

---

# Design Constraints for Libero Soc v10.0

## User's Guide

NOTE: Links and cross-references in this PDF file may point to external files and generate an error when clicked.

**View the online help included with software to enable all linked content.**



---

# Table of Contents

<b>Design Constraints</b> .....	<b>7</b>
<b>Families Supported</b> .....	<b>9</b>
Constraint Support by Family.....	10
Constraint Entry .....	13
Constraint File Format by Family .....	16
<b>Basic Concepts</b> .....	<b>17</b>
Region.....	19
Location.....	20
I/O Attributes .....	21
Assigning pins in Package Pins View .....	22
Editing I/O Attributes .....	23
Sorting Attributes.....	24
Formatting Rows and Columns.....	25
Manually Assigning Technologies to I/O Banks.....	28
Automatically Assigning Technologies to I/O Banks.....	31
Reserving Pins for Device Migration.....	32
Specifying an I/O Standard .....	34
<b>I/O Attributes</b> .....	<b>35</b>
I/O Attributes by Family or Device .....	36
Bank Name .....	37
Hold State .....	39
Hot Swappable.....	40
Input Delay .....	41
I/O Standard.....	42
I/O Threshold (or Output Level) .....	46
Locked.....	47
Macro Cell .....	48
Output Drive .....	49
Output Load .....	50
Output Level.....	51
Pin Number .....	52
Port Name .....	53
Resistor Pull.....	54
Schmitt Trigger.....	55
Skew .....	56
Slew .....	57
Use Register .....	58
Explicitly Reserved.....	59
I/O Bank Settings Dialog Box (IGLOO and ProASIC3 only).....	61

I/O Bank Settings Dialog Box.....	62
<b>Entering Constraints .....</b>	<b>64</b>
Importing Constraint Files .....	65
<b>Using GUI Tools .....</b>	<b>67</b>
MultiView Navigator (MVN).....	68
<b>Exporting Constraint Files .....</b>	<b>69</b>
<b>Constraints by Name: Timing.....</b>	<b>70</b>
Create Clock .....	71
Create Generated Clock .....	72
Set Clock Latency .....	74
Set False Path.....	77
Set Input Delay.....	78
Set Load on Output Port .....	79
Set Maximum Delay.....	80
Set Minimum Delay.....	81
Set Multicycle Path.....	82
Set Output Delay.....	83
Assign I/O to Pin .....	84
Assign I/O Macro to Location.....	85
Assign Macro to Region.....	86
Assign Net to Global Clock .....	87
Assign Net to Local Clock .....	88
Assign Net to Quadrant Clock.....	89
Assign Net to Region .....	90
Configure I/O Bank.....	91
Create Region .....	92
Delete Regions.....	93
Move Block.....	94
Move Region.....	95
Reserve Pins.....	96
Reset Attributes on an I/O to Default Settings .....	97
Reset an I/O Bank to Default Settings .....	98
Reset Net's Criticality to Default Level.....	99
Set Block Options.....	100
Set Net's Criticality .....	101
Set Port Block .....	102
Unassign I/O Macro from Location .....	103
Unassign Macro from Region .....	104
Unassign Macro(s) Driven by Net from Region .....	105
Unreserve Pins.....	106
<b>Constraints by Name: Netlist Optimization .....</b>	<b>107</b>
Delete Buffer Tree.....	108

Demote Global Net to Regular Net .....	109
Promote Regular Net to Global Net .....	110
Restore Buffer Tree.....	111
Set Preserve .....	112
<b>Constraints by File Format - SDC Command Reference .....</b>	<b>113</b>
About Synopsys Design Constraints (SDC) Files.....	114
SDC Syntax Conventions .....	115
create_clock.....	117
create_generated_clock.....	119
remove_clock_uncertainty .....	121
set_clock_latency.....	123
set_disable_timing .....	126
set_false_path.....	127
set_input_delay .....	128
set_load.....	130
set_max_delay (SDC).....	131
set_multicycle_path.....	135
set_output_delay.....	137
<b>Design Object Access Commands.....</b>	<b>139</b>
all_inputs .....	140
all_registers.....	141
all_registers.....	142
get_cells .....	143
get_clocks .....	144
get_pins.....	145
get_nets .....	146
get_ports .....	147
About Physical Design Constraint (PDC) Files.....	148
PDC Syntax Conventions .....	150
PDC Naming Conventions .....	152
assign_global_clock.....	154
assign_local_clock.....	155
assign_net_macros.....	157
assign_quadrant_clock .....	159
assign_region.....	161
define_region .....	162
delete_buffer_tree.....	165
dont_touch_buffer_tree.....	166
move_block.....	167
move_region .....	169
reserve .....	170
reset_floorplan .....	171
reset_io .....	172
reset_jobank.....	174

reset_net_critical .....	175
set_block_options .....	176
set_io (IGLOOe, Fusion, ProASIC3L, and ProASIC3E) .....	178
set_io (IGLOO PLUS) .....	184
set_io (IGLOO and ProASIC3).....	188
set_jobank (IGLOOe, IGLOO PLUS, Fusion, ProASIC3L, and ProASIC3E) .....	192
set_jobank (IGLOO and ProASIC3).....	195
set_location .....	197
set_multitile_location.....	198
set_net_critical .....	201
set_port_block.....	202
set_preserve .....	203
unassign_global_clock.....	204
unassign_local_clock.....	205
unassign_macro_from_region .....	206
unassign_net_macros.....	207
unassign_quadrant_clock .....	208
undefine_region .....	209
unreserve .....	210
<b>I/O Standards .....</b>	<b>211</b>
I/O Standards Table.....	212
I/O Standards Compatibility Matrix .....	214
IGLOOe, ProASIC3L, ProASIC3E, SmartFusion and Fusion I/O Standards Compatibility Matrix .....	215
IGLOO, IGLOO PLUS, and ProASIC3 I/O Standards Compatibility Matrix.....	217
I/O Standards and I/O Attributes Applicability.....	218
IGLOOe, ProASIC3L, ProASIC3E, SmartFusion and Fusion I/O Standards and I/O Attributes Applicability .....	219
IGLOO and ProASIC3 I/O Standards and Attributes Applicability .....	221
<b>Product Support .....</b>	<b>223</b>



---

# 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 Microsemi'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 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.

When we specify a family name, we refer to the [device family and all its derivatives](#), unless otherwise specified.

Table 1 - Constraint Support by Family

	IGLOO	SmartFusion and Fusion	ProASIC3
<b>Timing</b>			
<a href="#">Create a clock</a>	X	X	X
<a href="#">Create a generated clock</a>	X	X	X
<a href="#">Remove clock uncertainty</a>	X	X	X
<a href="#">Set clock latency</a>	X	X	X
<a href="#">Set clock uncertainty</a>	X	X	X
<a href="#">Set disable timing</a>	X	X	X
<a href="#">Set false path</a>	X	X	X
<a href="#">Set input delay</a>	X	X	X
<a href="#">Set load on output port</a>	X	X	X
<a href="#">Set maximum delay</a>	X	X	X
<a href="#">Set minimum delay</a>	X	X	X
<a href="#">Set multicycle path</a>	X	X	X
<a href="#">Set output delay</a>	X	X	X
<b>Physical Placement</b>			
<b>-Clocks</b>			
<a href="#">Assign Net to Global Clock</a>	X	X	X
<a href="#">Assign Net to Local Clock</a>	X	X	X
<a href="#">Assign Net to Quadrant Clock</a>	X	X	X
<b>-Regions</b>			
<a href="#">Assign Macro to Region</a>	X	X	X
<a href="#">Assign Net to Region</a>	X	X	X
<a href="#">Create Region</a>	X	X	X
<a href="#">Delete Regions</a>	X	X	X

	IGLOO	SmartFusion and Fusion	ProASIC3
<b>Timing</b>			
<a href="#">Move Region</a>	X	X	X
<a href="#">Unassign macro(s) driven by net</a>	X	X	X
<a href="#">Unassign Macro from Region</a>	X	X	X
<b>-I/Os</b>			
<a href="#">Assign I/O to pin</a>	X	X	X
<a href="#">Assign I/O Macro to Location</a>	X	X	X
<a href="#">Configure I/O Bank</a>	X	X	X
<a href="#">Reset attributes on I/O to default settings</a>	X	X	X
<a href="#">Reset I/O bank to default settings</a>	X	X	X
<a href="#">Reserve pins</a>	X	X	X
<a href="#">Unreserve pins</a>	X	X	X
<a href="#">Unassign I/O macro from location</a>	X	X	X
<b>-Block</b>			
<a href="#">Move Block</a>	X	X	X
<a href="#">Set port block</a>	X	X	X
<a href="#">Set Block Options</a>	X	X	X
<b>-Nets</b>			
<a href="#">Assign Net to Global Clock</a>	X	X	X
<a href="#">Assign Net to Local Clock</a>	X	X	X
<a href="#">Assign Net to Quadrant Clock</a>	X	X	X
<a href="#">Assign Net to Region</a>	X	X	X
<a href="#">Reset net's criticality to default level</a>			
<a href="#">Set Net's Criticality</a>			
<a href="#">Unassign macro(s) driven by net</a>	X	X	X
<b>Netlist Optimization</b>			
<a href="#">Delete buffer tree</a>	X	X	X

	IGLOO	SmartFusion and Fusion	ProASIC3
<b>Timing</b>			
<a href="#">Demote Global Net to Regular Net</a>	X	X	X
<a href="#">Promote regular net to global net</a>	X	X	X
<a href="#">Restore buffer tree</a>	X	X	X
<a href="#">Set preserve</a>	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	PDC	PIN	ChipPlanner	I/O Attribute Editor	PinEditor	Timer/SmartTime	Compile Options
<b>Timing</b>								
<a href="#">Create a clock</a>	X						X	
<a href="#">Create a generated clock</a>	X						X	
<a href="#">Remove clock uncertainty</a>	X						X	
<a href="#">Set clock latency</a>	X						X	
<a href="#">Set clock uncertainty</a>	X						X	
<a href="#">Set disable timing</a>	X						X	
<a href="#">Set false path</a>	X						X	
<a href="#">Set input delay</a>	X						X	
<a href="#">Set load on output port</a>	X				X	X	X	
<a href="#">Set maximum delay</a>	X						X	
<a href="#">Set minimum delay</a>	X						X	
<a href="#">Set multicycle path</a>	X						X	
<a href="#">Set output delay</a>	X						X	
<b>Physical Placement</b>								
<b>-Clocks</b>								
<a href="#">Assign Net to Global Clock</a>		X						
<a href="#">Assign Net to Local Clock</a>		X		X				
<a href="#">Assign Net to Quadrant Clock</a>		X		X				

<b>-Regions</b>							
<a href="#">Assign Macro to Region</a>		X		X			
<a href="#">Assign Net to Region</a>		X		X			
<a href="#">Create Region</a>		X		X			
<a href="#">Delete Regions</a>		X		X			
<a href="#">Move region</a>		X		X			
<a href="#">Unassign macro(s) driven by net</a>		X		X			
<a href="#">Unassign macro from region</a>		X		X			
<b>-I/Os</b>							
<a href="#">Assign I/O to pin</a>		X	X	X	X	X	
<a href="#">Assign I/O Macro to Location</a>		X		X			
<a href="#">Configure I/O Bank</a>		X		X		X	
<a href="#">Reset attributes on I/O to default settings</a>		X		X	X		
<a href="#">Reset I/O bank to default settings</a>		X		X	X		
<a href="#">Reserve pins</a>		X			X	X	
<a href="#">Unreserve pins</a>		X			X	X	
<a href="#">Unassign I/O macro from location</a>		X		X			
<b>-Blocks</b>							
<a href="#">Move Block</a>		X					
<a href="#">Set port block</a>		X		X			
<a href="#">Set Block Options</a>		X					X
<b>-Nets</b>							
<a href="#">Assign Net to Global Clock</a>		X					
<a href="#">Assign Net to Local Clock</a>		X		X			
<a href="#">Assign Net to Quadrant Clock</a>		X		X			
<a href="#">Assign Net to Region</a>		X		X			
<a href="#">Reset net's criticality to default level</a>		X					
<a href="#">Set Net's Criticality</a>		X					
<a href="#">Unassign macro(s) driven by net</a>		X		X			

Netlist Optimization							
<a href="#">Delete buffer tree</a>		<a href="#">X</a>					<a href="#">X</a>
<a href="#">Demote Global Net to Regular Net</a>		<a href="#">X</a>					<a href="#">X</a>
<a href="#">Promote regular net to global net</a>		<a href="#">X</a>					<a href="#">X</a>
<a href="#">Restore buffer tree</a>		<a href="#">X</a>					
<a href="#">Set preserve</a>		<a href="#">X</a>					

**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.

When we specify a family name, we refer to the [device family and all its derivatives](#), unless otherwise specified.

Table 3 - Constraint File Format by Family

Family	Timing	Physical Placement	Netlist Optimiziation
	SDC	PDC	PDC
IGLOO	X	X	
SmartFusion and Fusion	X	X	X
ProASIC3	X	X	

SDC– Synopsys Design Constraints

PDC – Physical Design 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.

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

### 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.

Best practice is to 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.

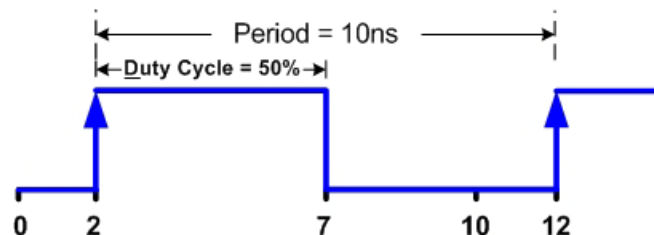
**First edge:** Indicates whether the first edge of the clock is rising or falling.

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

### Example 1:

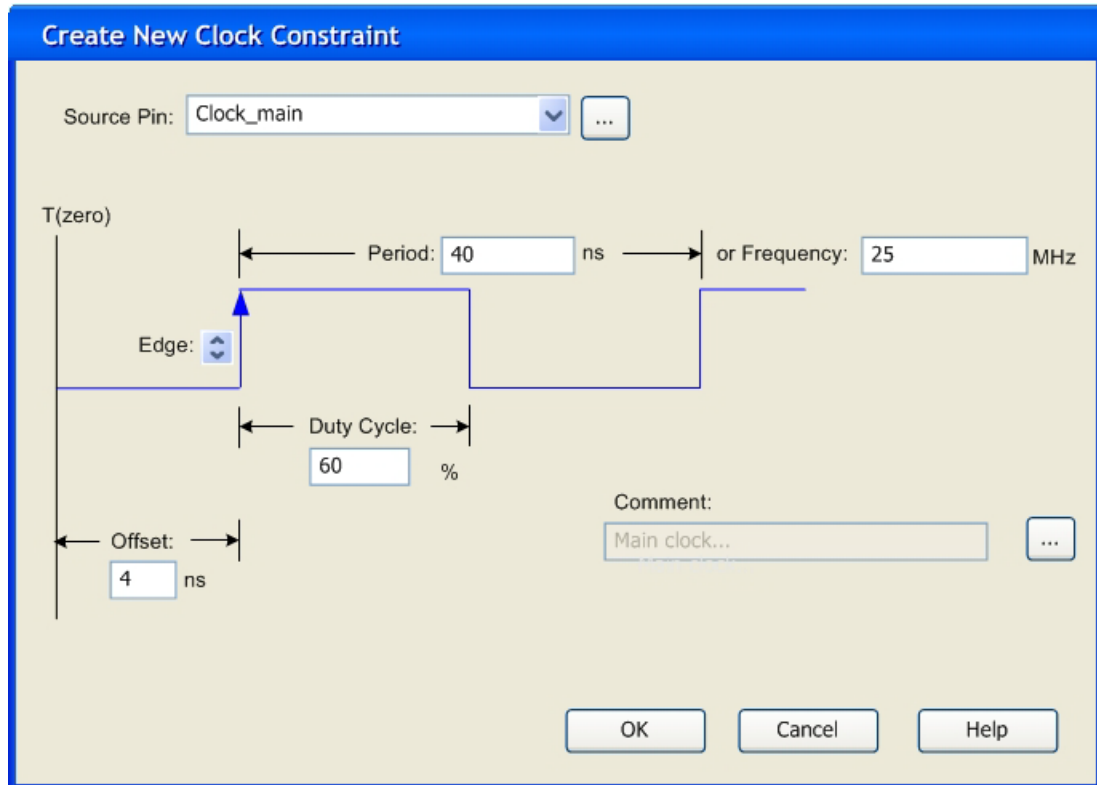
```
create_clock -period 10 -waveform {2 7}
```

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](#)

[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](#)

[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 [Welcome to I/O Attribute Editor](#), 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 SoC 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

## Assigning pins in Package Pins View

I/O Attribute Editor 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 (IGL00e, Fusion, ProASIC3L ProASIC3E). A device supports VREF if one or more of its I/O banks support VREF. IGL00 (excluding IGL00e) 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.

### **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](#)

## 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**.

### **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**.

**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**.

To unfreeze one or more frozen columns, from the **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.

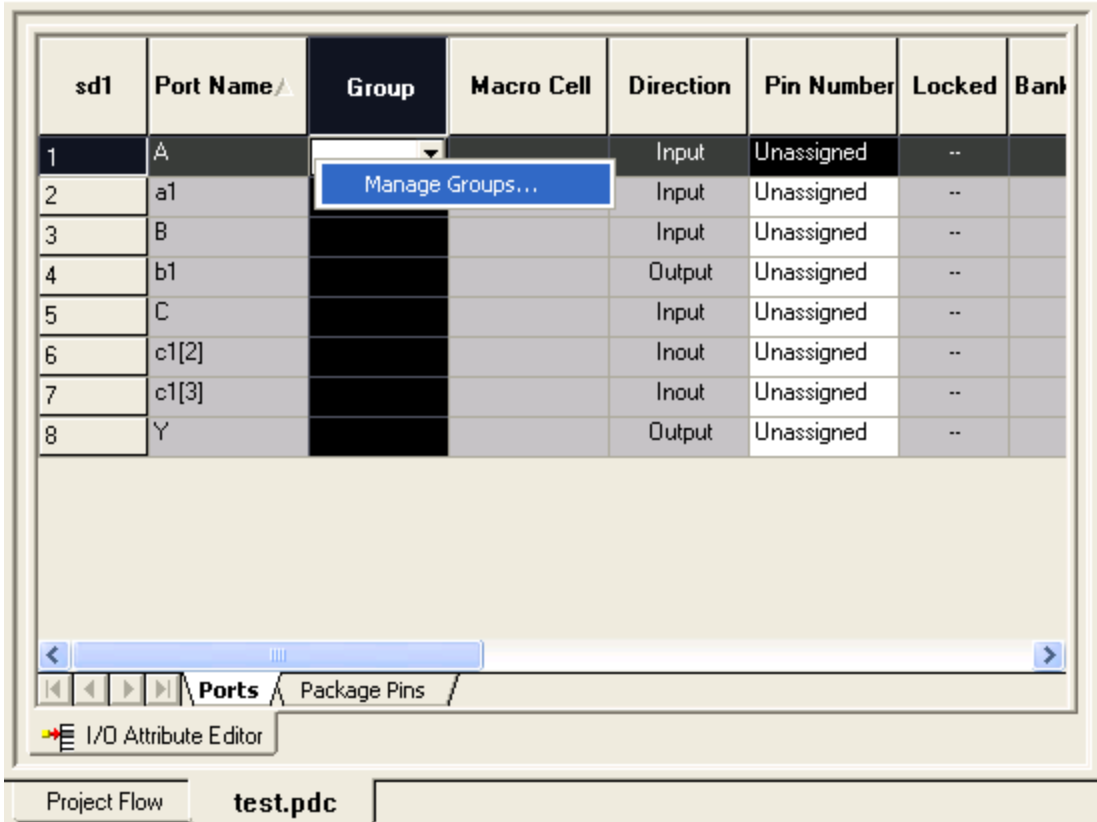


Figure 1 · Manage Groups Command on Right-click Menu

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

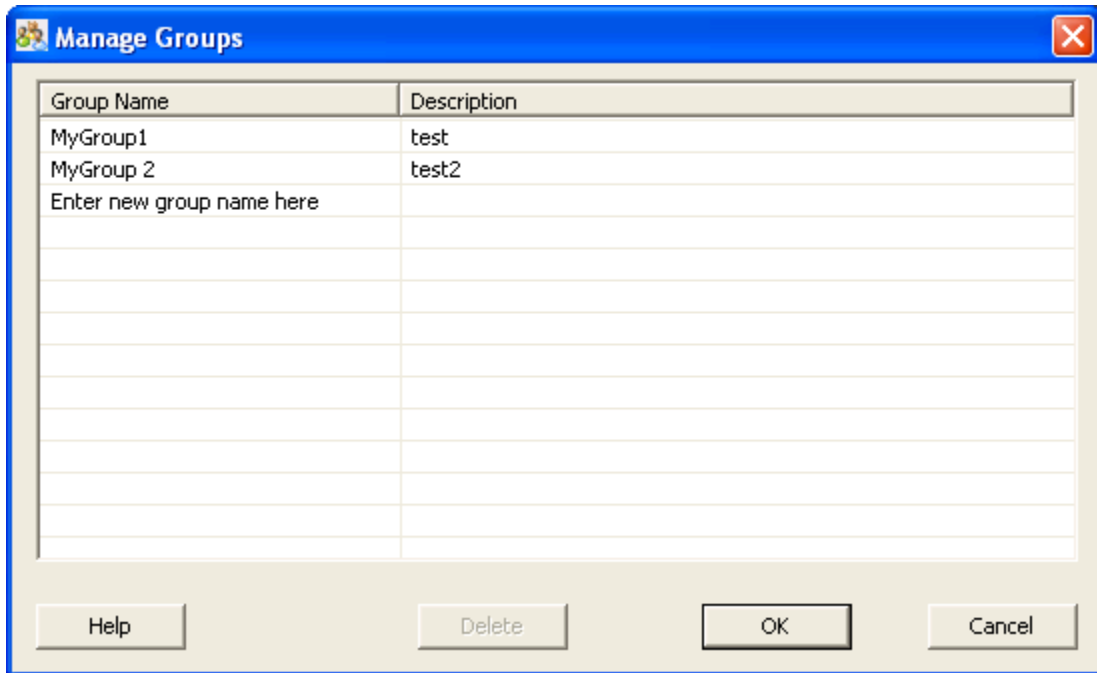


Figure 2 · Manage Groups Dialog Box

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

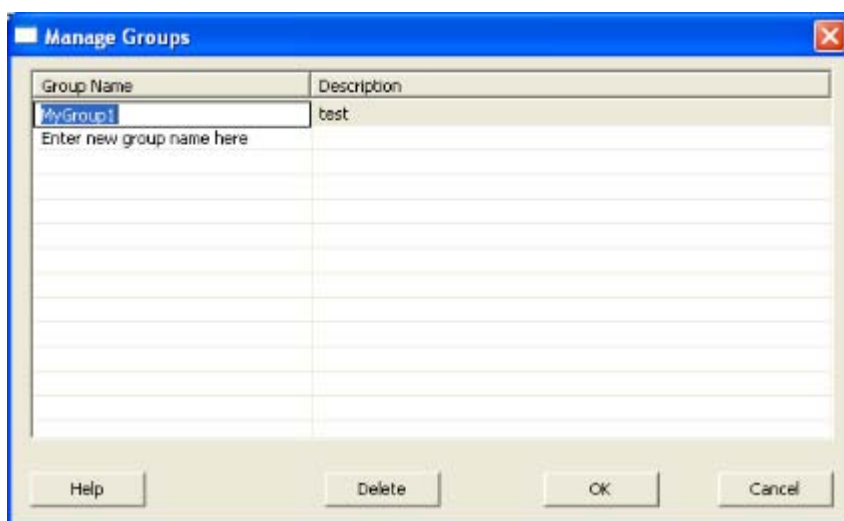
#### To create a new group:

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:



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, or ProASIC3 devices.

### *To assign technologies to I/O banks in IGLOOe, Fusion, ProASIC3L and ProASIC3E 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 LVTTTL 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. 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 your [Project Settings](#).
5. 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.

6. Click **OK**. Using PinEditor, proceed to assign I/Os with the same standards to the appropriate banks.

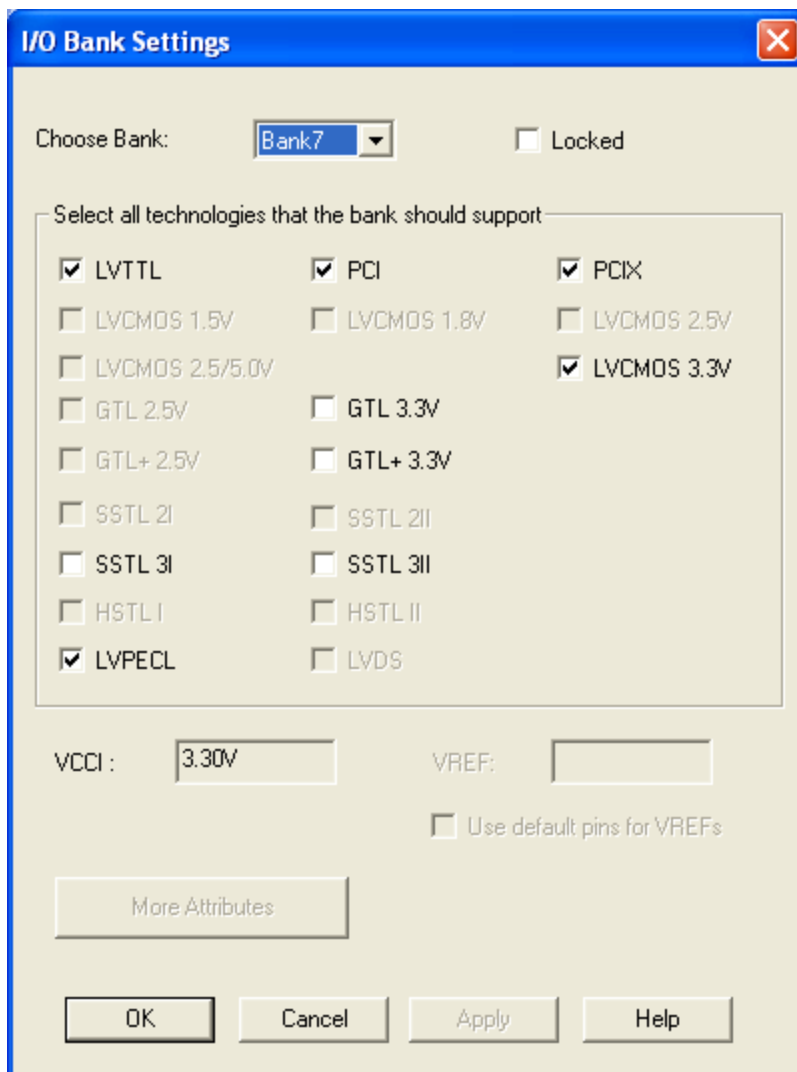


Figure 3 · I/O Bank Settings Dialog Box for IGLOOe, Fusion, ProASIC3L and ProASIC3E Devices

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. Microsemi SoC strongly recommends you use the defaults. VREF pins appear in red in ChipPlanner and are labeled VREF in PinEditor.

**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 LVTTTL also selects PCI, PCI-X, 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 your [Project Settings](#).
5. Click **OK**. Using PinEditor, proceed to assign I/Os with the same standards to the appropriate banks.

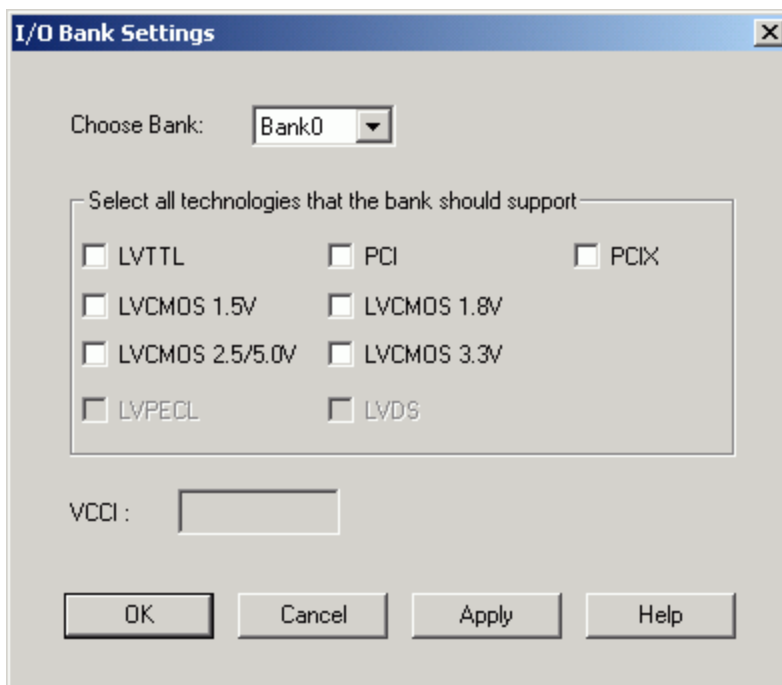


Figure 4 · I/O Bank Settings Dialog Box for IGLOO and ProASIC3 Devices

**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. 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.

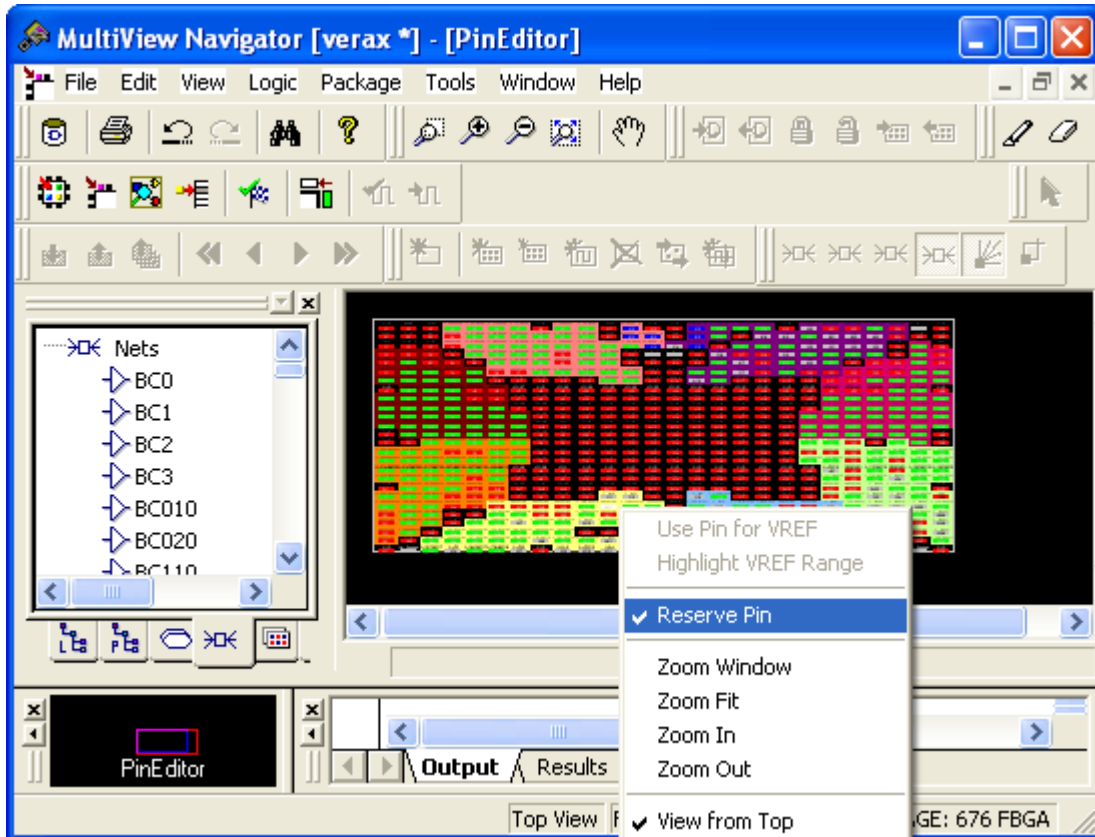


Figure 5 ·

Figure 6 · 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.



Figure 7 · Reserve Pins in I/O Attribute Editor

**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.

### 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, 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 Microsemi SoC family supports. The following table displays the attributes supported for each family.

Attribute	Family		
	IGLOO	SmartFusion and Fusion	ProASIC3
<a href="#">Bank Name</a>	X	X	X
<a href="#">I/O Standard</a>	X	X	X
<a href="#">I/O Threshold</a>	X, IGLOO PLUS only		
<a href="#">Output Drive</a>	X	X	X
<a href="#">Slew</a>	X	X	X
<a href="#">Resistor Pull</a>	X	X	X
<a href="#">Schmitt Trigger</a>	X, IGLOOe and IGLOO PLUS only	X	X, ProASIC3e and ProASIC3L only
<a href="#">Input Delay</a>	X, IGLOOe and IGLOO PLUS only	X	X, ProASIC3e and ProASIC3L only
<a href="#">Skew</a>	X	X	X
<a href="#">Output Load</a>	X	X	X
<a href="#">Use Register</a>	X	X	X
<a href="#">Hot Swappable</a>	X	X	X
<a href="#">Hold State</a>	X, IGLOO PLUS only		
<a href="#">User Reserved</a>	X	X	X, ProASIC3e and ProASIC3L only

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

## 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
SmartFusion	Yes
Fusion	Yes
ProASIC3	Yes

## Group

### Purpose

Indicates whether the port currently belongs to a group.

Families	Supported
IGLOO	Yes
SmartFusion	Yes
Fusion	Yes
ProASIC3	Yes

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

## 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

### 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

### 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.

## 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

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

IGLOO, ProASIC3, SmartFusion, and Fusion 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
<a href="#">CMOS</a>			
<a href="#">CUSTOM</a>			
<a href="#">GTL+</a>	IGLOOe only	X	ProASIC3E and ProASIC3L only
<a href="#">GTL 3.3 V</a>	IGLOOe only	X	ProASIC3E and ProASIC3L only
<a href="#">GTL 2.5 V</a>	IGLOOe only	X	ProASIC3E and ProASIC3L only
HSTL Class I	IGLOOe only	X	ProASIC3E and ProASIC3L only
<a href="#">HSTL Class II</a>	IGLOOe only	X	ProASIC3E and ProASIC3L only
<a href="#">LVCMOS 3.3 V</a>	IGLOOe only	X	ProASIC3E and ProASIC3L only
<a href="#">LVCMOS 2.5 V</a>	X	X	X
<a href="#">LVCMOS 2.5 V/5.0V</a>	IGLOOe only	X	X
<a href="#">LVCMOS 1.8 V</a>	X	X	X
<a href="#">LVCMOS 1.5</a>	X	X	X

I/O Standard	IGLOO	SmartFusion /Fusion	ProASIC3
<a href="#">V</a>			
<a href="#">LVCMOS 1.2 V</a>	X	X	ProASIC3 (A3PL), IGLOOe V2 only, IGLOO V2, and IGLOO PLUS only
<a href="#">LVDS</a>	IGLOO and IGLOO PLUS only		ProASIC3L only
<a href="#">LVPECL</a>	X	X	X
<a href="#">LVTTL/TTL</a>	X	X	X
<a href="#">PCI</a>	X	X	X
<a href="#">PCI-X 3.3 V</a>	X	X	X
<a href="#">SSTL2 Class I and II</a>	IGLOOe only	X	ProASIC3E and ProASIC3L only
<a href="#">SSTL3 Class I and II</a>	IGLOOe only	X	ProASIC3E and ProASIC3L only

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 Microsemi SoC web site ([www.actel.com](http://www.actel.com)) and check the datasheet for your device.

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

A low-power standard (JEDEC 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, ProASIC3, SmartFusion and Fusion 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 Microsemi SoC 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)

### **LVC MOS 3.3 V (Low-Voltage CMOS for 3.3 Volts)**

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

### **LVC MOS 2.5 V (Low-Voltage CMOS for 2.5 Volts)**

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

### **LVC MOS 2.5 V/5.0 V (Low-Voltage CMOS for 2.5 and 5.0 Volts)**

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

### **LVC MOS 1.8 V (Low-Voltage CMOS for 1.8 Volts)**

An extension of the LVC MOS 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.

### **LVC MOS 1.5 V (Low-Voltage CMOS for 1.5 volts)**

An extension of the LVC MOS 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.

### **LVC MOS 1.2 V (Low-Voltage CMOS for 1.2 volts)**

**Note:** Note: An extension of the LVC MOS 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).

### **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/ITTL (Low-Voltage Transistor-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 LVTTTL 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; Microsemi SoC 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

### 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

### 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.

## 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

## 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

### 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 LVTTTL 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

### 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

### Values

You can enter a capacitive 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  $V_{il}/V_{ih}$  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, LVTTTL, 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

### Values

, or

### Default value

LVTTTL

## 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

## 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

## 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

### 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.

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

## 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 Microsemi SoC 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

### 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

### 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 LVTTTL, all LVCMOS, or PCIX I/O standards. The other I/O standards have a preset slew value. For those devices that support additional slew values, Microsemi SoC 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.

## 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

### 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

## 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. Any banks not assigned I/O standards use the default standard selected in your [Project Settings](#).

### 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 LVTTTL 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, 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 LVTTTL 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) and ProASIC3E 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, or ProASIC3E 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 **SmartDesign 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.

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 LVTTTL.
- 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.

# Entering Constraints

---

You can enter design constraints in the following ways:

- **Importing constraint files:** You can import PDC or SDC constraint files.
- **Using constraint editor tools:** The constraint editor is a 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 PDC or other file syntax. .

For **IGLOO**, **ProASIC3**, **SmartFusion**, **Fusion**, use the tools within the MultiView Navigator:

- - - **Sets location and region assignments**
- - - **Sets the pin location constraints**
- - **I/O Attribute Editor - Sets I/O attributes**
- - **SmartTime Constraints Editor - Enables you to view and edit 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 .

### 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 .

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
Physical Design Constraint File	*.pdc	IGLOO, ProASIC3, SmartFusion, Fusion
Synopsys Constraint File	*.sdc	IGLOO, ProASIC3, SmartFusion, Fusion

### 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 .

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
SDC	*.sdc	IGLOO, ProASIC3, SmartFusion, Fusion
Physical Design Constraint **	*.pdc	IGLOO, Fusion, ProASIC3
Switching Activity Intermediate File/Format	*.saif	IGLOO, Fusion, ProASIC3
Value Change Dump file	*.vcd	IGLOO, Fusion, ProASIC3

(\*) 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 that enables you to view and edit physical constraints for the IGLOO, ProASIC3, SmartFusion and Fusion 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](#)

## 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
SDC	*.sdc	IGLOO, ProASIC3, SmartFusion, Fusion
Physical Design Constraint	*.pdc	IGLOO, ProASIC3, SmartFusion, Fusion

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	SmartTime
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

## 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](#)
- SmartTime - [Specifying Clock Constraint](#)

## See Also

[Constraint Entry](#)

[create\\_clock](#) (SDC)

[Clock](#) Definition

[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

### 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

## 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](#)

[set\\_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

### 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)

[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

Families	SDC	Timer/SmartTime
ProASIC3	X	X

## 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

## Purpose

Use this constraint to 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.

## 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](#)

**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	Timer/SmartTime
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

### 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](#)
- SmartTime - [Specifying False Path Constraint](#)

### See Also

[Constraint Entry](#)

[set\\_false\\_path](#) (SDC)

[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

### 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	I/O Attribute Editor
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	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](#)
- I/O Attribute Editor –
- 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\\_assign](#)

[Editing I/O Attributes](#)

## 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	SmartTime
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

### 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](#)
- SmartTime – [Specifying Maximum Delay Constraint](#)

### See Also

[Constraint Entry](#)

[set\\_max\\_delay](#) (SDC)

# 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	SmartTime
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

## 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	SmartTime
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

## 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)

[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	SmartTime
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

## 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](#)

## 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	PinEditor
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

### Purpose

Use this constraint to set the location of a pin.

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..

### 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`
- PinEditor (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)

[pin\\_assign](#)

[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	ChipPlanner
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	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](#)
- ChipPlanner -

## 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

### 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.

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.

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`
- ChipPlanner - Assigning a macro to a region

### See Also

[Constraint Entry](#)

[assign\\_region](#) (PDC)

[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
IGLOO	X
SmartFusion	X
Fusion	X
ProASIC3	X

### 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.

### 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`

### See Also

[Constraint Entry](#)

[assign\\_global\\_clock](#) (PDC)

# 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
IGLOO	X
SmartFusion	X
Fusion	X
ProASIC3	X

## 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

### See Also

[Constraint Entry](#)

[assign\\_local\\_clock](#) (PDC)

[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

## 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	ChipPlanner
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

## 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`
- ChipPlanner - Assigning a Net to a Region

### See Also

[Constraint Entry](#)

[assign\\_net\\_macros](#) (PDC)

# 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

## 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, SmartFusion and Fusion devices 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.

## 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	ChipPlanner
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

### 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.

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.

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`
- ChipPlanner - `Creating_regions`

### See Also

[Constraint Entry](#)

[define\\_region](#) (PDC)

*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

## 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
IGLOO	X
SmartFusion	X
Fusion	X
ProASIC3	X

### 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

### 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

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

### 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.

### 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

### Purpose

Use this constraint to reset an I/O bank's technology to the default setting. The default is specified in **Project > Project Settings**.

### 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	

### 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
IGLOO	X
SmartFusion	X
Fusion	X
ProASIC3	X

## 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		

## 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](#)

### See Also

[Constraint Entry](#)

[set\\_net\\_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
IGLOO	X
SmartFusion	X
Fusion	X
ProASIC3	X

### 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
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	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

## See Also

[set\\_location](#)

[set\\_multitile\\_location](#)

MultiView Navigator User's Guide: [Assigning Logic to Locations](#)

# 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

## 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

### 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

### 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 View](#) Assigning pins in Package Pins view

### See Also

[reserve](#)

[Constraint Entry](#)

---

# 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
IGLOO	X
SmartFusion	X
Fusion	X
ProASIC3	X

### 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

### See Also

[Constraint Entry](#)

[Netlist 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	Compile Options
IGLOO	X	X
SmartFusion	X	X
Fusion	X	X
ProASIC3	X	X

## 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
- Compile Options - -demote\_globals <value>

## See Also

[Constraint Entry](#)

[Netlist Optimization Constraints](#)

[unassign\\_global\\_clock](#) (PDC)

## 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

### 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`
- Compile Options - `-promote_globals <value>`

### See Also

[Constraint Entry](#)

[Netlist Optimization Constraints](#)

[assign\\_global\\_clock](#) (PDC)

*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	

### 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`

### 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		

### 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. Microsemi tools use a subset of the SDC format to capture supported timing constraints. 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 Microsemi tools and third-party EDA tools.

Command	Action
<a href="#">create_clock</a>	Creates a clock and defines its characteristics
<a href="#">create_generated_clock</a>	Creates an internally generated clock and defines its characteristics
<a href="#">remove_clock_uncertainty</a>	Removes a clock-to-clock uncertainty from the current timing scenario.
<a href="#">set_clock_latency</a>	Defines the delay between an external clock source and the definition pin of a clock within SmartTime
<a href="#">set_clock_uncertainty</a>	Defines the timing uncertainty between two clock waveforms or maximum skew
<a href="#">set_false_path</a>	Identifies paths that are to be considered false and excluded from the timing analysis
<a href="#">set_input_delay</a>	Defines the arrival time of an input relative to a clock
<a href="#">set_load</a>	Sets the load to a specified value on a specified port
<a href="#">set_max_delay</a>	Specifies the maximum delay for the timing paths
<a href="#">set_min_delay</a>	Specifies the minimum delay for the timing paths
<a href="#">set_multicycle_path</a>	Defines a path that takes multiple clock cycles
<a href="#">set_output_delay</a>	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 <i>Courier New</i> typeface.
<i>variable</i>	Variables appear in blue, italic <i>Courier New</i> typeface. You must substitute an appropriate value for the variable.
[-argument <i>value</i> ]	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, Microsemi 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, SmartFusion and Fusion

### 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}]
```

## Microsemi Implementation Specifics

- The -waveform in SDC accepts waveforms with multiple edges within a period. In Microsemi design implementation, only two waveforms are accepted.
- SDC accepts defining a clock on many sources using a single command. In Microsemi 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 Microsemi 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 and Fusion

### 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  $\frac{3}{4}$  of the period at the reference pin clk.

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

## Microsemi Implementation Specifics

- SDC accepts either `-multiply_by` or `-divide_by` option. In Microsemi 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 Microsemi 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 Microsemi 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 and Fusion

### 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 and Fusion

### 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 }
```

## Microsemi Implementation Specifics

SDC accepts a list of clocks to `-set_clock_latency`. In Microsemi 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 and Fusion

## 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
```

## Microsemi 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

### 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.

### Microsemi 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 and Fusion

### 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
```

### Microsemi Implementation Specifics

- SDC accepts multiple `-through` options in a single constraint to specify paths that traverse multiple points in the design. In Microsemi 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 and Fusion

### 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}
```

## Microsemi Implementation Specifics

In SDC, the -clock is an optional argument that allows you to set input delay for combinational designs. Microsemi 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

*capitance*

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

### 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 {01 02}
set_load 25 [get_ports out]
```

### Microsemi Implementation Specifics

- In SDC, you can use the set\_load command to specify capacitance value on nets. Microsemi 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 and Fusion

### 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*]
```

## Microsemi 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 and Fusion

### 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*]
```

## Microsemi 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 and Fusion

### 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}]
```

## Microsemi Implementation Specifics

- SDC allows multiple priority management on the multiple cycle path constraint depending on the scope of the object accessors. In Microsemi 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 and Fusion

### 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}
```

### Microsemi Implementation Specifics

- In SDC, the -clock is an optional argument that allows you to set the output delay for combinational designs. Microsemi 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	<a href="#">get_cells</a>
Clock	<a href="#">get_clocks</a>
Net	<a href="#">get_nets</a>
Port	<a href="#">get_ports</a>
Pin	<a href="#">get_pins</a>
Input ports	<a href="#">all_inputs</a>
Output ports	<a href="#">all_outputs</a>
Registers	<a href="#">all_registers</a>

## 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 and Fusion

### Exceptions

- None

### Example

```
set_max_delay -from [all_inputs] -to [get_clocks ck1]
```

### Microsemi 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 and Fusion

### 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]
```

### Microsemi 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 and Fusion

### 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]
```

### Microsemi 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

### 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}]
```

### Microsemi 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 SmartTime on which a clock constraint has been set.

### Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion

### 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 data1] -to \  
[get_clocks ck1]
```

### Microsemi 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

### Exceptions

None

### Example

```
create_clock -period 10 [get_pins clock_gen/reg2:Q]
```

### Microsemi 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

### 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_clock -name mainCLK -per 2.5 [get_nets {cknet}]
```

### Microsemi 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

### Exceptions

None

### Example

```
create_clock -period 10[get_ports CK1]
```

### Microsemi 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. Any constraint that you can enter using the PinEditor in MVN or ChipPlanner tool, you can also use in a PDC file.

**Note:** Designer supports the following PDC commands.

Command	Action
<a href="#">assign_global_clock</a>	Assigns regular nets to global clock networks by promoting the net using a CLKINT macro
<a href="#">assign_local_clock</a>	Assigns regular nets to LocalClock regions
<a href="#">assign_net_macros</a>	Assigns the macros connected to a net to a specified defined region
<a href="#">assign_quadrant_clock</a>	Assigns regular nets to a specific quadrant clock region
<a href="#">assign_region</a>	Assigns macros to a pre-specified region
<a href="#">define_region</a>	Defines either a rectangular or rectilinear region
<a href="#">delete_buffer_tree</a>	Removes all buffers and inverters from a given net
<a href="#">dont_touch_buffer_tree</a>	Restores all buffers and inverters that were removed from a given net with the delete_buffer_tree command
<a href="#">move_block</a>	Moves only the block core (COMB, SEQ) of the specified instance (I/Os or PLLs) to the specified location on the chip
<a href="#">move_region</a>	Moves a region to new coordinates
<a href="#">reset_floorplan</a>	Deletes all defined regions. Placed macros are not affected.
<a href="#">reset_io</a>	Resets all attributes on a macro to the default values
<a href="#">reset_iobank</a>	Resets an I/O banks technology to the default technology
<a href="#">reset_net_critical</a>	Resets net criticality to default level
<a href="#">set_io</a>	Sets the attributes of an I/O
<a href="#">set_iobank</a>	Specifies the I/O bank's technology and sets the VREF pins for the specified banks
<a href="#">set_location</a>	Places a given logic instance at a particular location
<a href="#">set_block_options</a>	Overrides the compile option for either a specific block or an instance of a block
<a href="#">set_multitile_location</a>	Assigns specified two-tile and four-tile macros to specified locations on the chip

Command	Action
<a href="#">set_port_block</a>	Sets properties on a port in the Block flow
<a href="#">set_preserve</a>	Preserves instances before compile so that instances are not combined
<a href="#">set_net_critical</a>	Sets net criticality, which is issued to influence placement and routing in favor of performance
<a href="#">set_reserve</a>	Reserves the specified pins in the design
<a href="#">set_unreserve</a>	Resets the specified pins in the design that were previously reserved
<a href="#">unassign_global_clock</a>	Assigns clock nets to regular nets
<a href="#">unassign_local_clock</a>	Unassigns the specified user-defined net from a LocalClock or QuadrantClock region
<a href="#">unassign_macro_from_region</a>	Unassigns macros from a specified region, if they are assigned to that region
<a href="#">unassign_net_macros</a>	Unassigns macros connected to a specified net from a defined region
<a href="#">unassign_quadrant_clock</a>	Unassigns the specified net from a QuadrantClock region
<a href="#">undefine_region</a>	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 -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 <i>value</i> ] [ <i>variable</i> ]+	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` 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` 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, ProASIC3, SmartFusion and Fusion 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	AV:B1	A\B1
Net name in a netlist	Net1:/net1	Net1\:/net1
Net name in MVN or a compile report	Net1V:/net1	Net1\V:/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 and 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 \(IGLOO, Fusion, and ProASIC3\)](#)

[unassign\\_global\\_clock](#)

[PDC Syntax Conventions](#)

[PDC Naming Conventions](#)

## assign\_local\_clock

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: <ul style="list-style-type: none"> <li>• A QuadrantClock region</li> <li>• A LocalClock region driven by a clock rib located on the top or bottom of the chip</li> </ul>

*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\_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 (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:

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)

*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 and Fusion

### 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 keyinlintZ0Z_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 and Fusion

### 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:

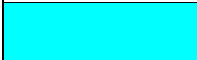


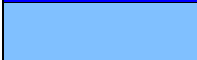






Region Type Value	Description
Empty	Empty regions cannot contain macros.
Exclusive	Only contains macros assigned to the region.
Inclusive	Can contain macros both assigned and unassigned to the region.

*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
	65535
	33023
	8421631
	9568200

Color	Decimal Value
	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.

Over-constrain Placement Value	Description
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  $M \times N$  Inclusive placement region would shrink to  $(M-2) \times (N-2)$ ). On the other hand, the prohibited region for external nets of Exclusive and Empty Region types would expand to  $(M+2) \times (N+2)$ .

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

## 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.

```
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

[dont\\_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, SmartFusion, Fusion and ProASIC3 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 and Fusion

### 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: 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 and Fusion

### 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 and Fusion

### 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 and Fusion

### 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 and Fusion

### 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 devices, I/O banks are numbered 0-7 (bank0, bank1,.. bank7).

### Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

### 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

IGLOO, ProASIC3, SmartFusion and Fusion

### 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](#)

[PDC Naming 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.
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.

Value	Description
discard	Discards any routing from the block, even if there are no conflicts.

## Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

## 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
LVTTTL	(Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3 V applications. It uses an LVTTTL input buffer and a push-pull output buffer.
LVCNOS33	(Low-Voltage CMOS for 3.3 Volts) An extension of the LVCNOS standard (JESD 8-5) used for general-purpose 3.3 V applications.
LVCNOS25	(Low-Voltage CMOS for 2.5 Volts) An extension of the LVCNOS standard (JESD 8-5) used for general-purpose 2.5 V applications.
LVCNOS25_50	(Low-Voltage CMOS for 2.5 and 5.0 Volts) An extension of the LVCNOS standard (JESD 8-5) used for general-purpose 2.5 V and 5.0V applications.
LVCNOS18	(Low-Voltage CMOS for 1.8 Volts) An extension of the LVCNOS 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.
LVCNOS15	(Low-Voltage CMOS for 1.5 volts) An extension of the LVCNOS

Value	Description
	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.
LVC MOS12	(Low-Voltage CMOS for 1.2 volts) An extension of the LVC MOS 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 LVTTTL 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.
HSTLI	(High-Speed Transceiver Logic) A general-purpose, high-speed 1.5 V bus standard (EIA/JESD 8-6). It has four classes; Microsemi SoC 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; Microsemi SoC 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; Microsemi SoC 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; Microsemi SoC

Value	Description
	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), and ProASIC3 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, SmartFusion and Fusion 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.

Not all I/O standards have a selectable slew. For ProASIC3 devices, this attribute is only available for LVTTTL, 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
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

Value	Description
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	
LVTTTL	X	X	X	X	X	X	X	High Low

I/O Standard	Output							Slew
	2	4	6	8	12	16	24	
LVC MOS33	X	X	X	X	X	X	X	High Low
LVC MOS25	X	X	X	X	X	X	X	High Low
LVC MOS25_50	X	X	X	X	X	X	X	High Low
LVC MOS18	x	x	x	x	-	-	-	High Low
LVC MOS15	x	x	-	-	-	-	-	High Low
LVC MOS12	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
LVTTTL	(Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3 V applications. It uses an LVTTTL 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.
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.

Value	Description
LVC MOS12	(Low-Voltage CMOS for 1.2 volts) An extension of the LVC MOS 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	Descripti T
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, SmartFusion and Fusion 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 LVTTTL, LVC MOS33, LVC MOS25\_50, LVC MOS18, LVC MOS15, 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
-------	-------------

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	
LVTTTL	X	X	X	X	X	X	High Low
LVC MOS33	X	X	X	X	X	X	High Low
LVC MOS25	X	X	X	X	X	X	High Low
LVC MOS18	X	X	X	X	-	-	High Low
LVC MOS15	X	X	-	-	-	-	High Low
LVC MOS12	x	-	-	-	-	-	High Low
PCI	-	-	-	-	-	-	High Low
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
LVTTTL	(Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3 V applications. It uses an LVTTTL input buffer and a push-pull output buffer.
LVC MOS33	(Low-Voltage CMOS for 3.3 Volts) An extension of the LVC MOS standard (JESD 8-5) used for general-purpose 3.3 V applications.
LVC MOS25_50	(Low-Voltage CMOS for 2.5 and 5.0 Volts) An extension of the LVC MOS standard (JESD 8-5) used for general-purpose 2.5 V and 5.0V applications.
LVC MOS18	(Low-Voltage CMOS for 1.8 Volts) An extension of the LVC MOS 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.
LVC MOS15	(Low-Voltage CMOS for 1.5 volts) An extension of the LVC MOS 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.
LVC MOS12	(Low-Voltage CMOS for 1.2 volts) An extension of the LVC MOS

Value	Description
	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 MHz and 66 MHz PCI bus applications. It uses an LVTTTL 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

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

`-slew value`

Sets the output slew rate. Slew control affects only the falling edges for some families. For ProASIC3, IGLOO, SmartFusion and Fusion 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 LVTTTL, 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	
LVTTTL	X	X	X	X	X	X	High Low
LVCOS33	X	X	X	X	X	X	High Low
LVCOS25_50	X	X	X	X	X	-	High Low
LVCOS18	X	X	X	X	-	-	High Low
LVCOS15	x	x	-	-	-	-	High Low
LVCOS12	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\_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)
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 Microsemi SoC web site ([www.actel.com](http://www.actel.com)) 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
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 Microsemi SoC web site ([www.actel.com](http://www.actel.com)) 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\_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

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 DFI1E1C0 DFI0E1P1 DFI1E0P0	DFI1E1C1 DFI0E1C0 DFI1E0P1 DFI0E0P0	DFI0E1C1 DFI1E0C0 DFI0E0P1	DFI1E0C1 DFI0E0C0 DFI1E1P0	DFI0E0C1 DFI1E1P1 DFI0E1P0
--	--	----------------------------------	----------------------------------	----------------------------------

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

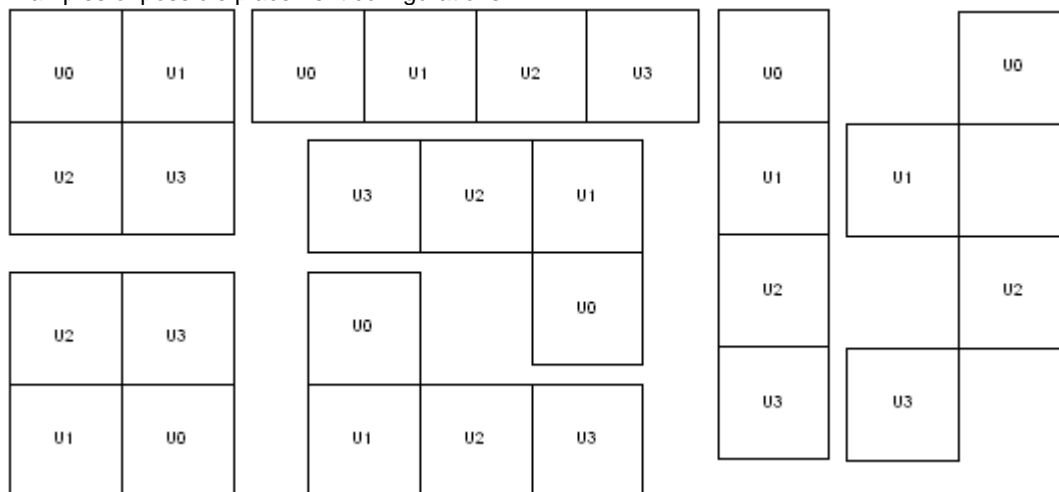
## Description

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_tileeff/U2 0 2 } \  
-tile { multi_tileeff/U3 0 3 } -fixed yes
```

As a result of this command, the four-tile macro placement looks like this:

U3
10,13
U2
10,12
U1
10,11
U0
10,10

The second example shows you how to configure a two-tile macro:

```
set_multitile_location multi_tileeff -location {10 10} \  
-tile { multi_tileeff/U0 0 0 } \  
-tile { multi_tileeff/U1 1 0 }
```

As a result of this command, the two-tile macro placement looks like this:

U0	U1
10,10	11,1

### 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_critical criticality_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 (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

IGLOO, ProASIC3, SmartFusion and Fusion

### 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 -name portName -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 and Fusion

### 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

### 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 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 reset\_n from the local clock region:

```
unassign_local_clock -net reset_n
```

### See Also

[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 and Fusion

### 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 and Fusion

### 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 keyinlintZ0Z_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 and Fusion

### 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 and Fusion

### Exceptions

None

### Examples

```
unreserve -pinname "F2"  
unreserve -pinname "F2 B4 B3"  
unreserve -pinname "124 63"
```

### See Also

[reserve](#)

[PDC Syntax Conventions](#)

[PDC Naming Conventions](#)

---

# 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 6 · I/O Standards

I/O Standard	IGLOO	SmartFusion and Fusion	ProASIC3
CMOS			
CUSTOM			
GTLP25	IGLOOe only	X	ProASIC3E and ProASIC3L only
GTLP33	IGLOOe only	X	ProASIC3E and ProASIC3L only
GTL33	IGLOOe only	X	ProASIC3E and ProASIC3L only
GTL25	IGLOOe only	X	ProASIC3E and ProASIC3L only
HSTL1	IGLOOe only	X	ProASIC3E and ProASIC3L only
HSTLII	IGLOOe only	X	ProASIC3E and ProASIC3L only
LVC MOS33	X	X	X
LVC MOS25	IGLOOe only	X	X
LVC MOS25_50	X	X	X
LVC MOS18	X	X	X
LVC MOS15	X	X	X
LVC MOS12	X		ProASIC3L only
LV TTL	X	X	X
TTL	X	X	X
PCI	X	X	X
PCIX	X	X	X
SSTL2I and SSTL2II	IGLOOe only	X	ProASIC3E and ProASIC3L only
SSTL3I and SSTL3II	IGLOOe only	X	ProASIC3E and ProASIC3L only

**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

## 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 7 · I/O Standards Compatibility for IGLOOe, ProASIC3L, ProASIC3E, SmartFusion and Fusion Devices

I/O Standard	LVTTTL	LVC MOS 3.3	LVC MOS 2.5	LVC MOS 2.5/5.0	LVC MOS 1.8	LVC MOS 1.5	LVC MOS 1.2	PCI, PCIX	GTL+ 3.3	GTL+ 2.5	GTL 3.3	GTL 2.5	HSTL Class I	HSTL Class II	SSTL2 Class I and II	SSTL3 Class I and II	LVDS	LVPECL
LVTTTL	X	X						X	X		X					X		X
LVC MOS 3.3	X	X						X	X		X					X		X
LVC MOS 2.5			X	X						X		X			X		X	
LVC MOS 2.5/5.0			X	X						X		X			X		X	
LVC MOS 1.8					X													
LVC MOS 1.5						X							X			X		
LVC MOS 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												
HSTL Class II						X												
SSTL2 Class I and II;			X	X													X	

I/O Standard	LVTTTL	LVC MOS 3.3	LVC MOS 2.5	LVC MOS 2.5/5.0	LVC MOS 1.8	LVC MOS 1.5	LVC MOS 1.2	PCI, PCIX	GTL+ 3.3	GTL+ 2.5	GTL 3.3	GTL 2.5	HSTL Class I	HSTL Class II	SSTL2 Class I and II	SSTL3 Class I and II	LVDS	LVPECL
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 LVC MOS12.**

## 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 8 - I/O Standards Compatibility for IGLOO, IGLOO PLUS, and ProASIC3 Devices

I/O Standard	LVTTTL	LVC MOS 3.3	LVC MOS 2.5/5.0	LVC MOS 1.8	LVC MOS 1.5	LVC MOS 1.2	PCI, PCIX	LVDS	LVPECL
LVTTTL	X	X					X		X
LVC MOS 3.3	X	X					X		X
LVC MOS 2.5/5.0			X					X	
LVC MOS 1.8				X					
LVC MOS 1.5					X				
LVC MOS 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 LVC MOS12.

## 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, SmartFusion, Fusion, ProASIC3L, and ProASIC3E
- IGLOO, IGLOO PLUS, and ProASIC3

## 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 9 · 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
LVTTTL	X	X	X	X	X	X	X	X	X
LVC MOS 3.3	X	X	X	X	X	X	X	X	X
LVC MOS 2.5/5.0	X	X	X	X	X	X	X	X	
LVC MOS 1.8	X	X	X	X	X	X	X	X	X
LVC MOS 1.5	X	X	X	X	X	X	X	X	X
LVC MOS 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
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

I/O Standard	Output Drive	Slew	Resistor Pull	Schmitt_trigger (input only)	In_delay (input only)	Skew	Output Load	Use register	Hot_Swappable
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 10 · 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
LVTTTL	X	X	X	X	X	X
LVC MOS 3.3	X	X	X	X	X	X
LVC MOS 2.5/5.0	X	X	X	X	X	X
LVC MOS 1.8	X	X	X	X	X	X
LVC MOS 1.5	X	X	X	X	X	X
LVC MOS 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 LVC MOS12.



---

# Product Support

---

The Microsemi SoC Products Group backs its products with various support services including a Customer Technical Support Center and Non-Technical Customer Service. This appendix contains information about contacting the SoC Products Group and using these support services.

## Contacting the Customer Technical Support Center

Microsemi 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.

### Technical Support

Microsemi customers can receive technical support on Microsemi SoC products by calling Technical Support Hotline anytime Monday through Friday. Customers also have the option to interactively submit and track cases online at My Cases or submit questions through email anytime during the week.

Web: [www.actel.com/mycases](http://www.actel.com/mycases)

Phone (North America): 1.800.262.1060

Phone (International): +1 650.318.4460

Email: [soc\\_tech@microsemi.com](mailto:soc_tech@microsemi.com)

### ITAR Technical Support

Microsemi customers can receive ITAR technical support on Microsemi SoC products by calling ITAR Technical Support Hotline: Monday through Friday, from 9 AM to 6 PM Pacific Time. Customers also have the option to interactively submit and track cases online at My Cases or submit questions through email anytime during the week.

Web: [www.actel.com/mycases](http://www.actel.com/mycases)

Phone (North America): 1.888.988.ITAR

Phone (International): +1 650.318.4900

Email: [soc\\_tech\\_itar@microsemi.com](mailto:soc_tech_itar@microsemi.com)

## Non-Technical Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

Microsemi's customer service representatives are available Monday through Friday, from 8 AM to 5 PM Pacific Time, to answer non-technical questions.

Phone: +1 650.318.2470



**Microsemi Corporate Headquarters**  
One Enterprise Drive, Aliso Viejo CA 92656  
Within the USA: (800) 713-4113  
Outside the USA: (949) 221-7100  
Fax: (949) 756-0308 · [www.microsemi.com](http://www.microsemi.com)

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at [www.microsemi.com](http://www.microsemi.com).

© 2011 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.