strength to 1mA (default) |
| 2 | Sets the output drive strength to 2mA |
| 4 | Sets the output drive strength to 4mA |
| 6 | Sets the output drive strength to 6mA |
| 8 | Sets the output drive strength to 8mA |
| 12 | Sets the output drive strength to 12mA |
| 16 | Sets the output drive strength to 16mA |

`-slew value`

Sets the output slew rate. Slew control affects only the falling edges for some families. For ProASIC3, IGLOO, Fusion, and Axcelerator families, slew control affects both rising and falling edges. Whether you can use the slew attribute depends on which I/O standard you have specified for this command.

For ProASIC3 devices, this attribute is only available for LVTTL, LVCMOS33, LVCMOS25_50, LVCMOS18, LVCMOS15, and PCIX outputs. For any of the I/O standards, the slew can be either high or low. The default is high. See the "Slew and Out_drive Settings" table under "Exceptions" in this topic. Also, refer to the IGLOO and ProASIC3 datasheets for more information. The following table shows the acceptable values for the -slew attribute:

| Value | Description |
|---|---|
| high | Sets the I/O slew to high |
| low | Sets the I/O slew to low |

`-res_pull value`

Allows you to include a weak resistor for either pull-up or pull-down of the input buffer. Not all I/O standards have a selectable resistor pull option. The following table shows the acceptable values for the -res_pull attribute:

| Value | Description |
|---|---|
| up | Includes a weak resistor for pull-up of the input buffer |
| down | Includes a weak resistor for pull-down of the input buffer |
| none | Does not include a weak resistor |

`-out_load value`

Determines what Timer will use as the loading on the output pin. This attribute applies only to outputs. You can enter a capacitive load as an integral number of picofarads (*pF*). Specify an integer between 0 and 1023*pF*.

`-skew value`

Specifies whether there is a fixed additional delay between the enable/disable time for a tristatable I/O. (A tristatable I/O is an I/O with three output states: high, low, and high impedance.) The following table shows the acceptable values for the -skew attribute:

| Value | Description |
|---|---|
| on | Yes, there is a fixed additional delay |
| off | No, there is not a fixed additional delay |

Note: Note: There is no skew support for AGL030 and AGL015 devices.

`-register value`

Specifies whether the register will be combined into the I/O. If this option is yes, the combiner combines the register into the I/O module if possible. This option overrides the default setting in the Compile options. I/O registers are off by default. The following table shows the acceptable values for the -register attribute:

| Value | Description |
|---|---|
| yes | Register combining is allowed on this I/O |
| no | Register combining is not allowed on this I/O |

See I/O Register Combining Rules for more details.

## Supported Families

IGLOO (excluding IGLOOe) and ProASIC3 (excluding ProASIC3L and ProASIC3E)

## Exceptions

- If an argument is not specified, the value is not changed, as long as it is consistent with other settings. If setting an attribute invalidates the I/Os location, then the I/O is unplaced.
- You can specify an out_drive strength and slew rate only for certain I/O standards per family. Not all I/O standards have a selectable output drive strength or slew. The following table shows I/O standards for which you can specify a slew and out_drive setting:

| I/O Standard | Output | | | | | | Slew |
|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 12 | 16 | |
| LVTTL | X | X | X | X | X | X | High Low |
| LVCMOS33 | X | X | X | X | X | X | High Low |
| LVCMOS25_50 | X | X | X | X | X | - | High Low |
| LVCMOS18 | X | X | X | X | - | - | High Low |
| LVCMOS15 | x | x | - | - | - | - | High Low |
| LVCMOS12 | x | - | - | - | - | - | High Low |
| PCIX | - | - | - | - | - | - | High Low |

## Examples

```
set_io IO_in\[2\] -iostd LVPECL -register no -pinname 366 -fixed no
```

### See Also

Assign I/O to pin

`reset_io`

PDC Syntax Conventions

PDC Naming Conventions

# set_io (Axcelerator)

Sets the attributes of an I/O for Axcelerator families. You can use the `set_io` command to assign an I/O technology, place, or lock the I/O at a given pin location.

Note: Note: To enter an argument on a separate line, you must enter a backslash (\) character at the end of the preceding line of the command.

```
set_io portname[ -pinname value][-fixed value][ -iostd value][-out_drive value][ -slew
value][-res_pull value][-in_delay value][-out_load value][-register value][-clamp_diode
value]
```

## Arguments

*portname*

Specifies the portname of the I/O macro to set.

`-pinname` *value*

Assigns the I/O macro to the specified pin.

`-fixed` *value*

Locks or unlocks the location of this I/O. Locked pins are not moved during layout. Therefore, locking this I/O ensures that the specified pin location is used during place-and-route. If this I/O is not currently assigned, then this argument has no effect. The following table shows the acceptable values this attribute:

| Value | Description |
|---|---|
| yes | The location of this I/O is locked |
| no | The location of this I/O is unlocked |

`-iostd` *value*

Sets the I/O standard for this macro. Choosing a standard allows the software to set other attributes such as the slew rate and output loading. If the voltage standard used with the I/O is not compatible with other I/Os in the I/O bank, then assigning an I/O standard to a port will invalidate its location and automatically unassign the I/O. The following table shows the acceptable values for Axcelerator devices:

| Value | Description |
|---|---|
| LVTTL | (Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer. |
| LVCMOS25 | (Low-Voltage CMOS for 2.5 Volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 2.5 V applications. |
| LVCMOS18 | (Low-Voltage CMOS for 1.8 Volts) An extension of the |

| Value | Description |
|---|---|
| | LVCMOS standard (JESD 8-5) used for general-purpose 1.8 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer. |
| LVCMOS15 | (Low-Voltage CMOS for 1.5 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.5 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer. |
| LVDS | A moderate-speed differential signalling system, in which the transmitter generates two different voltages which are compared at the receiver. It requires that one data bit be carried through two signal lines; therefore, you need two pins per input or output. It also requires an external resistor termination. The voltage swing between these two signal lines is approximately 350mV (millivolts). Axcelerator devices contain dedicated circuitry supporting a high-speed LVDS standard that has its own user specification. |
| LVPECL | PECL is another differential I/O standard. It requires that one data bit is carried through two signal lines; therefore, two pins are needed per input or output. It also requires an external resistor termination. The voltage swing between these two signal lines is approximately 850mV. When the power supply is +3.3 V, it is commonly referred to as low-voltage PECL (LVPECL). |
| PCI | (Peripheral Component Interface) Specifies support for both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5V-compliant for most families, excluding ProASIC3 families. |
| PCIX | (Peripheral Component Interface Extended) An enhanced version of the PCI specification that can support higher average bandwidth; it increases the speed that data can move within a computer from 66 MHz to 133 MHz. PCI-X is backwards compatible, which means that devices can operate at conventional PCI frequencies (33 MHz and 66 MHz). PCI-X is also more fault tolerant than PCI. |

| Value | Description |
|---|---|
| HSTLI | (High-Speed Transceiver Logic) A general-purpose, high-speed 1.5 V bus standard (EIA/JESD 8-6). It has four classes, of which Actel supports Class I for Axcelerator devices. It requires a differential amplifier input buffer and a push-pull output buffer. |
| SSTL3I | (Stub Series Terminated Logic for 3.3 V) A general-purpose 3.3 V memory bus standard (JESD8-8). It has two classes, of which Actel supports both. It requires a differential amplifier input buffer and a push-pull output buffer. |
| SSTL3II | See SSTL3I above. |
| SSTL2I | (Stub Series Terminated Logic for 2.5 V) A general-purpose 2.5 V memory bus standard (JESD8-9). It has two classes, of which Actel supports both. It requires a differential amplifier input buffer and a push-pull output buffer. |
| SSTL2II | See SSTL2I above. |
| GTLP33 | (Gunning Transceiver Logic Plus) A high-speed bus standard (JESD8.3). It requires a differential amplifier input buffer and an open-drain output buffer. Even though output is open-drain, IGLOO, ProASIC, and Axcelerator support still needs the VCCI to be connected to 2.5 V or 3.3 V. |
| GTLP25 | See GTLP33 above. |

`-out_drive` *`value`*

Sets the I/O output drive strength in mA. This argument is used only for LVTTL, PCI, and PCIX standards. The LVTTL standard supports all four strengths. For PCI, it only supports the 16 mA. For PCIX, it only supports the 12 mA. The following table shows the acceptable values for this attribute:

| Value | Description |
|---|---|
| 8 | Sets the output drive strength to 8mA |
| 12 | Sets the output drive strength to 12mA |
| 16 | Sets the output drive strength to 16mA |

| Value | Description |
|---|---|
| 24 | Sets the output drive strength to 24mA |

`-slew value`

Sets the output slew rate. This attribute is only available for LVTTL, PCI, and PCIX outputs. For LVTTL, it can either be high or low. For PCI and PCIX, it can only be set to high. The following table shows the acceptable values for this attribute:

| Value | Description |
|---|---|
| high | Sets the I/O slew to high |
| low | Sets the I/O slew to low |

`-res_pull value`

Allows you to include a weak resistor for either pull-up or pull-down of the input buffer. The following table shows the acceptable values for this attribute:

| Value | Description |
|---|---|
| up | Includes a weak resistor for pull-up of the input buffer |
| down | Includes a weak resistor for pull-down of the input buffer |
| none | Does not include a weak resistor |

`-in_delay value`

Turns the input I/O delay on or off. The value of this delay is set on a bank-wide basis either by using the set_iobank PDC command or from the I/O Banks Settings dialog box in ChipPlanner or PinEditor. Refer to the Axcelerator datasheet for more details. The following table shows the acceptable values for this attribute:

| Value | Description |
|---|---|
| on | Turns the input I/O delay on |
| off | Turns the input I/O delay off |

`-out_load value`

Determines what Timer will use as the loading on the output pin. This attribute applies only to outputs. You can enter a capacitive load as an integral number of picofarads (*pF*). The default is 35*pF*.

`-register value`

Specifies whether the register will be combined into the I/O. If this option is yes, the combiner combines the register into the I/O module if possible. This option overrides the default setting in the Compile options. I/O registers are off by default. The following table shows the acceptable values for this attribute:

| Value | Description |
|---|---|
| yes | Register combining is allowed on this I/O |
| no | Register combining is not allowed on this I/O |

`-clamp_diode` *value*

Specifies whether to add a power clamp diode to the I/O buffer. This attribute option is available to all I/O buffers with I/O technology set to LVTTL. A clamp diode provides circuit protection from voltage spikes, surges, electrostatic discharge and other over-voltage conditions. If the option is set to yes, a clamp diode to VCCI is added to the I/O buffer. This option overrides the default setting in the software. The default for this option is "no". The following table shows the acceptable values for this attribute:

| Value | Description |
|---|---|
| yes | The LVTTL I/O will be clamped to VCCI using a clamp diode |
| no | The LVTTL I/O will not have a VCCI clamp diode (compatible with LVTTL standard) |

## Supported Families

Axcelerator

## Exceptions

- If an argument is not specified, the value is not changed, as long as it is consistent with other settings. If setting an attribute invalidates the I/Os location, then the I/O is unplaced.
- When using this command in an auxiliary PDC file, the -register argument is not honored. To combine a given I/O with the register without losing your floorplan, you must open PinEditor and select the one you need to combine and rerun compile.

## Examples

```
set_io REG_RBB_OUT_15_ -iostd LVTTL -res_pull up -in_delay on -pinname J18 -fixed yes
set_io ADDOUT2 \
   -iostd PCI \
   -register yes \
   -out_drive 16 \
   -slew high \
   -out_load 10 \
   -pinname T21 \
   -fixed yes
```

### See Also

Assign I/O to pin

`reset_io`

PDC Syntax Conventions

PDC Naming Conventions

# set_iobank (IGLOOe, IGLOO PLUS, Fusion, ProASIC3L, and ProASIC3E)

Sets the input/output supply voltage (vcci) and the input reference voltage (vref) for the specified I/O bank. This command applies only to IGLOOe, Fusion, ProASIC3L (A3PE3000L only), and ProASIC3E devices.

```
set_iobank bankname [-vcci vcci_voltage] [-vref vref_voltage]
[-fixed value] [-vrefpins value]
```

## Arguments

*bankname*

Specifies the name of the bank. I/O banks are numbered 0 through N (bank0, bank1,...bankN). See the datasheet for your device to determine how many banks it has.

`-vcci` *vcci_voltage*

Sets the input/output supply voltage. You can enter one of the following values:

| Vcci Voltage | Compatible Standards |
|---|---|
| 3.3 V | LVTTL, LVCMOS 3.3, PCI 3.3, PCI-X 3.3, SSTL3 (Class I and II), GTL+ 3.3, GTL 3.3, LVPECL |
| 2.5 V | LVCMOS 2.5, LVCMOS 2.5/5.0, SSTL2 (Class I and II), GTL+2.5, GTL 2.5, LVDS |
| 1.8 V | LVCMOS 1.8 |
| 1.5 V | LVCMOS 1.5, HSTL (Class I and II) |
| 1.2 V | LVCMOS 1.2 |

Note: Note: 1.2 voltage is supported for ProASIC3L (A3PL), IGLOOe V2 only, IGLOO V2, and IGLOO PLUS.

`-vref` *vref_voltage*

Sets the input reference voltage. This option is only supported by ProASIC3E, IGLOOe and ProASIC3L(3000 die only) devices. You can enter one of the following values:

| Vref Voltage | Compatible Standards |
|---|---|
| 1.5 V | SSTL3 (Class I and II) |
| 1.25V | SSTL2 (Class I and II) |

| Vref Voltage | Compatible Standards |
|---|---|
| 1.0V | GTL+ 2.5, GTL+ 3.3 |
| 0.8V | GTL 2.5, GTL 3.3 |
| 0.75V | HSTL (Class I and Class II) |

`-fixed value`

Specifies if the I/O technologies (vcci and vccr voltage) assigned to the bank are locked. You can enter one of the following values:

| Value | Description |
|---|---|
| yes | The technologies are locked. |
| no | The technologies are not locked. |

`-vrefpins value`

Specifies if the I/O technologies (vcci and vccr voltage) assigned to the bank are locked. This option is only supported by ProASIC3E, IGLOOe and ProASIC3L(3000 die only) devices. You can enter one of the following values:

| Value | Description |
|---|---|
| default | Because the VREF pins are not locked, the I/O Bank Assigner can assign a VREF pin. |
| pinnum | The specified VREF pin(s) are locked if the -fixed option is "yes". The I/O Bank Assigner cannot remove locked VREF pins. |

Note: Note: The set_vref and set_vref_defaults PDC commands are no longer supported. You can now use the set_iobanks command to set the vref pins. If you used the set_vref and set_vref_defaults commands in an existing design, when you export the PDC commands, the Designer software replaces the old set_vref and set_vref_defaults commands with the set_iobanks command.

## Supported Families

IGLOO (IGLOOe and IGLOO PLUS only), ProASIC3 (ProASIC3L A3PE3000L die and ProASIC3E only), SmartFusion, Fusion

Note: Note: Refer to the IGLOOe and ProASIC3E datasheet on the Actel web site for details about the legal values for the vcci and vref arguments

## Exceptions

- Any pins assigned to the specified I/O bank that are incompatible with the default technology are unassigned.

## Examples

The following example assigns 3.3 V to the input/output supply voltage (vcci) and 1.5 V to the input reference voltage (vref) for I/O bank 0.

```
set_iobank bank0 -vcci 3.3 -vref 1.5
```

The following example shows that even though you can import a set_iobank command with the -vrefpins argument set to "default", the exported PDC file will show the specific default pins instead of "default."

Imported PDC file contains:

```
set_iobank bank3 -vcci 3.3 -vref 1.8 -fixed yes -vrefpins {default}
```

Exported PDC file contains:

```
set_iobank bank3 -vcci 3.3 -vref 1.8 -fixed yes -vrefpins {N3 P8 M8}
```

### See Also

Configure I/O Bank

`reset_io`
`reset_iobank`
PDC Syntax Conventions

PDC Naming Conventions

# set_iobank (IGLOO and ProASIC3)

Sets the input/output supply voltage (vcci) for the specified I/O bank. This command applies only to IGLOO and ProASIC3 devices. For information about IGLOOe, IGLOO PLUS, Fusion, ProASIC3L, and ProASIC3E devices, see set_iobank (IGLOOe, IGLOO PLUS, Fusion, ProASIC3L, and ProASIC3E).

```
set_iobank bankname [-vcci vcci_voltage] [-fixed value] [-vrefpins value]
```

## Arguments

*bankname*

Specifies the name of the bank. I/O banks are numbered 0 through N (bank0, bank1,...bankN). See the datasheet for your device to determine how many banks it has.

`-vcci` *vcci_voltage*

Sets the input/output supply voltage. You can enter one of the following values:

| Vcci Voltage | Compatible Standards |
|---|---|
| 3.3 V | LVTTL, LVCMOS 3.3, PCI 3.3, PCI-X 3.3 |
| 2.5 V | LVCMOS 2.5/5.0, LVDS, LVPECL |
| 1.8 V | LVCMOS 1.8 |
| 1.5 V | LVCMOS 1.5 |
| 1.2 V | LVCMOS 1.2 |

Note: Note: 1.2 voltage is supported for ProASIC3 (A3PL), IGLOOe V2 only, IGLOO V2, and IGLOO PLUS.

`-fixed` *value*

Specifies if the I/O technologies (vcci and vccr voltage) assigned to the bank are locked. You can enter one of the following values:

| Value | Description |
|---|---|
| yes | The technologies are locked. |
| no | The technologies are not locked. |

`-vrefpins` *value*

Specifies if the I/O technologies (vcci and vccr voltage) assigned to the bank are locked. This option is only supported by ProASIC3E, IGLOOe and ProASIC3L(3000 die only) devices. You can enter one of the following values:

| Value | Description |
|---|---|

| Value | Description |
|---|---|
| default | Because the VREF pins are not locked, the I/O Bank Assigner can assign a VREF pin. |
| pinnum | The VREF pin(s) that are locked when the -fixed option is "yes". The I/O Bank Assigner cannot remove locked VREF pins. |

Note: Note: The set_vref and set_vref_defaults PDC commands are no longer supported. You can now use the set_iobanks command to set the vref pins. If you used the set_vref and set_vref_defaults commands in an existing design, when you export the PDC commands, the Designer software replaces the old set_vref and set_vref_defaults commands with the set_iobanks command.

## Supported Families

IGLOO and ProASIC3

Note: Note: Refer to the ProASIC3 datasheets on the Actel web site for details about the legal values for the vcci argument.

## Exceptions

- Any pins assigned to the specified I/O bank that are incompatible with the default technology are unassigned.

## Examples

The following example assigns 3.3 V to the input/output supply voltage (vcci) for I/O bank 0.

```
set_iobank bank0 -vcci 3.3
```

The following example shows that even though you can import a set_iobank command with the -vrefpins argument set to "default", the exported PDC file will show the specific default pins instead of "default."

Imported PDC file contains:

```
set_iobank bank3 -vcci 3.3 -fixed yes -vrefpins {default}
```

Exported PDC file contains:

```
set_iobank bank3 -vcci 3.3 -fixed yes -vrefpins {N3 P8 M8}
```

### See Also

Configure I/O Bank

reset_io
reset_iobank
PDC Syntax Conventions

PDC Naming Conventions

# set_iobank (Axcelerator)

Sets the input/output supply voltage (vcci) and the input reference voltage (vref) for the specified I/O bank. It also sets the input delay value and enables or disables the low-power mode for input and output buffers. This command applies only to Axcelerator families.

```
set_iobank bankname [-vcci vcci_voltage] [-vref vref_voltage] [-inputdelay bits_setting] [-
lpinput value] [-lpoutput value] [-fixed value] [-vrefpins value]
```

## Arguments

*bankname*

Specifies the name of the bank. Axcelerator devices have eight I/O banks, which are numbered 0 through 7. So, their default bank names are bank0, bank1,...bank7.

`-vcci` *vcci_voltage*

Sets the input/output supply voltage. You can enter one of the following values:

| Vcci Voltage | Compatible Standards |
|---|---|
| 3.3 V | LVTTL, PCI 3.3, PCIX 3.3, SSTL3 (Class I and II), GTL+ 3.3, LVPECL |
| 2.5 V | LVCMOS 2.5, SSTL2 (Class I and II), GTL+2.5, LVDS |
| 1.8 V | LVCMOS 1.8 |
| 1.5 V | LVCMOS 1.5, HSTL (Class I) |

`-vref` *vref_voltage*

Sets the input reference voltage. This option is only supported by ProASIC3E, IGLOOe and ProASIC3L(3000 die only) devices. You can enter one of the following values:

| Vref Voltage | Compatible Standards |
|---|---|
| 1.5 V | SSTL3 (Class I and II) |
| 1.25V | SSTL2 (Class I and II) |
| 1.0V | GTL+ 2.5, GTL+ 3.3 |
| 0.75V | HSTL (Class I) |

`-inputdelay` *bits_setting*

Sets the input delay value (between 0 and 31). A five-bit programmable input delay element is associated with each I/O. Setting the value of this delay is optional for each input buffer within the bank (that is, you can enable or disable the delay element for the I/O). When the input buffer drives a register within the I/O, the delay element is

activated by default to ensure a zero hold-time. You can set the default for this property in Designer. The value of this delay is set on a bank-wide basis. You can enter one of the following values (0-31):

| Bits Setting | Delay | Bits Setting | Delay |
|---|---|---|---|
| 0 | 0.54 | 16 | 2.01 |
| 1 | 0.65 | 17 | 2.13 |
| 2 | 0.71 | 18 | 2.19 |
| 3 | 0.83 | 19 | 2.3 |
| 4 | 0.9 | 20 | 2.38 |
| 5 | 1.01 | 21 | 2.49 |
| 6 | 1.08 | 22 | 2.55 |
| 7 | 1.19 | 23 | 2.67 |
| 8 | 1.27 | 24 | 2.75 |
| 9 | 1.39 | 25 | 2.87 |
| 10 | 1.45 | 26 | 2.93 |
| 11 | 1.56 | 27 | 3.04 |
| 12 | 1.64 | 28 | 3.12 |
| 13 | 1.75 | 29 | 3.23 |
| 14 > | 1.81 | 30 | 3.29 |
| 15 | 1.93 | 31 | 3.41 |

`-lpinput` *`value`*

Enables or disables the low-power mode for input buffers. You can enter one of the following values:

| Value | Description |
|---|---|
| on | Enables the low-power mode for input buffers. |
| off | Disables the low-power mode for input buffers. |

`-lpoutput value`

Enables or disables the low-power mode for output buffers. You can enter one of the following values:

| Value | Description |
|---|---|
| on | Enables the low-power mode for output buffers. |
| off | Disables the low-power mode for output buffers. |

`-fixed value`

Specifies if the I/O technologies (vcci and vccr voltage) assigned to the bank are locked. You can enter one of the following values:

| Value | Description |
|---|---|
| yes | The technologies are locked. |
| no | The technologies are not locked. |

`-vrefpins value`

Specifies if the I/O technologies (vcci and vccr voltage) assigned to the bank are locked. This option is only supported by ProASIC3E, IGLOOe and ProASIC3L(3000 die only) devices. You can enter one of the following values:

| Value | Description |
|---|---|
| default | Because the VREF pins are not locked, the I/O Bank Assigner can assign a VREF pin. |
| pinnum | The VREF pin(s) that are locked when the -fixed option is "yes". The I/O Bank Assigner cannot remove locked VREF pins. |

Note: Note: The set_vref and set_vref_defaults PDC commands are no longer supported. You can now use the set_iobanks command to set the vref pins. If you used the set_vref and set_vref_defaults commands in an existing design, when you export the PDC commands, the Designer software replaces the old set_vref and set_vref_defaults commands with the set_iobanks command.

## Supported Families

Axcelerator

Note: Note: Refer to the Axcelerator datasheet on the Actel web site for details about the legal values for the vcci and vref arguments.

## Exceptions

- Any pins assigned to the specified I/O bank that are incompatible with the default technology are unassigned.
- Delay values are approximate and will vary with process, temperature, and voltage.
- The arguments -inputdelay, -lpinput, and -lpoutput do not apply to RTAXS devices.

## Examples

The following example assigns 3.3 V to the input/output supply voltage (vcci) and 1.5 V to the input reference voltage (vref) for I/O bank 0. It also sets the input delay value to 1 and turns on the low-power mode for the input and output buffers.

```
set_iobank bank0 -vcci 3.3 -vref 1.5 -inputdelay 1 -lpinput on -lpoutput on
```

The following example shows that even though you can import a set_iobank command with the -vrefpins argument set to "default", the exported PDC file will show the specific default pins instead of "default."

Imported PDC file contains:

```
set_iobank bank3 -vcci 3.3 -vref 1.8  -inputdelay 1 -lpinput on -lpoutput on -fixed yes
  -vrefpins {default}
```

Exported PDC file contains:

```
set_iobank bank3 -vcci 3.3 -vref 1.8  -inputdelay 1 -lpinput on -lpoutput on -fixed yes
-vrefpins {N3 P8 M8}
```

## See Also

Configure I/O Bank

`reset_io`

`reset_iobank`

PDC Syntax Conventions

PDC Naming Conventions

# set_location

Assigns the specified macro to a particular location on the chip.

```
set_location macro_name -fixed value x y
```

## Arguments

*macro_name*

Specifies the name of the macro in the netlist to assign to a particular location on the chip.

-fixed *value*

Sets whether the location of this instance is fixed (that is, locked). Locked instances are not moved during layout. The default is yes. The following table shows the acceptable values for this argument:

| Value | Description |
|---|---|
| yes | The location of this instance is locked. |
| no | The location of this instance is unlocked. |

*x y*

The x and y coordinates specify where to place the macro on the chip. Use the ChipPlanner tool to determine the x and y coordinates of the location.

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

## Exceptions

- None

## Examples

This example assigns and locks the macro with the name "mem_data_in\[57\]" at the location x=7, y=2:

```
set_iobank mem_data_in\[57\] -fixed no 7 2
```

### See Also

Assign macro to location

set_multitile_location

PDC Syntax Conventions

PDC Naming Conventions

# set_multitile_location (IGLOO, Fusion, and ProASIC3)

Assigns specified two-tile and four-tile macros to specified locations on the chip. Use this command only for multi-tile, flip-flop macros and, in some cases, enable flip-flop macros).

```
set_multitile_location macro_name [-fixed value]\
-location {x y} \
-tile {name1 relative_x1 relative_y1} \
-tile {name2 relative_x2 relative_y2} \
[-tile {name3 relative_x3 relative_y3} \ ]
[-tile {name4 relative_x4 relative_y4} \ ]
```

## Arguments

*macro_name*

Specifies the hierarchical name of the macro in the netlist to assign to a particular location on the chip.

-fixed *value*

Sets whether the location of this set of macros is fixed (that is, locked). Locked macros are not moved during layout. The default is yes. The following table shows the acceptable values for this argument:

| Value | Description |
|---|---|
| yes | The location of this instance is locked. |
| no | The location of this instance is unlocked. |

-location {*x y*}

The x and y coordinates specify the absolute placement of the macro on the chip. You can use the ChipPlanner tool to determine the x and y coordinates of the location.

-tile {*name1 relative_x1 relative_y1*}

Specifies the hierarchical name and location, relative to the macro specified as the *macro_name*, of the first tile in a two- or four-tile macro. The relative placement of macro *name1* inside the macro cannot be offset by more than one. (See Notes below for placement rules.) If the macro uses four-tile macros, then you must define all four tiles. Likewise, if the macro uses two-tile macros, you must define both tiles.

You can place the following two-tile and four-tile macros with the set_multitile_location command:

| **Four-tile macro** | | | |
|---|---|---|---|
| DFN1P1C1 | DFI1P1C1 | DFN0P1C1 | DFI0P1C1 |
| **Two-tile macro** | | | |
| DLN1P1C1 | DLI1P1C1 | DLN0P1C1 | DLI0P1C1 |

Due to the ProASIC3 architecture, if the CLR and PRE pins are NOT driven by a clock net (global, quadrant or local clock net), the enable flip-flop macros (shown below) are mapped to two-tile flip-flop macros. When CLR and PRE pins are not driven by a clock net, you must use the set_multitile_location command instead of the set_location command.

| | | | | |
|---|---|---|---|---|
| DFN1E1C0 | DFN0E1C0 | DFN1E0C0 | DFN0E0C0 | DFN1E1C1 |
| DFN0E1C1 | DFN1E0C1 | DFN0E0C1 | DFN1E1P1 | DFN0E1P1 |
| DFN1E0P1 | DFN0E0P1 | DFN1E1P0 | DFN0E1P0 | DFN1E0P0 |
| DFN0E0P0 | DFI1E1C1 | DFI0E1C1 | DFI1E0C1 | DFI0E0C1 |
| DFI1E1C0 | DFI0E1C0 | DFI1E0C0 | DFI0E0C0 | DFI1E1P1 |
| DFI0E1P1 | DFI1E0P1 | DFI0E0P1 | DFI1E1P0 | DFI0E1P0 |
| DFI1E0P0 | DFI0E0P0 | | | |

During compile, Designer maps the specified enable flip-flop macro to a two-tiled macro.

If the CLR and PRE pins are driven by a clock net, Designer maps these macros to one tile during compile. In this case, you cannot use the set_multitile_location command to place them. Instead, you must use the set_location command.

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

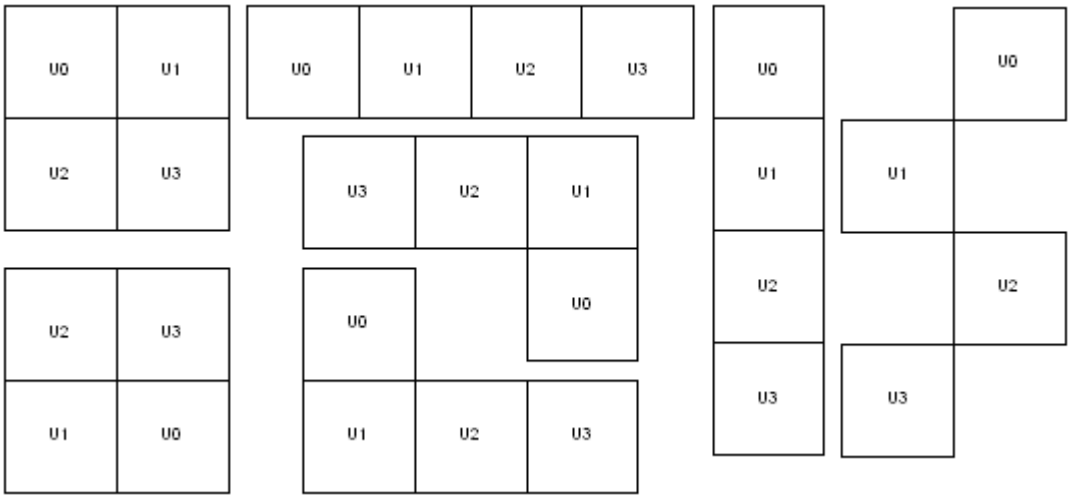
## Description

For two-tile flip-flop macros, the software appends U0 and U1 to the macro name. For four-tile flip-flop macros, the software appends U0, U1, U2 and U3 to the macro name. The macros specified in the -tile option cannot be offset by more than one.

To ensure efficiency, you must use local connections between certain tiles in the macros. The distance between U0 and U1, U1 and U2, and U2 and U3 must not be more than one in either direction (X or Y). The required local connection between tiles is denoted by the dashes below:

Four-tile macros: U0 --- U1 --- U2 --- U3 Two-tile macros: U0 --- U1

Examples of possible placement configurations:




## Exceptions

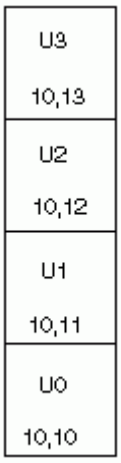
- None

## Examples

This example assigns and locks the macro with instance name “multi_tileff/U0 “ at the location X=10, Y=10 by specifying the relative positions of all the macros.

```
set_multitile_location multi_tileff -location {10 10} \
                       -tile { multi_tileff/U0 0 0 } \
                       -tile { multi_tileff/U1 0 1 } \
                       -tile { multi_tileff/U2 0 2 } \
                       -tile { multi_tileff/U3 0 3 } -fixed yes
```

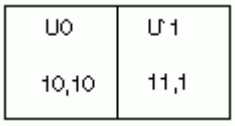
As a result of this command, the four-tile macro placement looks like this:



The second example shows you how to configure a two-tile macro:

```
set_multitile_location multi_tileff -location {10 10} \
                       -tile { multi_tileff/U0 0 0 } \
                       -tile { multi_tileff/U1 1 0 }
```

As a result of this command, the two-tile macro placement looks like this:




## See Also

Assign macro to location

set_location

PDC Syntax Conventions

PDC Naming Conventions

# set_net_critical

Sets the net criticality, which influences place-and-route in favor of performance.

```
set_net_criticalcriticality_number [ hier_net_name]+
```

## Arguments

*criticality_number*

Sets the criticality level from 1 to 10, with 1 being the least critical and 10 being the most critical. The default is 5. Criticality numbers are used in timing-driven place and route.

*hier_net_name*

Specifies the net name, which can be an AFL (Actel Flattened Netlist) net name or a net regular expression using wildcard characters. You must specify at least one net name. You can use the following wildcard characters in names:

| Wildcard | What It Does |
|---|---|
| \ | Interprets the next character as a non-special character |
| ? | Matches any single character |
| * | Matches any string |
| [] | Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range) |

Note: Note: This command must have at least two parameters.

## Supported Families

Axcelerator

## Description

Increasing a net's criticality forces place-and-route to keep instances connected to the specified net as close as possible at the cost of other (less critical) nets.

## Exceptions

- The net names are AFL names, which means they must be visible in Timer and ChipPlanner.

## Examples

This example sets the criticality level to 9 for all addr nets:

```
set_net_critical 9 addr*
```

### See Also

Set Net's Criticality

`reset_net_critical`

PDC Syntax Conventions

PDC Naming Conventions

# set_port_block

Sets properties on a port in the block flow. This PDC command applies to only one I/O.

```
set_port_block -namePortName -remove_ios value -add_interface value]
```

## Arguments

`-name portName`

Specify the name of the port.

`-remove_ios value`

Sets whether or not to remove I/Os connected to the specified port from the netlist. The following table shows the acceptable values for this argument:

| Value | Description |
|---|---|
| yes | Remove I/Os connected to the specified port from the netlist. |
| no | Do not remove I/Os connected to the specified port from the netlist. |

`-add_interface value`

Adds an interface macro each time the fanout of the net connected to the port is greater than the value specified. The value must be a positive integer.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator and RTAX

## Exceptions

- You must import this PDC command as a source file, not as an auxiliary file.
- TRIBUFF and BIBUF macros cannot be removed even if you specify "-remove_ios yes".
- You must enable the block flow before calling this command. To enable the block flow, either select the "Enable block mode" option in the **Setup Design** dialog box, or use the -block argument in the new_design Tcl command to enable block mode.

## Examples

This example removes any I/Os connected to portA, excluding TRIBUFF and BIBUF I/Os:

```
set_port_block -name portA -remove_ios yes
```

### See Also

new_design

# set_preserve

Sets a preserve property on instances before compile, so compile will preserve these instances and not combine them.

```
set_preserve hier_inst_name
```

## Arguments

*hier_inst_name*

Specifies the full hierarchical name of the macro in the netlist to preserve.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, and Axcelerator

## Exceptions

- This command is not supported in post compiled designs. If importing a PDC file that includes this command, you must import it as a source file.

## Examples

In some cases, you may want to preserve some instances for timing purposes. For example, you may want registers to be combined with input of a bibuf and keep the output as it is.

If the outbuf of a bi-directional signal test[1] needs to be preserved while inbuf is required to combine with the registers, use the following PDC commands:

```
set_io test\[1\] -REGISTER yes
set_preserve test\[31\]
```

If any internal instance is required to be preserved, use the set_preserve command as shown in the following example:

```
set_preserve top/inst1 top/inst2
```

### See Also

PDC Syntax Conventions

PDC Naming Conventions

I/O Register Combining

# unassign_global_clock

Demotes clock nets to regular nets. The unassign_global_clock command is not supported in auxiliary PDC files.

```
unassign_global_clock -net netname
```

## Arguments

`-net netname`

Specifies the name of the clock net to demote to a regular net.

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

## Exceptions

- You cannot assign "essential" clock nets to regular nets. Clock nets that are driven by the following macros are "essential" global nets: CLKDLY, PLL, and CLKBIBUF.

## Examples

```
unassign_global_clock -net globalReset
```

### See Also

assign_global_clock

PDC Syntax Conventions

PDC Naming Conventions

# unassign_local_clock

Unassigns the specified net from a LocalClock region.

```
unassign_local_clock -net netname
```

## Arguments

`-net netname`

Specifies the name of the net to unassign.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- This command is not supported in auxiliary PDC files. If importing a PDC file that includes this command, you must import it as a source file.

## Examples

This example unassigns the net named reset_n from the local clock region:

```
unassign_local_clock -net reset_n
```

### See Also

assign_local_clock (Axcelerator)
assign_local_clock (IGLOO, Fusion, and ProASIC3)
PDC Syntax Conventions
PDC Naming Conventions

# unassign_macro_from_region

Specifies the name of the macro to be unassigned.

```
unassign_macro_from_region [region_name] macro_name
```

## Arguments

*region_name*

Specifies the region where the macro or macros are to be removed.

*macro_name*

Specifies the macro to be unassigned from the region. Macro names are case sensitive. You can unassign a collection of macros by assigning a prefix to their names. You cannot use hierarchical net names from ADL. However, you can use the following wildcard characters in macro names:

| Wildcard | What It Does |
|---|---|
| \ | Interprets the next character as a non-special character |
| ? | Matches any single character |
| * | Matches any string |
| [] | Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range) |

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

If the macro was not previously assigned, an error message is generated.

## Examples

```
unassign_macro_from_region macro21
```

### See Also

Unassign macro from region

assign_net_macros

PDC Syntax Conventions

PDC Naming Conventions

# unassign_net_macros

Unassigns macros connected to a specified net.

```
unassign_net_macros region_name [net1]+
```

## Arguments

*region_name*

Specifies the name of the region containing the macros in the net(s) to unassign.

*net1*

Specifies the name of the net(s) that contain the macros to unassign from the specified region. You must specify at least one net name. Optionally, you can specify additional nets to unassign.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- If the region is currently not assigned, an error message appears in the Log window if you try to unassign it.

## Examples

```
unassign_net_macros cluster_region1  keyin1intZ0Z_62
```

### See Also

Unassign macros on net from region

assign_net_macros

PDC Syntax Conventions

PDC Naming Conventions

# unassign_quadrant_clock

Unassigns the specified net from a QuadrantClock region. If the unassigned net is a clock net, it will not be demoted to a regular net.

```
unassign_quadrant_clock -net netname
```

## Arguments

`-net netname`

Specifies the name of the net to unassign from a quadrant clock region.

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

## Exceptions

- This command is not supported in auxiliary PDC files. If importing a PDC file that includes this command, you must import it as a source file.

## Examples

This example unassigns the net named qnet_n from the quadrant clock region:

```
unassign_quadrant_clock -net qnet_n
```

### See Also

Unassign macro from region

`assign_quadrant_clock`

PDC Syntax Conventions

PDC Naming Conventions

# undefine_region

Removes the specified region. All macros assigned to the region are unassigned.

```
undefine_region region_name
```

## Arguments

*region_name*

Specifies the region to be removed.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- To use this command, the region must have been previously defined.

## Examples

```
undefine_region cluster_region1
```

### See Also

Delete region

define_region

PDC Syntax Conventions

PDC Naming Conventions

# unreserve

Resets the named pins in the current device, so they are no longer reserved. You can then use these pins in your design.

```
unreserve -pinname "list of package pins"
```

## Arguments

`-pinname "list of package pins"`

Specifies the package pin name(s) to unreserve.

## Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion and Axcelerator

## Exceptions

- None.

## Examples

```
unreserve -pinname "F2"
unreserve -pinname "F2 B4 B3"
unreserve -pinname "124 63"
```

### See Also

`reserve`

PDC Syntax Conventions

PDC Naming Conventions

# Constraints by File Format: GCF Command Reference

# About GCF Files

You can define the following types of constraints in a GCF constraint file:

- Global resource constraints
- Netlist Optimization Constraints
- Placement constraints

Note: Note: Only ProASIC devices support GCF Timing commands. Use the SDC timing commands to import timing constraints for ProASIC$^{PLUS}$ devices. All other GCF commands are supported by both ProASIC and ProASIC$^{PLUS}$ devices.

Designer supports the following GCF commands.

| GCF Placement Command | Action |
|---|---|
| net_critical_ports | Specifies a specific subset of critical ports on a net |
| set_critical | Specifies critical nets and their relative criticality over other critical nets. |
| set_critical_port | Identifies design I/O ports that have above-normal criticality |
| set_empty_io | Specifies a location in which no I/O pin should be placed |
| set_empty_location | Specifies a location in which no cell should be placed |
| set_initial_io | Initially assigns package pins to I/O ports or sets the location of I/O ports |
| set_initial_location | Initially sets the location of a cell instance at specified x, y coordinates |
| set_io | Either assigns package pins to I/O ports or sets the location of I/O ports at a specified location on a device |
| set_io_region | Places specific I/O instances into a target rectangular region |
| set_location | Assigns a cell instance to the specified x,y coordinates |
| set_memory_region | Creates and assigns memory to a region |

| GCF Placement Command | Action |
|---|---|
| set_net_region | Places all the connected instances, driver, and all the driven instances for the net(s) into the specified rectangle |

| Global Resource Commands | Action |
|---|---|
| dont_fix_globals | Disables the default action that automatically corrects the choice of global assignment to use only the highest fanout nets |
| read | Specifies the name of the constraint file containing the constraints to use |
| set_auto_global | Specifies the maximum number of global resources to use |
| set_auto_global_fanout | Sets the minimum fan-out a net must have to be considered for automatic promotion to a global |
| set_global | Classifies the specified nets as global nets |
| set_noglobal | Classifies nets to avoid automatic promotion to global nets |
| use_global | Specifies a single spine (LocalClock) or a rectangle of spine region which may encompass more than one spine region |
| | |

| Netlist Optimization Command | Action |
|---|---|
| dont_optimize | Prevents buffers and inverters from being removed from the netlist |
| dont_touch | Enables you to selectively disable optimization of named hierarchical instances |
| optimize | Turns on all netlist optimizations (the default mode);enables you to remove buffers and inverters from the netlist |
| set_max_fanout | Sets the maximum fan-out limit on the specified nets when |

| Netlist Optimization Command | Action |
|---|---|
| | optimizing buffers and inverters |

## See Also

Constraint Entry

GCF Syntax Conventions

Importing Constraint Files

# GCF Syntax Conventions

A ProASIC constraint consists of a statement and an argument, terminated by a semicolon. Statements are not case sensitive. However, cell instance, net, and port names used as arguments may be quoted and are case sensitive. Except for white spaces, all ASCII characters can be used. Comments are allowed in constraints files and must be preceded by two forward slashes (//). Time values are given in nanoseconds. When constraints are duplicated, the last one specified for a specific item overwrites any previous similar constraints already specified for the considered item.

This section describes syntax conventions for notation, user data variables, and comments. Comments begin with double slashes (//) and are terminated by a newline character.

Table 6 · Syntax Conventions for Notation

| Notation | Description |
|---|---|
| item | Represents a syntax item |
| item ::= definition | item is defined as definition |
| item ::= definition1 \|\|= definition2 | item is defined as either definition1 or definition2 (Multiple alternative syntax definitions are allowed) |
| [ item ] | Item is optional |
| { item } | Item is a list of required items. At least one item must appear. |
| KEYWORD | Keywords appear in uppercase characters in bold type for easy identification, but are not case sensitive. |
| VARIABLE | Represents a variable and appears in uppercase characters for easy identification |

Table 7 · Syntax Conventions for User Data Variables

| Notation | Description |
|---|---|
| FILEIDENTIFIER | Represents a hierarchical filename. |
| IDENTIFIER | Represents the name of a design object. Can be a block, cell instance, net, or port. IDENTIFIERS can use any ASCII character except the white space and the slash (/), which is the hierarchical divider character (see |

| Notation | Description |
|---|---|
| | QPATH below). IDENTIFIERS are case sensitive |
| POSFLOAT | Represents a positive real number; for example, 4.3, 1.15, 2.35 |
| POSNUMBER | Represents a positive integer; for example, 1, 12, 140, 64. When representing time, POSNUMBER is expressed in nanoseconds (ns) |
| QPATH | Represents a hierarchical IDENTIFIER. The levels of the hierarchy are represented by IDENTIFIERS divided by a slash (/). The QPATH hierarchical IDENTIFIER may or may not be quoted |

# Placement Constraints

It is possible to use placement constraints to specify block-instance and macro placement. You can specify initial, fixed, region, and macro placements. Also, placement obstructions (locations that are not to be used and thus to be kept empty during placement instances) can be specified.

For example, a constraint that places two connected blocks close together usually improves the timing performance for those blocks. Similarly, a constraint that assigns an I/O pin to a particular net forces the router to make the connection between the driving or receiving cell and the I/O itself.

Like all constraints, placement constraints limit Designer's freedom when processing the design. For instance, assigning a fixed location makes that location unavailable during placement optimization. Such removal usually limits the program's ability to produce a chip-wide solution.

## See Also

Constraint Support by Family

Constraint Entry Table

# Macro

Defines the locations of a sub-design as a macro so that you are able to reuse this placement in different instantiations of the sub-design.

```
macro name (x1, y1 x2, y2) {macro_statements}
```

## Arguments

*name*

Specifies the macro name identifier

*x1, y1*

Specifies the lower-left coordinate of the macro.

*x2, y2*

Specifies the upper-right coordinate of the macro.

*macro_statements*

Any of the following statements:

set_location (x, y) *hier_inst_name;*

set_initial_location (x, y) *hier_inst_name;*

set_empty_location (x, y);

You can use the *set_location* or *set_initial_location* statements to place or initially place a sub-design instance by calling its macro and then applying a translation and rotation.

set_initial_location (x, y) *hier_inst_name*

*macro_name* [transformations];

Where hier_inst_name is the hierarchical name of the instance of the sub-design, x, y is the final location of the lower-left corner of the macro after all transformations have been completed, macro_name is the name of previously defined macro, and transformations are optional, and any of the following in any order:

- **flip lr** - flip cell from left to right
- **flip ud** - flip cell from up to down
- **rotate 90 cw** - rotate 90˚ clockwise
- **rotate 270 cw** - rotate 270˚ clockwise
- **rotate 90 ccw** - rotate 90˚ counter-clockwise
- **rotate 180 ccw** - rotate 180˚ counter-clockwise
- **rotate 270 ccw** - rotate 270˚ counter-clockwise

The transformations are processed in the order in which they are defined in the statement.

## Supported Families

ProASIC

## Description

The macro constraint must precede the corresponding set_location that places the macro in the GCF file(s) as in the following example:

```
macro mult (1,1 6,6) {
set_location...
}
```

Nested macros are not allowed. The location statements inside a macro definition must refer to individual cell instances and not to complete sub-designs.

## Exceptions

- None

## Examples

```
set_location (3,3) a/b mult flip lr;
set_initial_location (3,3) a/b mult flip lr;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# Package Pin and Pad Location

Generally, you are concerned with the mapping of signals (ports) to the pins of the selected package. However, you may want to control the allocation of signals to particular pads. This is accomplished by assigning ports to the pad location rather than to the package pin. Because all pads are pre-bonded to package pins, the effect is to assign ports to package pins, with the emphasis on pad location rather than package pin.

Pad location is described by the letters N (North), S (South), E (East) or W (West) followed by a space and a number. This location code determines the direction and offset of the pad with respect to the die.

The top edge of the viewer contains the North pads and the right edge contains the East pads. The number refers to the pad position along its edge. For example, N 48 corresponds to the 48th pad along the North edge of the die. The figure below shows the numbering system used for pad location.

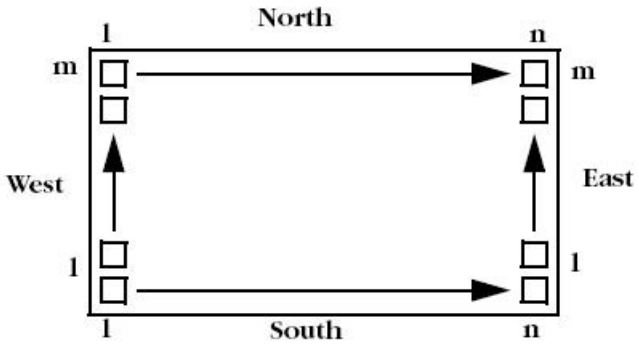



Figure 17 · Pad Location

# net_critical_ports

Defines a subset of critical ports on a net. The resulting netlist contains a buffer in the origin net, together with the critical ports. The inserted buffer drives the remaining connections on the original net.

```
net_critical__ports hier_net_name instance_port_name [ , instance_port_name ];
```

## Arguments

`hier_net_name`
Specifies the name of the hierarchical net(s).
`instance_port_name`
Specifies the name of the ports on the net(s).

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

The following example identifies two inputs of the net /u1/u2/net1 that are more critical than all other connections on that net. All other connections on the net will be buffered with a BUF cell that will be placed in a tile to reduce fanout delay on the specified inputs:

```
net_critical_ports /u1/u2/net1 nandbk1.A sigproc.C;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_critical

Specifies critical nets and their relative criticality over other critical nets.

```
set_critical criticality_numberhier_net_name
[,hier_net_name…];
```

## Arguments

`criticality_number`

Sets the criticality level from 1 to 5, with 1 being the least critical and 5 being the most critical. The default is 1. Criticality numbers are used in timing-driven place and route.

*`hier_net_name`*

Specifies the full hierarchical name of the net(s).

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

The following example sets the timing of u1/u2/ net1 more critical than u1/u2/net5 and u1/u2/net3:

```
set_critical 5 /u1/u2/net1;
set_critical 2 /u1/u2/net5, u1/u2/net3;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_critical_port

Identifies design I/O ports that have above-normal criticality.

```
set_critical_port criticality_number signal_name
[,signal_name…];
```

## Arguments

`criticality_number`

A number from 1 to 5 that is relative in criticality to other critical I/O signals, from least (1) to most critical (5). The default is 1.

`signal_name`

Specifies the name of a user-defined signal associated with a specific I/O pin on the port.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

The following example sets all nets associated with device ports IOBus[3] and IOBus[5] to criticality number 3:

```
set_critical_port 3 IOBus[3], IOBus[5];
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_empty_io

Specifies a location in which no I/O pin should be placed. The location can be specified by side and offset or by name.

```
set_empty_io { package_pin| pad_location};
```

## Arguments

`package_pin`

The symbolic name for the pin.

`pad_location`

Specifies the location of the pad in the layout.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

The following example forces pin B5 and the pin associated with the fourth tile on the North side to be empty:

```
set_empty_io B5, (N,4);
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_empty_location

Specifies a location in which no cell should be placed.

```
set_empty_location (x ,y);
set_empty_location (xbl ,ybl xtr ,ytr);
```

## Arguments

`x, y`

Specifies the tile coordinates for the empty cell location.

$x_{bl}$ ,$y_{bl}$ $x_{tr}$ ,$y_{tr}$

Specifies the tile coordinates for the bottom-left and top-right corner of the region.

Note: Only white spaces are allowed between the coordinates.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

The following example informs the placement program that location (3, 7) is unavailable for cell placement:

```
set_empty_location( 3 ,7);
set_empty_location(113 ,1 60 ,80 );
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_initial_io

Initially assigns package pins to I/O ports or sets the location of I/O ports at a specified side of a device. The placer can reassign or relocate the cells during place-and-route.

```
set_initial_io { package_pin| pad_location} io_port_name
[, io_port_name , … ];
```

## Arguments

*package_pin*

A package pin number for a specified I/O cell. If you use *package_pin*, only one *io_port_name* argument is allowed (required if no pin location is given).

*pad_location*

A pad location number on the chip. It constrains the pin location of a specified I/O cell to a specific pad location on the chip. Only one *io_port_name* argument is allowed. You can specify multiple, comma-separated ports only when they are assigned to a location.

*io_port_name*

The name of the I/O port to assign to the specified package pin.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

The following example initially places the I/O associated with net in3 to package pin A11:

```
set_initial_io A11 in3;
```

The next example initially places the I/O associated with net in4 on the fourth tile:

```
set_initial_io (1,4) in4;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_initial_location

Initially sets the location of a cell instance at specified x, y coordinates. The placer can relocate the cell instance during place-and-route.

```
set_initial_location ( x, y) hier_inst_name;
```

## Arguments

`x, y`

Specifies the tile coordinates for the location of a specified cell instance.

*`hier_inst_name`*

Specifies the hierarchical path to a cell instance.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

```
set_initial_location (43,105) bk3/fp5/nand3_4;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_io

Either assigns package pins to I/O ports or sets the location of I/O ports at a specified location on a device. This hard constraint cannot be overruled by the placer. This constraint also may have an impact on the timing results of a design.

```
set_io {pinName |location_Definition};
```

## Arguments

`pinName`

Specifies the name of a pin to assign to a port.

`location_Definition`

Specifies the location of the I/O port.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- If a hard constraint is not suitable, use the set_initial_io constraint.

## Examples

```
set_io A9 in1;
set_io (2,22) in2;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_io_region

Enables you to place specific I/O instances into a target rectangular region. The global I/Os are excluded from this constraint.

```
set_io_region (x1,y1  x2, y2) p1 [, p2, p3, .... , pn] ;"
```

## Arguments

`x1, y1  x2, y2`

Specifies the lower-left and upper-right corners of the rectangle that define the region.

`p1....pn`

Specifies one or more I/O instance names or ports.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

Multiple instances or ports must be separated by commas as shown in the following example:

```
set_io_region (0,41 0,48) "acc[3]", "acc[4]";
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_location

Assigns a cell instance to the location specified by the x,y coordinates. The placer cannot relocate the cell instance during place-and-route.

```
set_location (x, y )  hier_inst_name ;
set_location (xbl ,ybl xtr ,ytr)  hier_inst_name/*;
```

## Arguments

*x, y*

Specifies the tile coordinates for the location of a cell.

*hier_inst_name*

Specifies the hierarchical path to a cell instance.

$x_{bl}$ ,$y_{bl}$ $x_{tr}$ ,$y_{tr}$

Specifies the tile coordinates for the bottom-left and top-right corner of the region.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

```
set_location (1,15) u4/u3/nand3_4;
set_location (1,1 32,32) datapath/*;
```

## set_location macro call

This statement has been extended to allow you to place a sub-design instance by calling its macro and then applying a translation and rotation.

```
set_location (x, y) hier_subdesign_inst_name macro_name
[transformations];
```

where

`hier_subdesign_inst_name` is the hierarchical name of the instance of the sub-design; `(x, y)` is the final location of the lower left corner of the macro after all transformations have been completed; `macro_name` is the name of the previously defined macro; `transformations` are optional, and any of the following in any order:

- `flip lr` - flip cell from left to right
- `flip ud` - flip cell from up to down
- `rotate 90 cw` - rotate 90 ° clockwise

- `rotate 180 cw` - rotate 180 ° clockwise
- `rotate 270 cw` - rotate 270 ° clockwise
- `rotate 90 ccw` - rotate 90 ° counter-clockwise
- `rotate 180 ccw` - rotate 180 ° counter-clockwise

## See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_memory_region

Creates a region and assigns memory to it.

```
set_memory_region (x1,y1 x2,y2) memory1_name [,...,memoryn_name];
```

## Arguments

*(x1,y1 x2,y2)*

The coordinates x1 and y1 specify the bottom-left corner, while x2 and y2 specify the top-right corner of the rectangle that defines the region in which the memory macros will be assigned. The macros are constrained to this region.

*memory1_name,...*

Specifies the memory macro(s) to assign to the region. Macro names are hierarchical names in the user netlist. You can use wildcards in macro names. The wildcard character (*) matches any string.

## Supported Families

ProASIC PLUS and ProASIC

## Description

You can only assign names of memory macros to the region. Do not specify names of individual tiles. For cascaded memory, the set_memory_region constraint applies to the whole cascaded block, even if your statement mentions only one macro out of the whole cascaded block.

## Exceptions

- None

## Examples

```
set_memory_region (1,101 32,101) M1/U0;
set_memory_region (1,101 48,101) M1/U0,M1/U1;
set_memory_region (1,101 128,101) M1/U*;
```

Note: You can also use set_net_region and use_global to assign memory to regions.

Note: Additionally, you can use the MultiView Navigator (MVN) to create regions that include memory. MVN regions can span core, I/O, and/or memory.

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_net_region

Places all the connected instances, driver, and all the driven instances for the net(s) into the specified rectangle.

```
set_net_region (x1,y1 x2,y2) <net_name_wildcard>;
```

## Arguments

(*x1,y1 x2,y2*)

The coordinates x1 and y1 specify the bottom-left corner, while x2 and y2 specify the top-right corner of the rectangle that defines the region. The nets are constrained to this region.

`net_name_wildcard`

Specifies the net(s) to assign to the region. You can use wildcards in net names. The wildcard character (*) matches any string.

## Supported Families

ProASIC PLUS and ProASIC

## Description

This command puts the region constraint on all the connected instances, which will be processed by the placer. Only white spaces are allowed between the coordinates.

The RAMs and I/Os are assigned to the LocalClock region unless the Compile option “Include RAM and I/O in Spine and Net Regions” is cleared. For designs created with v5.1 or earlier, this option is cleared by default. See "Compile Options" in the online help for more information.

## Exceptions

- None

Examples

```
set_net_region (1,101 32,101) addr*;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# About Global Resource Constraints

Each ProASIC and ProASIC[PLUS] device includes four global networks that have access to every tile. These four global networks provide high speed, low skew routing resources to signals such as clocks and global resets.

Once the netlist is imported, Designer sets global resource parameters and promotes the highest fanout nets to the remaining global resources unless the dont_fix_globals statement has been specified in a constraint file.

Note: Note: When using the dont_fix_globals statement, global assignments made in the constraint files and design netlist will be honored (the constraint file entries will take precedence).

These global resource parameters can be supplemented by including global resource constraints in a constraint file. Global resource constraints can define which signals are assigned to global resources and which signals cannot be promoted to global resources. Global resource constraints can also override the default action that selects high fanout nets for use by the global resources.

## See Also

Constraint Support by Family

Constraint Entry Table

# Priority Order for Global Promotion

While assigning signals to global resources, Designer considers this information in the following priority:

1. The `set_global` and `set_io` statements (instances of those global cells, which cannot be demoted)
2. Nets with the highest potential fan-out above 32 (after removal of all buffers and inverters)
3. Global cell instantiation in a netlist (global cells which can be demoted)

Note: Note: By default, a net with a fan-out of less than 32 will not be promoted to global automatically unless the `set_global` or `set_io` constraint statements is used for this net. You can override this threshold of 32 by using the `set_auto_global_fanout` constraint statement.

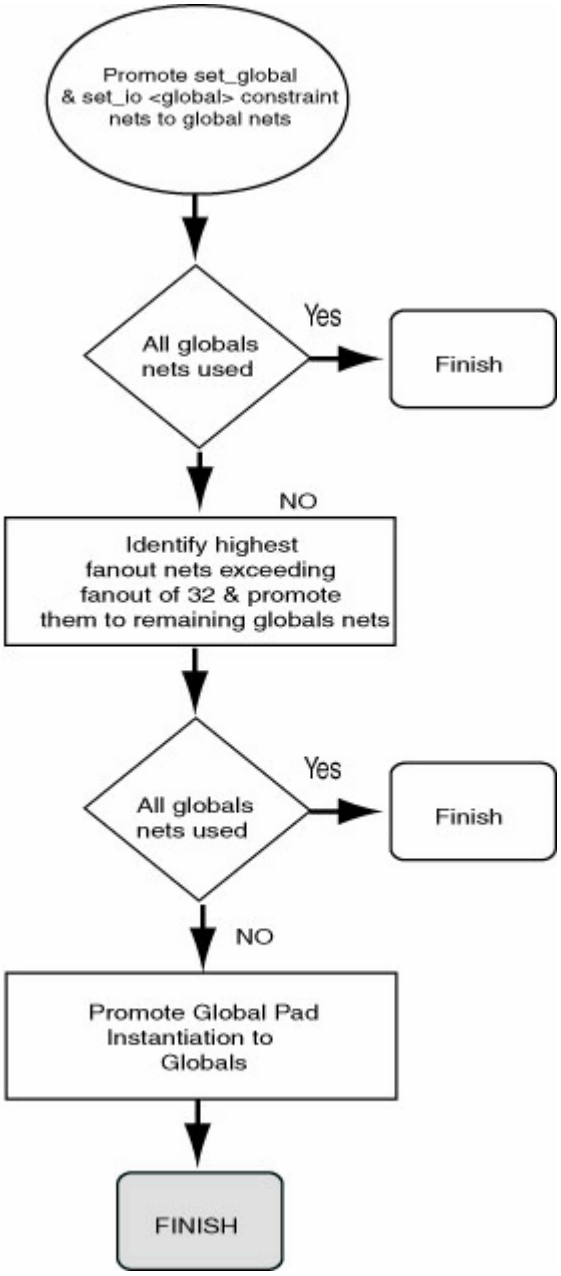



Figure 18 · Priority Order for Global Promotion

# dont_fix_globals

Disables the default action that automatically corrects the choice of global assignment to use only the highest fan-out nets.

```
dont_fix_globals;
```

## Arguments

- None

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

If you do not want the Designer software to automatically assign the highest fan-out nets to the available global resources but respect your choice of global nets instead, then include the following statement in a constraint file:

```
dont_fix_globals;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# read

Specifies the name of the constraint file containing the constraints to use.

```
read [ -eco] [ -initial] file ;
```

## Arguments

`-eco`

Specifies that the constraint file is to be read in eco (engineering change order) mode. In this mode, no errors will be reported when certain nets or instances are not found in the design. Instead a warning is generated.

`-initial`

Specifies that the constraint file is to be read in initial mode. In this mode, all fixed location statements will be interpreted as initial locations instead.

`file (required)`

The name of the constraint file, surrounded by double quotes.

## Supported Families

ProASIC PLUS and ProASIC

## Description

A constraint file can contain multiple read statements. For example, you can put pin assignments in one file, optimization constraints in another, placement constraints in yet another, and read them all in through a master constraint file.

## Exceptions

- None

## Examples

The following example instruct the Designer software to use constraints from the GCF files pinmap.gcf and decoder.gcf. A full path specification is given for pinmap.gcf. The file decoder.gcf has no path specification and is assumed to be in the design working directory.

```
read "/net/aries/designs/pinmap.gcf";
read "decoder.gcf";
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_auto_global

Specifies the maximum number of global resources to use. The Designer software assigns global resources to high fan-out signals automatically.

```
set_auto_global number;
```

## Arguments

`number`

The maximum number of global resources to use in your design. If you specify a number that exceeds the actual number of global resources available in the device, the Designer software ignores the statement. If you specify 0, no automatic assignment to global resources occurs.

## Supported Families

ProASIC

PLUS and ProASIC

## Exceptions

- None

## Examples

The following example specifies that of the possible four global nets available, the Designer software can automatically promote only two high fan-out nets:

```
set_auto_global 2;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_auto_global_fanout

Sets the minimum fan-out a net must have to be considered for automatic promotion to a global.

```
set_auto_global_fanout number;
```

## Arguments

`number`

The minimum fan-out for a net. The default number is 32.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

For example, the following statement determines that a net must have a fan-out of at least 12 before Designer considers it for automatic promotion to a global resource.

```
set_auto_global_fanout 12;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_global

Classifies the specified nets as global nets.

```
set_global hier_net_name [, hier_net_name ...];
```

## Arguments

*hier_net_name*

Specifies the name of the hierarchical net(s).

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Example

```
set_global u1/u3/net_clk, u3/u1/net_7;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# set_noglobal

Classifies nets to avoid automatic promotion to global nets. If the net was previously assigned to a global resource, this statement will demote it from that resource.

```
set_noglobal hier_net_name[ , hier_net_name… ];
```

## Arguments

*hier_net_name*

Specifies the name of the hierarchical net(s).

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Example

```
set_noglobal u2/u8/net_14;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# use_global

Specifies a single spine (LocalClock) or a rectangular spine region which may encompass more than one spine.

```
use_global spine <net_name>;
```

## Arguments

`spine`

Specifies one of the spines: T1 to T<n> or B1 to B<n>.

`net_name`

Specifies the name of a net.

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Example

If you specify the spine rectangle as B1, T3, the driven instances of the given net get a region constraint which encloses the rectangle, including the spine rectangle B1, T1, B2, T2, B2, T3.

```
use_global B1, T3 <net_name>;
```

The constraint tries to place the driver as close to the center of the rectangle as possible.

The RAMs and I/Os are assigned to the LocalClock region unless the Compile option "Include RAM and I/O in Spine and Net Regions" is cleared. For designs created with v5.1 or earlier, this option is cleared by default. See "Compile Options" in the online help for more information.

You can specify the following type of rectangles:

- Bn, Bm : n<=m will mean Bn, Bn+1, ... Bm
- Tn, Tm : n<=m will mean Tn, Tn+1, ... Tm
- Bn, Tm : n<=m will mean Bn, Tn, Bn+1,Tn+1 ... Bm, Tm
- Tn, Bm : n<=m will mean Bn, Tn, Bn+1,Tn+1 ... Bm, Tm

The following table summarizes the available spines.

Table 8 · Global Spine Usage

| Device | Spine |
|---|---|
| A500K050 | T1 to T3 |

| Device | Spine |
|---|---|
| | B1 to B3 |
| A500K130 | T1 to T5 |
| | B1 to B5 |
| A500K180 | T1 to T6 |
| | B1 to B6 |
| A500K270 | T1 to T7 |
| | B1 to B7 |
| APA075 | T1 to T3 |
| | B1 to B3 |
| APA150 | T1 to T4 |
| | B1 to B4 |
| APA300 | T1 to T4 |
| | B1 to B4 |
| APA450 | T1 to T6 |
| | B1 to B4 |
| APA600 | T1 to T7 |
| | B1 to B7 |
| APA750 | T1 to T8 |
| | B1 to B8 |
| APA1000 | T1 to T11 |
| | B1 to B11 |

Note: Note: T1 and B1 are the leftmost top and bottom global spines, respectively.

## See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

# Netlist Optimization Constraints

Netlist optimization attempts to remove all cells from a netlist that have no effect on the functional behavior of the circuit. This reduces the overall size of a design and produces faster place-and-route times. This optimization is based on the propagation of constants and inverter pushing and takes advantage of inverted inputs of the basic logic elements.

Netlist optimization can be controlled by including netlist optimization constraints in constraint files submitted to Designer.

By default, all optimizations will be performed on the netlist. To control the amount of optimization that takes place, netlist optimization constraints can be used. Netlist optimization constraints can turn off all optimizations or disable the default action that allows all optimizations to limit the type of optimizations performed. The constraints can also define a maximum fanout to be allowed after optimizations are performed and isolate particular instances and hierarchical blocks from the effect of optimization.

After completion of netlist optimization, the design is a functionally identical representation of the design produced internally for use by Designer. View the design's layout after successful placement and routing. After optimization, a number of instances that do not contribute to the functionality of the design may have been removed.

To keep the SDF file consistent with the original input netlist, deleted cells are written with zero delay so that back-annotation is performed transparently.

### See Also

Constraint Support by Family

Constraint Entry Table

# dont_optimize

Prevents buffers and inverters from being removed from the netlist.

```
dont_optimize [option];
```

## Arguments

[*option* ]

Specifies the netlist optimization option to turn off and preserve.

| Option | Description |
|---|---|
| buffer | Prevents buffers from being removed from the design |
| inverter | Prevents inverters from being removed from the design |
| const | Replaces all logical elements with one or more inputs connected to a constant (logical "1" or "0") by the appropriate logic function. If the replacement logic function is identified as an inverter or buffer, that element is removed. |
| dangling | Removes all cells driving unconnected nets |

## Supported Families

ProASIC PLUS and ProASIC

## Description

This statement does not optimize your buffers or inverters; instead, it prevents them from being removed. When followed by one or more of the netlist optimization options, this statement turns off the named option (and preserves it).

If you have buffers or inverters that are connected to global nets, promoted global nets, or spine nets, this command is ignored and buffers and inverters are removed. To avoid removing them, use the dont_touch option.

## Exceptions

- None

## Examples

```
dont_optimize buffer inverter;
```

## See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

dont_touch

optimize

Netlist Optimization Constraints

# dont_touch

Enables you to selectively disable optimization of named hierarchical instances.

```
dont_touch {hier_instance_name [hier_instance_name...];
```

## Arguments

*hier_instance_name* [*hier_instance_name* ... ]

Specifies the name of the hierarchical instance(s) for which you want to disable optimization.

## Supported Families

ProASIC PLUS and ProASIC

## Description

You can use the wildcard character (*) to isolate all sub-blocks under the named block. If you use this constraint, any instances (including buffers and inverters that are connected to global nets, promoted global nets, and spine nets) stay intact.

## Exceptions

- Use “dont_touch” with instance names only. You cannot use this constraint with net names.

## Examples

The statement in this example will enable only the buffer and inverter optimization types and optimization will be done on all instances except those contained in the block called /U1/myblock.

```
dont_touch /U1/myblock/*;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

dont_optimize

Netlist Optimization Constraints

# Optimize

Turns on all netlist optimizations (the default mode). When followed by one or more of the netlist optimization types, this statement enables only the named optimization(s).

```
optimize [{ option}];
```

## Arguments

[{ *option* }]

Specifies the netlist optimization option to turn on.

| Option | Description |
|---|---|
| | Removes all buffers in the design, provided you have not exceeded the maximum fan-out |
| inverter | Removes all inverters in the design, provided you have not exceeded the maximum fan-out |
| const | Replaces all logical elements with one or more inputs connected to a constant (logical "1" or "0") by the appropriate logic function. If the replacement logic function is identified as an inverter or buffer, that element is removed. |
| dangling | Recursively removes all cells driving unconnected nets |

## Supported Families

ProASIC PLUS and ProASIC

## Exceptions

- None

## Examples

```
optimize buffer inverter;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

dont_optimize

# set_max_fanout

Sets the maximum fan-out limit on the specified nets when optimizing buffers and inverters.

```
set_max_fanout NUMBER net_name_wildcard;
```

## Arguments

`NUMBER net_name_wildcard`

Specifies the maximum fan-out limit on the specified net(s) during optimization.

## Supported Families

ProASIC PLUS and ProASIC

## Description

The set_max_fanout constraint is optimized to accept individual net names. If you specify a net name, the set_max_fanout constraint applies only to the named net or nets and not to the entire design.

The buffers and inverters are not removed if the fan-out for the given net exceeds the given limit. If no net name is given, then the command is applied to all the nets in the design. The net name can be a simple net or a name having wildcard characters.

## Exceptions

- None

## Examples

```
set_max_fanout 15 testNet*;
```

### See Also

Constraint Support by Family

Constraint Entry Table

GCF Syntax Conventions

Netlist Optimization Constraints

# About Delay Constraint Files (DCF)

Delay constraint information can be described in a *.dcf file and imported into Designer. Because the DCF language was developed to interact directly with the Timer tool and is therefore not a recommended method.

Note: Note: DCF files are only valid with earlier Antifuse families such as eX, SX-A, and SX. Although they are supported in eX and SX-A, Actel recommends that you use SDC files for all your constraints.

DCF files are platform dependent. If you transfer from PC to UNIX or vice-versa, you must manually translate carriage-returns (unix2dos, dos2unix, or via ftp). PC text files have an extra character for carriage returns compared to UNIX text files.

Supported command categories:

| Categories | Action |
|---|---|
| global_clocks | Describes the clock waveforms from the global clock distribution network; local clocks, such as gated clocks, are not directly supported |
| max_delays/min_delays | Describes max/min delays |
| io_arrival_times | Defines the arrival time to an input port |
| global_stops | Defines pins in don't care or false path |
| pin_loads | Defines the capacitance loading on package pins |

### See Also

Constraint entry

DCF syntax rules

Importing constraint files

# DCF Syntax Rules

The syntax rules for DCF are listed below. Note that these rules cannot be used as a parsing grammar. Terminal symbols are in upper case. Non-terminal symbols, which are enclosed with <>, are in lower case. Symbols enclosed with [] are optional. The symbol | separates alternatives.

```
<DCF> =
<sec_def_name>
<sec_io_arr>
<sec_min_del>
<sec_max_del>
<sec_clk>
<sec_global_stop>
<sec_def_name> =
SECTION TOP_LEVEL_DEF_NAME <stop>
<variable>.
END <stop>
<sec_io_arr> =
SECTION IO_ARRIVAL_TIMES <stop>
[<io_arr_clauses>]
```

```
END <stop>
<io_arr_clauses> = <io_arr_clause> | <io_arr_clause> <io_arr_clauses>
<io_arr_clause> = [<number>:] <number> <timeunit> <io_list>.
<io_list> = <io> | <io> <io_list>
<io> = INPAD | OUTPAD| <variable>
<sec_max_del> =
SECTION MAX_DELAYS <stop>
<delay_clauses>
END <stop>
<sec_min_del> =
SECTION MIN_DELAYS <stop>
<delay_clauses>
END <stop>
<delay_clauses> = <delay_clause> | <delay_clause> <delay_clauses>
<delay_clause> =
DELAY <time>; SOURCE <source_list>; SINK <sink_list>;
[STOP <stop_list>]; [PASS <pass_list>].
<source_list> = {<sources>} [EXCEPT {<sources>}]
<sources> = INPAD | CLOCKED | <name_list>
<name_list> = <variable> | <variable> <name_list>
<sink_list> = {<sinks>}[EXCEPT {<sinks>}]
<sinks> = OUTPAD | GATED | <name_list>
<stop_list> = {<name_list>} [EXCEPT {<sinks>}]
<pass_list> = {<name_list>} [EXCEPT {<sinks>}]
<sec_clk> =
SECTION GLOBAL_CLOCKS <stop>
[<waveform_clauses>]
[<relational_clauses>]
END <stop>
<waveform_clauses> = <waveform_clause> | <waveform_clause> <waveform_clauses>
<waveform_clause> = WAVEFORM <variable> RISE <time>
FALL<time> PERIOD <time> [EXCEPT SOURCE {macrolist}]
[EXCEPT SINK {macrolist}].
<relational_clauses> = <check_clause> | <check_clause> <check_clauses>
<check_clause> =
MULTICYCLE <variable> SOURCE CYCLE<value> [EXCEPT <name_list>]
[; DESTINATION <clkname> CYCLE<value> <clkname> CYCLE<value>
[EXCEPT<name_list>]].
<clkname> = <clockMacro>
<time> = <number> <unit>
<number> = <int>
<stop> =. | /* NULL */
<unit> = NS | MS | PS
<variable> = same as variable in C language.
<int> = same as int in C language.
sec_global_stops> =
Section GLOBAL_STOPS.
```

```
{<pinNameList>}.
End.
<sec_pin_loads> =
Section PIN_LOADS.
<pinLoadClauses>
End.
<pinLoadClauses> = <pinLoadClause> | <pinLoadCause>l<pinLoadClauses>]
<pinLoadClause> = <number> <capUnit> [TTL | CMOS] <pinNameList>.
<capUnit> = PF | NF | UF | MF
```

# io_arrival_times

## SECTION IO_ARRIVAL_TIMES

Use the IO_ARRIVAL_TIMES section to define the arrival time to an input port.

```
[early_arrival_time:] late_arrival_time timeunit {source_io_list} EXCEPT {source_io_list}.
```

## Arguments

*{early, late}_arrival_time*

Signal arrival time relative to the reference time.

*source_io_list*

An INPAD or primary input pin.

Note:

Note: Note: The section IO_ARRIVAL_TIMES can be empty. For example,

```
SECTION IO_ARRIVAL_TIMES.
END.
```

Note: is entirely equivalent to

```
SECTION IO_ARRIVAL_TIMES.
0 0 {INPAD}.
END
```

# max_delays/min_delays

## SECTION MAX_DELAYS/MIN_DELAYS

Use the MAX_DELAYS and MIN_DELAYS section to specify the maximum and minimum delays.

```
DELAY value timeunit; SOURCE {source_name_list}; SINK {sink_name_list}; [STOP
{stop_name_list}]; [PASS {bypass_name_list}].
```

## Arguments

`DELAY` *`value`*

Specifies an integer.

`timeunit`

Allows {PS, NS, US}.

*`source_name_list`*

A list of signal sources. It can be one of the following: a macro output pin, macro name, or primary input.

*`sink_name_list`*

A list of signal destinations. It can be a name of a macro or a primary output.

*`stop_name_list`*

The list of pin names through which further propagation of signals will not be considered. This allows you to eliminate certain paths from consideration.

*`bypass_name_list`*

The list of latches which are allowed to be intermediate path points. By default, latches are considered to be sinks or path terminals.

`INPAD/OUTPAD/GATED`

Valid values for any of the lists, such as source_name_list or stop_name_list.

Normally there is no need to specify any timing requirements from any source to any sink clocked by an external global clock. This timing requirement can be generated automatically from the GLOBAL_CLOCK specifications and the sequential elements setup and hold times. For example, the timing constraint from a primary input to a sequential element can be derived from the sequential elements clocking waveform and the signal arrival time of the primary input.

A problem exists when two different internally-generated clock signals interact. This is due to the unpredictable and unknown skew between the two clock networks because of the routing delays from:

PAD >> internalMacro >> CLKINT

where CLKINT is the input pin of the global clock distribution network. The automatically generated path constraints will not incorporate the skew between the two clocks. In such cases, the path constraints should be expressed explicitly using the MAX/MIN_DELAYS section.

Note: NOTE: The most stringent timing constraint dominates. Hence, all general constraints should be looser than the specific constraints. For example, in the following example, the 26.0ns constraint dominates the 42.0ns constraint:

```
DELAY 42.0 ns SOURCE INPAD SINK OUTPAD.
DELAY 26.0 ns SOURCE {$1I23:Q $1I24:Q} SINK {ack_0}.
```

If the general constraint is tighter than the specific constraint, the specific constraint will effectively become a no-operation. In the following example, the looser constraint of 42.0ns has no effect since the general constraint of 26.0ns dominates.

```
DELAY 26.0 ns SOURCE INPAD SINK OUTPAD.
DELAY 42.0 ns SOURCE {$1I23:Q $1I24:Q} SINK {ack_0}.
```

The section MAX_DELAYS can be empty if there are no purely combinatorial paths from external sources to external sinks, and if every sequential element in the design is clocked by an external global clock. In this case, the timing constraints are generated automatically using the information in the GLOBAL_CLOCK section. Likewise, the MIN_DELAYS section can be empty.

One final word about external/internal sinks and sources with regard to the flip-flops and/or latches in the IOs: these flip-flops act as internal, not external, sources/sinks.

# global_clocks

## SECTION GLOBAL_CLOCKS

Use the GLOBAL_CLOCKS section to describe the clock waveforms from the global clock distribution networks. Local clocks, such as gated clocks, are not directly supported. The clock waveforms are used to generate the timing constraints of the paths between two sequential elements. To allow more user control when clocks interact, there are provisions to specify the clock period transitions, which should be considered. By default, the closest transitions are used when two clocks interact. The clock waveform specification has the following format:

```
WAVEFORM clkname RISE value FALL value PERIOD value [EXCEPT SOURCE {sequential list }| EXCEPT
SINK{sequential list}].
```

## Arguments

`clkname`

Specifies the name of the macro driving the clock network.

`RISE/FALL/PERIOD`*`value`*

Specify the value as either an integer or a floating point number followed by an unit selected from {NS, US, PS}. The default time unit is 0.1ns.

`EXCEPT` *`{SOURCE|SINK} {sequential list}`*

Specify the list of sequential elements which should not be included as endpoints in the automatically generated paths involving sequential elements.

## MULTICLOCK

Use the MULTICLOCK specification to specify which clock periods should be considered during the generation of the path constraints involving sequential elements. The default specification is to consider only the closest clock periods of the SOURCE and DESTINATION clocks. This specification has the following syntax:

```
MULTICYCLE SOURCE clkA CYCLE value EXCEPT {seqlist}; DESTINATION clkB CYCLE value EXCEPT
{seqlist}.
MULTICYCLE SOURCE clkA CYCLE value EXCEPT {seqlist}.
```

### Arguments

`clkA/clkB`

The name of the macro driving the clock network.

`EXCEPT {seqlist}.`

By default all sequential elements clocked by the clock driver are included. The EXCEPT seqlist is a list of all the sequential elements or specific pins to be excluded.

`CYCLE value`

By default, the closest transitions are considered. CYCLE provides the ability to use transitions from one or more clock periods past the closest transition. CYCLE zero indicates the closest transitions. CYCLE one skips the closest set of transitions and uses the next set of transitions. The term cycle is used to avoid confusion with the term period in the clock waveform specification. This allows you to specify a cycling-stealing clocking regime.

# global_stops

## SECTION GLOBAL_STOPS

Use the GLOBAL_STOPS section to disable dont care/false paths by preventing the specified pins from being used in ANY timingcritical paths. Any path involving pins that appear in this section should be removed from consideration.

```
{<pinNameList>}.
End.
```

# pin_loads

## SECTION PIN_LOADS

Use the PIN_LOADS section to define the capacitance loading and logic (TTL/CMOS) at a package pin. The default logic family is TTL.

```
capValue capUnit [TTL/CMOS]{[macList|pinList]} [EXCEPT {}].
```

# About PIN Files

Note: Note: PIN files are only valid with earlier Antifuse families such as eX, SX-A, and SX. Although PIN files are supported in Axcelerator, Actel recommends that you use PDC files instead to set pin locations for Axcelerator.

You can describe the location of a pin in a *.PIN file and import it into Designer. PIN files contain two commands: DEF and PIN.

| Commands | Description |
|---|---|
| DEF | Define top-level design entity |
| PIN | Define I/O location |

## DEF

Syntax:

```
DEF <design_name>
```

The following example defines top-level structure as TARG32_WRP.

```
DEF TARG32_WRP.
```

## PIN

Syntax:

```
PIN <pin_name>;PIN:<package_pin_number>.
```

The following example assigns signal RST to package pin 156.

```
PIN RST; PIN:156.
```

### See Also

Assign I/O to pin

# I/O Standards

# I/O Standards Table

Use the I/O Standards table to see which I/O standards can be applied to each family.

Table 9 · I/O Standards

| I/O Standard | IGLOO | Fusion | ProASIC3 | Axcelerator | RTSX-S | SX-A |
|---|---|---|---|---|---|---|
| CMOS | | | | | X | |
| CUSTOM | | | | | X | X |
| GTLP25 | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| GTLP33 | IGLOOe only | X | ProASIC3E and ProASIC3L only | | | |
| GTL33 | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| GTL25 | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| HSTL1 | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| HSTLII | IGLOOe only | X | ProASIC3E and ProASIC3L only | | | |

| I/O Standard | IGLOO | Fusion | ProASIC3 | Axcelerator | RTSX-S | SX-A |
|---|---|---|---|---|---|---|
| LVCMOS33 | X | X | X | | | |
| LVCMOS25 | IGLOOe only | X | X | X | | |
| LVCMOS25_50 | X | X | X | | | |
| LVCMOS18 | X | X | X | X | | |
| LVCMOS15 | X | X | X | X | | |
| LVCMOS12 | X | | ProASIC3L only | | | |
| LVTTL | X | X | X | X | X | X |
| TTL | X | X | X | X | X | X |
| PCI | X | X | X | X | X | X |
| PCIX | X | X | X | X | | |
| SSTL2I and SSTL2II | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |
| SSTL3I and SSTL3II | IGLOOe only | X | ProASIC3E and ProASIC3L only | X | | |

Note: Note: 1.2 voltage is supported for ProASIC3 (A3PL), IGLOOe V2 only, IGLOO V2, and IGLOO PLUS devices only.

## See Also

I/O Standard

# I/O Standards Compatibility Matrix

Not all I/O standards are compatible with each other in the same device.

Use the following tables to determine which I/O standards are compatible for the following devices:

- IGLOOe, Fusion, ProASIC3L, and ProASIC3E
- IGLOO, IGLOO PLUS, and ProASIC3
- Axcelerator

# IGLOOe, ProASIC3L, ProASIC3E, SmartFusion and Fusion I/O Standards Compatibility Matrix

The following table displays all of the I/O standards available for IGLOOe, SmartFusion, Fusion, ProASIC3L, and ProASIC3E devices. Not all I/O standards are compatible with each other. Two I/O standards are compatible if they have identical VCCI values and if both of them need a VREF and their VREF values are identical.

Use the table below to determine which I/O standards are compatible with which for your device.

Table 10 · I/O Standards Compatibility for IGLOOe, ProASIC3L, ProASIC3E, SmartFusion and Fusion Devices

| I/O Standard | LVTTL | LVCMOS | LVCMOS | LVCMOS | LVCMOS | LVCMOS | LVCMOS 1.2 | PCI, PCIX | GTL+ 3.3 | GTL+ 2.5 | GTL 3.3 | GTL 2.5 | HSTL | HSTL | SSTL2 | SSTL3 | LVDS | LVPECL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LVTTL | X | X | | | | | | X | X | | X | | | | | X | | X |
| LVCMOS 3.3 | X | X | | | | | | X | X | | X | | | | | X | | X |
| LVCMOS 2.5 | | | X | X | | | | | | X | | X | | | X | | X | |
| LVCMOS 2.5/5.0 | | | X | X | | | | | | X | | X | | | X | | X | |
| LVCMOS 1.8 | | | | | X | | | | | | | | | | | | | |
| LVCMOS 1.5 | | | | | | X | | | | | | | X | | | X | | |
| LVCMOS 1.2 | | | | | | | x (see Note below) | | | | | | X | | | X | | |
| PCI, PCIX | X | X | | | | | | X | | | X | | | | | X | | X |
| GTL+ 3.3 | X | X | | | | | | | | | | | | | X | X | | X |
| GTL+ 2.5 | | | X | X | | | | | | | | | | | | | X | |
| GTL 3.3 | X | X | | | | | | | | | | | | | | | | X |
| GTL 2.5 | | | X | X | | | | | | | | | | | | | X | |
| HSTL Class I | | | | | | X | | | | | | | | | | | | |

| I/O Standard | LVTTL | LVCMOS | LVCMOS | LVCMOS | LVCMOS | LVCMOS | LVCMOS 1.2 | PCI, PCIX | GTL+ 3.3 | GTL+ 2.5 | GTL 3.3 | GTL 2.5 | HSTL | HSTL | SSTL2 | SSTL3 | LVDS | LVPECL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HSTL Class II | | | | | | X | | | | | | | | | | | | |
| SSTL2 Class I and II; | | | X | X | | | | | | | | | | | | | X | |
| SSTL3 Class I and II | X | X | | | | | | X | X | | | | | | X | X | | X |
| LVDS | | | X | X | | | | | | | | | | | | | X | |
| LVPECL | X | X | | | | | | | | | | | | | X | X | | X |

**Note: Only ProASIC3L, IGLOO, IGLOOe, and IGLOO PLUS devices support LVCMOS12.**

# IGLOO, IGLOO PLUS, and ProASIC3 I/O Standards Compatibility Matrix

The following table displays all of the I/O standards available for your IGLOO, IGLOO PLUS, and ProASIC3 devices. Not all I/O standards are compatible with each other. Two I/O standards are compatible if they have identical VCCI values.

Use the table below to determine which I/O standards are compatible with which for your device.

Table 11 · I/O Standards Compatibility for IGLOO, IGLOO PLUS, and ProASIC3 Devices

| I/O Standard | LVTTL | LVCMOS 3.3 | LVCMOS 2.5/5.0 | LVCMOS 1.8 | LVCMOS 1.5 | LVCMOS 1.2 | PCI, PCIX | LVDS | LVPECL |
|---|---|---|---|---|---|---|---|---|---|
| LVTTL | X | X | | | | | X | | X |
| LVCMOS 3.3 | X | X | | | | | X | | X |
| LVCMOS 2.5/5.0 | | | X | | | | | X | |
| LVCMOS 1.8 | | | | X | | | | | |
| LVCMOS 1.5 | | | | | X | | | | |
| LVCMOS 1.2 | | | | | | X (see Note below) | | | |
| PCI, PCIX | X | X | | | | | X | | X |
| LVDS | | | X | | | | | X | |
| LVPECL | X | X | | | | | X | | X |

Note: Note: Only ProASIC3L, IGLOO, IGLOOe, and IGLOO PLUS devices support LVCMOS12.

# Axcelerator I/O Standards Compatibility Matrix

The following table displays all of the I/O standards available for your Axcelerator device. Not all I/O standards are compatible with each other. Two I/O standards are compatible if they have identical VCCI values and if both of them need a VREF and their VREF values are identical.

Use the table below to determine which I/O standards are compatible with which for your device.

Table 12 · I/O Standards Compatibility for Axcelerator Devices

| I/O Standard | LVTTL | LVCMOS 2.5 | LVCMOS 1.8 | LVCMOS 1.5 | PCI, PCIX | GTL+ 3.3 | GTL+ 2.5 | HSTL Class I | SSTL2 Class I and II | SSTL3 Class I and II | LVDS | LVPECL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LVTTL | X | | | | X | X | | | | X | | X |
| LVCMOS 2.5 | | X | | | | | X | | X | | X | |
| LVCMOS 1.8 | | | X | | | | | | | | | |
| LVCMOS 1.5 | | | | X | | | | X | | | | |
| PCI, PCIX | X | | | | X | X | | | | X | | X |
| GTL+ 3.3 | X | | | | X | X | | | | | | X |
| GTL+ 2.5 | | X | | | | | X | | | | | |
| HSTL Class I | | | | X | | | | X | | | | |
| SSTL2 Class I and II; | | X | | | | | | | X | | X | |
| SSTL3 Class I and II | X | | | | X | | | | | X | | X |
| LVDS | | X | | | | | X | | X | | X | |

| I/O Standard | LVTTL | LVCMOS 2.5 | LVCMOS 1.8 | LVCMOS 1.5 | PCI, PCIX | GTL+ 3.3 | GTL+ 2.5 | HSTL Class I | SSTL2 Class I and II | SSTL3 Class I and II | LVDS | LVPECL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LVPECL | X | | | | X | X | | | | X | | X |

# I/O Standards and I/O Attributes Applicability

Not all I/O attributes are applicable to all I/O standards.

Use the following tables to determine which I/O attributes you can modify for each of the following devices:

- IGLOOe, Fusion, ProASIC3L, and ProASIC3E
- IGLOO, IGLOO PLUS, and ProASIC3
- Axcelerator

# IGLOOe, ProASIC3L, ProASIC3E, SmartFusion and Fusion I/O Standards and I/O Attributes Applicability

The following table shows which I/O attributes apply to which I/O standards for IGLOOe, ProASIC3L, ProASIC3E, SmartFusion and Fusion devices. Not all I/O attributes are applicable for all I/O standards. Also, some attributes are preset and cannot be changed. Therefore, use the table below to determine which I/O attributes you can modify per I/O standard for your device.

Table 13 · I/O Standards and I/O Attributes Applicability for IGLOOe, ProASIC3L, ProASIC3E, SmartFusion and Fusion Devices

| I/O Standard | Output Drive | Slew | Resistor Pull | Schmitt_ trigger (input only | In_delay (input only) | Skew | Output Load | Use register | Hot_ Swappable |
|---|---|---|---|---|---|---|---|---|---|
| LVTTL | X | X | X | X | X | X | X | X | X |
| LVCMOS 3.3 | X | X | X | X | X | X | X | X | X |
| LVCMOS 2.5/5.0 | X | X | X | X | X | X | X | X | |
| LVCMOS 1.8 | X | X | X | X | X | X | X | X | X |
| LVCMOS 1.5 | X | X | X | X | X | X | X | X | X |
| LVCMOS 1.2 | X | X | X | X | X | X | X | X | X |
| PCI | | | | X | X | X | | X | |
| PCIX | | X | | X | X | X | | X | |
| GTL+ 3.3 | | | | | X | X | | X | X |
| GTL+ 2.5 | | | | | X | X | | X | X |
| GTL 3.3 | | | | | X | X | | X | X |

| I/O Standard | Output Drive | Slew | Resistor Pull | Schmitt_ trigger (input only | In_delay (input only) | Skew | Output Load | Use register | Hot_ Swappable |
|---|---|---|---|---|---|---|---|---|---|
| GTL 2.5 | | | | | X | X | | X | X |
| HSTL Class I and II | | | | | X | X | | X | X |
| SSTL2 Class I and II | | | | | X | X | | X | X |
| SSTL3 Class I and II | | | | | X | X | | X | X |
| LVDS | | | | | X | X | | X | X |
| LVPECL | | | | | X | X | | X | X |

Note: Note: Only ProASIC3L, IGLOOe, IGLOO PLUS, and IGLOO devices support LVCMOS12.

# IGLOO and ProASIC3 I/O Standards and Attributes Applicability

The following table shows which I/O attributes apply to which I/O standards for IGLOO and ProASIC3 devices. Not all I/O attributes are applicable for all I/O standards. Also, some attributes are preset and cannot be changed. Therefore, use the table below to determine which I/O attributes you can modify per I/O standard for your device.

Table 14 · I/O Standards and I/O Attributes Applicability for IGLOO and ProASIC3 Devices

| I/O Standard | Output Drive | Slew | Resistor Pull | Skew | Output Load | Use register |
|---|---|---|---|---|---|---|
| LVTTL | X | X | X | X | X | X |
| LVCMOS 3.3 | X | X | X | X | X | X |
| LVCMOS 2.5/5.0 | X | X | X | X | X | X |
| LVCMOS 1.8 | X | X | X | X | X | X |
| LVCMOS 1.5 | X | X | X | X | X | X |
| LVCMOS 1.2 | X | X | X | X | X | X |
| PCI | | | | X | | X |
| PCIX | | | | X | | X |
| LVDS | | | | X | | X |
| LVPECL | | | | X | | X |

Note: Note: Only ProASIC3L, IGLOO, IGLOOe, and IGLOO PLUS devices support LVCMOS12.

# Axcelerator I/O Standards and I/O Attributes Applicability

The following table shows which I/O attributes apply to which I/O standards for Axcelerator devices. Not all I/O attributes are applicable for all I/O standards. Also, some attributes are preset and cannot be changed. Therefore, use the table below to determine which I/O attributes you can modify per I/O standard for your device.

Table 15 · I/O Standards and I/O Attributes Applicability for Axcelerator Devices

| I/O Standard | Output Drive | Slew | Resistor Pull | In_delay (input only) | Output Load | Use register |
|---|---|---|---|---|---|---|
| LVTTL | X | X | X | X | X | X |
| LVCMOS 2.5/5.0 | | | X | X | X | X |
| LVCMOS 1.8 | | | X | X | X | X |
| LVCMOS 1.5 | | | X | X | X | X |
| PCI | | | | X | | X |
| PCIX | | | | X | | X |
| GTL+ 3.3 | | | | | | X |
| GTL+ 2.5 | | | | | | X |
| HSTL Class I and II | | | | | | X |
| SSTL2 Class I and II | | | | | | X |
| SSTL3 Class I and II | | | | | | X |
| LVDS | | | | | | X |
| LVPECL | | | | | | X |

# Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

## Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650. 318.8044**

## Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

## Actel Technical Support

Visit the Actel Customer Support website (http://www.actel.com/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

## Website

You can browse a variety of technical and non-technical information on Actel's home page, at http://www.actel.com/.

## Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

### Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is tech@actel.com.

## Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

**650.318.4460**
**800.262.1060**

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/company/contact/default.aspx.


**Actel is the leader in low-power and mixed-signal FPGAs and offers the most comprehensive portfolio of system and power management solutions. Power Matters. Learn more at http://www.actel.com .**

***Actel Corporation*** *• 2061 Stierlin Court • Mountain View, CA 94043 • USA*
*Phone 650.318.4200 • Fax 650.318.4600 • Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060*

***Actel Europe Ltd****. • River Court, Meadows Business Park • Station Approach, Blackwater • Camberley Surrey GU17 9AB • United Kingdom*
*Phone +44 (0) 1276 609 300 • Fax +44 (0) 1276 607 540*

***Actel Japan*** *• EXOS Ebisu Building 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
*Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • http://jp.actel.com*

***Actel Hong Kong*** *• Room 2107, China Resources Building • 26 Harbour Road • Wanchai • Hong Kong*
*Phone +852 2185 6460 • Fax +852 2185 6488 • www.actel.com.cn*

5-02-00046-15/11.10