UG0688
PDC Commands User Guide
SmartFusion2, IGLOO2, RTG4
Libero SoC v12.0

a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

5-02-00688-2.0/12.18

Table of Contents

Introduction

In the FPGA design world, constraint files are as important as design source files. Physical design constraints (PDC) are used to constrain the I/O attributes, placement, and routing during the physical layout phase.

You can enter PDC commands manually using the Libero SoC Text Editor. PDC commands can also be generated using Libero SoC's interactive tools.The I/O Attribute Editor is the interactive tool for making I/O attribute changes, and the Chip Planner is the interactive tool for making floorplanning changes. When changes are made in the I/O Attribute Editor or the Chip Planner, the PDC file(s) are updated to reflect the changes. These PDC commands can be used as part of a script file to constrain the Place and Route step of your design.

Supported Families

This User Guide covers the PDC commands applicable to SmarfFusion2, IGLOO2, and RTG4 families.

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` *`value`*`]`
`[`*`variable`*`]+`
	Optional arguments begin and end with a square bracket with one exception: if the square bracket is followed by a plus sign (+), 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: 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

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.

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
```

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.

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?

Using PDC Commands

When writing PDC commands, follow these rules:

- Always use the macro name as it appears in the netlist.
- 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.
- 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
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*
Net name in a netlist	Net1/:net1	Net1\\/:net1

When exporting PDC commands, the software always exports names using the PDC rules described in this section.

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 (SmartFusion2, IGLOO2, and RTG4)	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.)

1 – I/O PDC commands

I/O PDC commands are used to set and reset I/O standards, voltages values and attributes.

set_iobank

PDC command; sets the input/output supply voltage (vcci) and the input reference voltage (vref) for the specified I/O bank.

DDRIO banks have a dedicated vref pin and you do not need to set any pin on these banks. (See the device datasheet to see which banks are of type DDRIO.)

Diff I/Os do not need a vref pin.

```
set_iobank bankname \
     [-vcci vcci_voltage]\
     [-vref vref_voltage]\
     [-fixed value]\
     [-vrefpins value]\
     [-updateiostd 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, LVPECL
2.5 V	LVCMOS 2.5, SSTL2 (Class I and II), LVDS, BUSLVDS, MLVDS, MINILVDS, RSDS
1.8 V	LVCMOS 1.8, LPDDRI, LPDDRII, SSTL18I
1.5 V	LVCMOS 1.5, SSTL 1.5 (Class I and II), HSTL (Class I and II)
1.2 V	LVCMOS 1.2

-vref vref_voltage

Sets the input reference voltage. You can enter one of the following values:

Vref Voltage	**Compatible Standards**
1.25 V	SSTL2 (Class I and II)
1.0 V	SSTL18 (Class I and II), LPDDR (Class I and II)
0.75 V	SSTL15 (Class I and II), 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. 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.

-updateiostd 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	If there are I/O's placed on the bank, we keep the placement and change the host to one which is compatible with this bank setting. Check the I/O Attributes to see the one used by the tool.
no	If there are I/O's placed and locked on the bank, the command will fail. If they are placed I/Os they will be unplaced.

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

"reset_io"

"reset_iobank"

reset_io

PDC command; 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 wild card characters in port names:

Wild card	What It Does
\	Interprets the next character as a non-special character
?	Matches any single character
*	Matches any string

-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.

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

"set_io (SmartFusion2 and IGLOO2)"
"set_io (RTG4 only)"

reset_iobank

PDC command; resets an I/O bank's technology to the default technology

```
reset_iobank bankname
```

Arguments

bankname

Specifies the I/O bank to be reset to the default technology. I/O banks are numbered 0-7 (bank0, bank1,. bank7).

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

"set_iobank"

reserve

PDC command; 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.

Exceptions

None

Examples

```
reserve -pinname "F2"
reserve -pinname "F2 B4 B3"
reserve -pinname "124 17"
```

See Also

"unreserve"

set_io (SmartFusion2 and IGLOO2)

PDC command; sets the attributes of an I/O.

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. There are three I/O Bank types available in SmartFusion2 and IGLOO2: MSIOD, MSIO and DDRIO.

```
set_io portname\
    [-iostd value]\
    [-pre_emphasis value]\
    [-lpe value]\
    [-ff_io_state value]\
    [-out_drive value]\
    [-slew value]\
    [-res_pull value]\
    [-schmitt_trigger value]\
    [-input_delay value]\
    [-odt_static value]\
    [-odt_imp value]\
    [-ff_io_avail value]\
    [-register value]\
    [-in_reg value]\
    [-out_reg value]\
    [-en_reg value]\
```

Arguments

portname

Specifies the portname of the I/O macro.

-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 a list of supported I/Os by Bank type.

MSIOD	MSIO	DDRIO
-	LVTTL	-
-	LVCMOS33	-
-	PCI	-
-	LVPECL (Input ONLY)	-
-	LVDS33	-
LVCMOS12	LVCMOS12	LVCMOS12
LVCMOS15	LVCMOS15	LVCMOS15
LVCMOS18	LVCMOS18	LVCMOS18
LVCMOS25	LVCMOS25	LVCMOS25
SSTL2I	SSTL2I	SSTL2I (DDR1)
-	STL2II	SSTL2II (DDR1)
SSTL18I	SSTL18I	SSTL18I (DDR2)
-	SSTL18II	SSTL18II (DDR2)
HSTLI	HSTLI	HSTLI
-	-	HSTLII
-	-	SSTL15I (DDR3)
-	-	SSTL15II (DDR3)
-	-	LPDDRI
-	-	LPDDRII
LVDS	LVDS	-
RSDS	RSDS	-

MINILVDS	MINILVDS	-
BUSLVDS (Input ONLY)	BUSLVDS	-
MLVDS (Input ONLY)	MLVDS	-

I/O standards support for single and differential I/Os is shown in the table below.

Value	**Single**	**Differential**	**Description**
LVTTL	X	-	(Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer.
LVCMOS33	X	-	(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	X	-	(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	X	-	(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	X	-	(Low-Voltage CMOS for 1.5 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.5 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer.
LVCMOS12	X	-	(Low-Voltage CMOS for 1.2 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.2 V applications.
LVDS		X	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).
LVDS33		X	LVDS for 3.3 V
BUSLVDS		X	2.5 V BUSLVDS
MLVDS		X	
MINILVDS		X	
RSDS		X	
LVPECL (only for inputs)		X	PECL is another differential I/O standard. It requires that one data bit is carried through two signal lines; therefore, two pins are needed per input or output. It also requires an external resistor termination. The voltage swing between these two signal lines is approximately 850mV. When the power supply is +3.3 V, it is commonly referred to as low-voltage PECL (LVPECL).

PCI			(Peripheral Component Interface) Specifies support for both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5V-compliant for most families.
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	X	X	(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. It requires a differential amplifier input buffer and a push-pull output buffer.
HSTLII	X	X	(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. It requires a differential amplifier input buffer and a push-pull output buffer.
SSTL2I	X	X	(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 supports both. It requires a differential amplifier input buffer and a push-pull output buffer.
SSTL2II	X	X	See SSTL2I above.
SSTL15I	X	X	(Stub Series Terminated Logic for 1.5 V) A general-purpose 1.5 V memory bus standard (JESD8-9). It has two classes; Microsemi SoC supports both. It requires a differential amplifier input buffer and a push-pull output buffer
SSTL15II	X	X	See SSTL15I
SSTL18II	X	X	(Stub Series Terminated Logic for 1.8 V) A general-purpose 1.8 V memory bus standard (JESD8-9). It has two classes; Microsemi SoC supports both. It requires a differential amplifier input buffer and a push-pull output buffer

-pre_emphasis value

The pre-emphasis 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. Possible values are shown in the table below. The output buffer has a programmable slew rate for both high-to-low and low-to-high transitions. The low rate is incompatible with 3.3 V PCI requirements.

Value	**Description**
NONE	Sets to none (default)
MIN	Sets to minimum

MEDIUM	Sets to medium
MAX	Sets to maximum

-lpe value

Sets the state at which your device exits from Low Power mode. Possible values are shown in the table below.

Value	**Description**
OFF	Default; no LPE set
Wake_on_Change	Exits from Low Power mode on change
Wake_on_0	Exits from Low Power mode on 0
Wake_on_1	Exits from Low Power mode on 1

-ff_io_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 FF_IO_STATE attributes. When you set this attribute to LAST_VALUE, the I/O remains in the same state in which it was functioning before the device went into Flash*Freeze mode. Possible values are shown in the table below.

Value	**Description**
TRISTATE	Sets the I/O to tristate (default).
LAST_VALUE	Preserves the previous state of the I/O.

-out_drive value

Sets the strength of the output buffer to 2, 4, 6, 8, 10, 12, 16, or 20 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. The table below lists acceptable values.

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
10	Sets the output drive strength to 10mA
12	Sets the output drive strength to 12mA
16	Sets the output drive strength to 16mA
20	Sets the output drive strength to 20mA

-slew value

Sets the output slew rate. Slew control affects only the falling edges for some 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.

See the Slew and Out_drive Settings table under Exceptions in this topic. The table below shows the acceptable values for the -slew attribute.

Value	Description
SLOW	Sets the I/O slew to slow
MEDIUM	Sets the I/O slew to medium
MEDIUM_FAST	Sets the I/O slew to medium fast
FAST	Sets the I/O slew to fast

-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

-input_delay value

Specifies whether this I/O has an input delay. You can specify an input delay between 0 and 63. The input delay is not a delay value but rather a selection from 0 to 63. 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 -input_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
...	...
63	Sets the input delay to 63

-odt_static value

On-die termination (ODT) is the technology where the termination resistor for impedance matching in transmission lines is located inside a semiconductor chip instead of on a printed circuit board. Possible value are listed in the table below.

Value	Description
on	Yes, the termination resistor for impedance matching is located inside the chip
off	No, the termination resistor is on the printed circuit board

-odt_imp value

On-die termination (ODT) is the technology where the termination resistor for impedance matching in transmission lines is located inside a semiconductor chip instead of on a printed circuit board.

Port Configuration (PC) bits are static configuration bits set during programming to configure the IO(s) as per your choice. See your device datasheet for a full range of possible values.

-ff_io_availvalue

Indicates the I/O is available in Flash*Freeze mode. The table below lists possible values.

Value	Description
yes	I/O is available in Flash*Freeze mode
no	Default; I/O is unavailable in Flash*Freeze mode

-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. 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

-in_reg value

Specifies whether the input register will be combined into the I/O. The -register option must be set to yes to be enable -in_reg. If in_reg is set to yes, the combiner combines the register into the I/O module if possible. This is off by default. The following table shows the acceptable values for the -in_reg attribute:

Value	Description
yes	Input register combining is allowed on this I/O
no	Input register combining is not allowed on this I/O

-out_reg value

Specifies whether the output register will be combined into the I/O. The -register option must be set to yes to enable -out_reg. If -out_reg is set to yes, the combiner combines the register into the I/O module if possible. This is off by default. The following table shows the acceptable values for the -out_reg attribute:

Value	Description
yes	Output register combining is allowed on this I/O
no	Output register combining is not allowed on this I/O

-en_reg value

Specifies whether the enable register will be combined into the I/O. The -register option must be set to yes to enable -en_reg. If -en_reg is set to yes, the combiner combines the register into the I/O module if possible. This is off by default. The following table shows the acceptable values for the -en_reg attribute:

Value	Description
yes	Enable register combining is allowed on this I/O
no	Enable register combining is not allowed on this I/O

Examples

```
set_io IO_in\[2\] -iostd LVCMOS25 \
     -slew slow \
     -schmitt_trigger off \
     -input_delay off \
     -ff_io_avail no \
```

See Also

"reset_io"

set_io (RTG4 only)

PDC command; sets the attributes of an I/O for RTG4 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. There are three I/O Bank types available in RTG4: MSIOD, MSIO and DDRIO.

```
set_io portname\
   [-direction input | output]\
   [-iostd value]\
   [-pre_emphasis value]\
   [-lpe value]\
   [-ff_io_state value]\
   [-out_drive value]\
   [-out_load value]\
   [-slew value]\
   [-res_pull value]\
   [-schmitt_trigger value]\
   [-input_delay value]\
   [-odt_static value]\
   [-odt_imp value]\
   [-ff_io_avail value]\
   [-register value]\
   [-in_reg value]\
   [-out_reg value]\
   [-en_reg value]
```

Arguments

portname

Specifies the portname of the I/O macro.

-direction value

Specifies the direction of the I/O ports. Valid values are input, output, inout.

-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 a list of supported I/Os by Bank type.

MSIOD	MSIO	DDRIO
-	LVTTL	-
-	LVCMOS33	-
-	PCI	-
-	LVPECL (Input ONLY)	-
-	LVDS33	-
LVCMOS12	LVCMOS12	LVCMOS12
LVCMOS15	LVCMOS15	LVCMOS15
LVCMOS18	LVCMOS18	LVCMOS18
LVCMOS25	LVCMOS25	LVCMOS25
SSTL2I	SSTL2I	SSTL2I (DDR1)
SSTL2II	STL2II	SSTL2II (DDR1)
SSTL18I	SSTL18I	SSTL18I (DDR2)
SSTL18II	SSTL18II	SSTL18II (DDR2)
HSTLI	HSTLI	HSTLI
-	-	HSTLII
-	-	SSTL15I (DDR3) Only for IOs used by FDDR
-	-	SSTL15II (DDR3) Only for I/OS used by FDDR
-	-	LPDDRI
-	-	LPDDRII
LVDS	LVDS	-
RSDS	RSDS	-
MINILVDS	MINILVDS	-
BUSLVDS (Input ONLY)	BUSLVDS	-
MLVDS (Input ONLY)	MLVDS	-

I/O standards support for single and differential I/Os is shown in the table below.

Value	Single	Differential	Description
LVTTL	X	-	(Low-Voltage TTL) A general purpose standard (EIA/ JESDSA) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer.

LVCMOS33	X	-	(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	X	-	(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	X	-	(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	X	-	(Low-Voltage CMOS for 1.5 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.5 V applications. It uses a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer.
LVCMOS12	X	-	(Low-Voltage CMOS for 1.2 volts) An extension of the LVCMOS standard (JESD 8-5) used for general-purpose 1.2 V applications. This I/O standard is supported only in ProASIC3L and the IGLOO family of devices.
LVDS		X	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).
LVDS33		X	LVDS for 3.3 V
BUSLVDS		X	2.5 V BUSLVDS
MLVDS		X	
MINILVDS		X	
RSDS		X	
LVPECL (only for inputs)		X	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).
HSTLI	X	X	High-Speed Transceiver Logic Class I. A general-purpose, high-speed 1.5 V bus standard (EIA/JESD 8-6). It has four classes; Microsemi SoC supports Class I and Class II. It requires a differential amplifier input buffer and a push-pull output buffer.
HSTLII	X	X	High-Speed Transceiver Logic Class II. A general-purpose, high-speed 1.5 V bus standard (EIA/JESD 8-6). It has four classes; Microsemi SoC supports Class I and Class II. It requires a differential amplifier input buffer and a push-pull output buffer.

HSTL18I	X	X	High-Speed Transceiver Logic 1.8 V Class I. A general-purpose, high-speed 1.8 V bus. It has four classes; Microsemi SoC supports Class I and Class II. It requires a differential amplifier input buffer and a push-pull output buffer.
HSTL18II	X	X	High-Speed Transceiver Logic 1.8 V Class II. A general-purpose, high-speed 1.8 V bus. It has four classes; Microsemi SoC supports Class I and Class II. It requires a differential amplifier input buffer and a push-pull output buffer.
SSTL2I	X	X	(Stub Series Terminated Logic for 2.5 V) A general-purpose 2.5 V memory bus standard (JESD8-9). It has two classes: Class I and Class II; Microsemi SoC supports both. It requires a differential amplifier input buffer and a push-pull output buffer.
SSTL2II	X	X	See SSTL2I above.
SSTL15I	X	X	Stub Series Terminated Logic for 1.5 V Class I. A general-purpose 1.5 V memory bus standard (JESD8-9). It has two classes: Class I and Class II; Microsemi supports both. It requires a differential amplifier input buffer and a push-pull output buffer.
SSTL15II	X	X	Stub Series Terminated Logic for 1.5 V Class II. See SSTL15I above.
SSTL18I	X	X	Stub Series Terminated Logic for 1.8 V Class I. A general-purpose 1.8 V memory bus standard (JESD8-9). It has two classes; Microsemi SoC supports both. It requires a differential amplifier input buffer and a push-pull output buffer.
SSTL18II	X	X	Stub Series Terminated Logic for 1.8 V Class II. A general-purpose 1.8 V memory bus standard (JESD8-9). It has two classes; Microsemi SoC supports both. It requires a differential amplifier input buffer and a push-pull output buffer.
LPDDRI	X	X	
LPDDRII	X	X	

-pre_emphasis value

The pre-emphasis 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. Possible values are shown in the table below. The output buffer has a programmable slew rate for both high-to-low and low-to-high transitions.

Value	Description	Applicable to I/O Standards
NONE	Sets to none (default)	LVDS, RSDS
MIN	Sets to minimum	LVDS, RSDS
MEDIUM	Sets to medium	RSDS only
MAX	Sets to maximum	LVDS, RSDS

-lpe value

Sets the state at which your device exits from Low Power mode. Possible values are shown in the table below.

Value	Description
OFF	Default; no LPE set
Wake_on_Change	Exits from Low Power mode on change
Wake_on_0	Exits from Low Power mode on 0
Wake_on_1	Exits from Low Power mode on 1

-ff_io_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 FF_IO_STATE attribute. When you set this attribute to LAST_VALUE, the I/O remains in the same state in which it was functioning before the device went into Flash*Freeze mode. Possible values are shown in the table below.

Value	Description
TRISTATE	Sets the I/O to tristate (default).
LAST_VALUE	Preserves the previous state of the I/O.

-out_drive value

Sets the strength of the output buffer to 2, 4, 6, 8, 10, 12, 16, or 20 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 and I/O Bank Type -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. The table below lists acceptable values.

Value (mA)	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
10	Sets the output drive strength to 10mA
12	Sets the output drive strength to 12mA
16	Sets the output drive strength to 16mA
20	Sets the output drive strength to 20mA

I/O Standard	User -set Valid Output Drive Values (mA) Per I/O Bank Type			Valid Output Drive Value for Die
	MSIO	MSIOD	DDRIO	

LVTTL	2			2
	4			4
	8			8
	12			12
	16	-		16
LVCMOS33	2			2
	4			4
	8			8
	12			12
	16			16
LVCMOS12	2	2	2	2
	4	4	4	4
		6	6	6
LVCMOS15	2	2	2	2
	4	4	4	4
	6	6	6	6
	8		8	8
			10	10
			12	12
LVCMOS18	2	2	2	2
	4	4	4	4
	6	6	6	6
	8	8	8	8
	10		10	10
	12		12	12
			16	16

-out_load value

Sets the output load (in pF) of output signals.

-slew value

Sets the output slew rate. Slew control affects only the falling edges for some 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.

The table below lists the acceptable values for the -slew attribute.

Value	**Description**	**I/O Standard**	**IO Bank Type**
SLOW	Sets the I/O slew to slow	LVCMOS12 LVCMOS15 LVCMOS18	MSIO, MSIOD, DDRIO

MEDIUM	Sets the I/O slew to medium	LVCMOS12 LVCMOS15 LVCMOS18	DDRIO
FAST	Sets the I/O slew to fast	LVCMOS12 LVCMOS15	DDRIO

-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 for different I/O Standard and I/O Bank combinations:

Value	I/O Standard	I/O Bank Type	Description
up	LVTTL, LVCMOS33 PCI	MSIO	Includes a weak resistor for pull-up of the input buffer
	LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25	MSIO/MSIOD/ DDRIO	
down	LVTTL, LVCMOS33 PCI	MSIO	Includes a weak resistor for pull-down of the input buffer
	LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25	MSIO/MSIOD/ DDRIO	
none	LVTTL, LVCMOS33 PCI	MSIO	Does not include a weak resistor (Default value)
	LVCMOS12, LVCMOS15, LVCMOS18, LVCMOS25	MSIO/MSIOD/ DDRIO	

-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 (Default value)

The applicable valid values are dependent on the I/O Standard and the I/O Bank Type.

I/O Standard	I/O Bank Type		
	MSIO	MSIOD	DDRIO
LVTTL	Off On	N/A	N/A
LVCMOS33	Off On	N/A	N/A
PCI	Off On	N/A	N/A
LVCMOS12	Off On	Off On	Off On
LVCMOS15	Off On	Off On	Off On

LVCMOS18	Off On	Off On	Off On
LVCMOS25	Off On	Off On	Off On

-input_delay value

Specifies whether this I/O has an input delay. You can specify an input delay between 0 and 63. The input delay is not an absolute delay value but rather a selection from 0 to 63. 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 -input_delay attribute:

Value	Description
off	This I/O does not have an input delay (Default value)
0	Sets the input delay to 0
1	Sets the input delay to 1
2	Sets the input delay to 2
...	...
63	Sets the input delay to 63

-odt_static value

On-die termination (ODT) is the technology where the termination resistor for impedance matching in transmission lines is located inside a semiconductor chip instead of on a printed circuit board. Possible value are listed in the table below.

Value	Description
on	Yes, the termination resistor for impedance matching is located inside the chip
off	No, the termination resistor is on the printed circuit board (Default value)

The valid value for each I/O Standard and I/O Bank Type combination is listed in the table below.

I/O Standard	I/O Bank Type		
	MSIO	MSIOD	DDRIO
LVPECL	Off On	N/A	N/A
LVDS33	Off On	N/A	N/A
SSTL18I (DDR2)	Off On	Off On	Off On
SSTL18II (DDR2)	Off On	Off On	Off On
HSTL18I	Off On	Off On	Off On
HSTL18II	N/A	N/A	Off On
HSTLI	Off	Off	Off On
HSTLII	Off	Off	Off On
SSTL15I (DDR3)	N/A	N/A	Off On
SSTL15II (DDR3)	N/A	N/A	Off On
LPDDRI	N/A	N/A	Off On
LPDDRII	N/A	N/A	Off On

LVDS	Off On	Off On	N/A
RSDS	Off On	Off On	N/A
MINILVDS	Off On	Off On	N/A
BUSLVDS	Off On	Off On	N/A
MLVDS	Off On	Off On	N/A

-odt_imp value

On-die termination (ODT) is the technology where the termination resistor for impedance matching in transmission lines is located inside a semiconductor chip instead of on a printed circuit board. The valid value for each I/O Standard and I/O Bank type is listed in the table below. When the value for an I/O standard is not listed, the impedance value is fixed for the specific I/O standard and is not user-selectable.

I/O Standard	**I/O Bank Type**		
	MSIO	**MSIOD**	**DDRIO**
SSTL18I (DDR2)	50 75 150	50 75 150	50 75 150
SSTL18II (DDR2)	50 75 150	50 75 150	50 75 150
HSTL18I	50 75 150	50 75 150	50 75 150
HSTL18II	-	-	50 75 150
LPDDRI	-	-	50 75 150
LPDDRII	-	-	50 75 150
SSTL15I (DDR3)	-	-	20 30 40 60 120
SSTL15II (DDR3)	-	-	20 30 40 60 120

Port Configuration (PC) bits are static configuration bits set during programming to configure the IO(s) as per your choice. See your device datasheet for a full range of possible values.

-ff_io_avail value

Indicates the I/O is available in Flash*Freeze mode. The table below lists possible values.

Value	**Description**
yes	I/O is available in Flash*Freeze mode
no	Default; I/O is unavailable in Flash*Freeze mode

-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. 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

-in_reg value

Specifies whether the input register will be combined into the I/O. The -register option must be set to yes to be enable -in_reg. If in_reg is set to yes, the combiner combines the register into the I/O module if possible. This is off by default. The following table shows the acceptable values for the -in_reg attribute:

Value	Description
yes	Input register combining is allowed on this I/O
no	Input register combining is not allowed on this I/O

-out_reg value

Specifies whether the output register will be combined into the I/O. The -register option must be set to yes to enable -out_reg. If -out_reg is set to yes, the combiner combines the register into the I/O module if possible. This is off by default. The following table shows the acceptable values for the -out_reg attribute:

Value	Description
yes	Output register combining is allowed on this I/O
no	Output register combining is not allowed on this I/O

-en_reg value

Specifies whether the enable register will be combined into the I/O. The -register option must be set to yes to enable -en_reg. If -en_reg is set to yes, the combiner combines the register into the I/O module if possible. This is off by default. The following table shows the acceptable values for the -en_reg attribute:

Value	Description
yes	Enable register combining is allowed on this I/O
no	Enable register combining is not allowed on this I/O

Examples

```
set_io IO_in\[2\] -iostd LVCMOS25 \
     -slew slow \
     -schmitt_trigger off \
     -input_delay off \
     -ff_io_avail no
```

See Also

"reset_io"

unreserve

PDC command; 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.

Exceptions

None

Examples

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

See Also

"reserve"

2 – Netlist Attributes PDC Commands

Netlist Attributes PDC Commands are used to set netlist-specific constraints. These commands are placed in a Compile Netlist Constraint (*.ndc) file and used by the Libero SoC Compile engine to optimize the post-synthesis netlist.

set_mitigation (RTG4 devices only)

This command sets the mitigation option on a per-instance basis for RTG4 devices. For the Enhanced Constraint flow, import this NDC command as a Netlist Attributes constraint file (*.ndc) and associate it with Synthesis in the **Constraint Manager**.

For the Classic Constraint Flow, save the command as a PDC file (*.pdc) and import the *.pdc file into the Project (**Design Flow Window > Floorplan Constraints > Import Files**). After import, associate the file with Compile (Right-click the PDC file and select **Use for Compile**).

Note this NDC/PDC command overrides the project-wide global setting for mitigation option (**Project Settings > Analysis Operating Conditions > Enable Single Event Transient Mitigation**). The global mitigation setting is project-wide and applies to ALL instances in the design that the mitigation option is valid. If you want to override the global setting for certain specific instances in the design (and allow the global setting to remain valid on all other instances), use the set_mitigation command for the specific instances because this command sets the mitigation option on a per-instance basis.

```
set_mitigation  -inst_name <instance_name>   -mitigated  <value>
```

Arguments

-inst_name <instance_name>

Specifies the name of the instance in the netlist to set the mitigation option. A hierarchical instance name is allowed. The wildcard character "*" in the instance name is also supported. When the mitigation option is set on an instance that contains design elements that can have the mitigation option, all the design elements within that instance are set with this option. This applies to MACC blocks, µSRAM blocks, LSRAM blocks, I/O Flip-Flops, and regular Flip-Flops within that instance.

-mitigated <value>

Sets the mitigation value for the instance. Acceptable values for this argument: are listed in the table below.

Value	**Description**
Yes | on | true | 1	The mitigation option is enabled on this instance. “Yes”, “On”, “True” are case-insensitive.
No | off | false | 0	The mitigation option is disabled on this instance. No, Off, False are case-insensitive.

Examples

This example sets the mitigation option on the instance blk1/* globally in the netlist:

```
set_mitigation -inst_name blk1/* -mitigated yes
```

After the first command above, to set a different mitigation value on a specific instance, e.g. instance FF_1 inside the blk1 instance, use the <-mitigated No> argument as follows:

```
set_mitigation -inst_name blk1/FF_1 -mitigated No
```

Note: The last set_mitigation command overrides a previous set_mitigation command, if there is a conflict.

Return Value

Returns "0" on success and "1" on failure.

set_ioff

This command specifies whether or not a register is combined with an I/O after synthesis. This command is placed in a Compile Netlist Constraint (*.ndc) file and passed to the Libero SoC Compile engine for netlist optimization after synthesis.

```
set_ioff {<portname>} \
          [-in_reg yes|no] \
          [-out_reg yes|no] \
          [-en_reg yes|no]
```

Arguments

<portname>

Specifies the name of the I/O port to be combined with a register. The port can be an input, output, or inout port.

-in_reg

Specifies whether the input register is combined into the port <portname>. Valid values are “yes” or “no”.

-out_reg

Specifes whether the output register is combined into the port <portname>. Valid values are “yes” or “no”.

-en_reg

Specifes whether the enable register is combined into the port <portname>. Valid values are “yes” or “no”.

Example

The following command specifies that for the port my_in_out[1] , the output register is combined into the port, but is not combined into the input register nor the enable register:

```
set_ioff {my_in_out[1]} –in_reg no –out_reg yes –en_reg no
```

The following command specifies that for the port my_in_out[19], the enable register is combined into the port, but is not combined into the input register nor the output register:

```
set_ioff {my_in_out[19]} –in_reg no –out_reg no –en_reg yes
```

The set_ioff command applies to scalar I/Os only. For an I/O bus, use the for-loop available in Tcl. The following command combines each scalar member of the 32-bit I/O bus DataA with input registers:

```
for { set i 0 } { i < 32 } { incr i } { set_ioff "DataA\[$i\]" -in_reg yes }
```

Alternatively, you can use a wild card to include all scalar signals of an I/O bus:

```
set_ioff {DataA[*]} -in_reg yes
```

Return Value

The command returns “0” on success and “1” on failure.

set_preserve

This command 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.

Exceptions

You must put this command in a PDC constraint file and associate it to Place and Route.

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
```

3 – Floorplanning PDC Commands

Floorplanning PDC commands are used to create and edit user regions and to assign/unassign logic to these regions.

assign_region

PDC command; 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 wild card characters in macro names:

Wild Card	What it does
\	Interprets the next character as a non-special character
?	Matches any single character
*	Matches any string

Exceptions

- The region must be created before you can assign macros to it. If the region creation PDC command and the macro assignment command are in different PDC files, the order of the PDC files is important.
- 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

"unassign_macro_from_region"

assign_net_macros

PDC command; 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 wild card characters in net names:net1

Wild Card	What it does
\	Interprets the next character as a non-special character
?	Matches any single character
*	Matches any string

-include_driver

Specifies whether to add the driver of the net(s) to the region. You can enter one of the following values:

Value	Descriptions
Yes	Include the driver in the list of macros assigned to the region (default).
No	Do not assign the driver to the region.

Exceptions

- Placed macros (not connected to the net) that are inside the area occupied by the net region are automatically unplaced.
- Net region constraints are internally converted into constraints on macros. PDC export results as a series of assign_region <region_name> macro1 statements for all the connected macros.
- If the region does not have enough space for all of the macros, or if the region constraint is impossible, the constraint is rejected and a warning message appears in the Log window.
- For overlapping regions, the intersection must be at least as big as the overlapping macro count.
- If a macro on the net cannot legally be placed in the region, it is not placed and a warning message appears in the Log window.
- Net region constraints may result in a single macro being assigned to multiple regions. These net region constraints result in constraining the macro to the intersection of all the regions affected by the constraint.

Examples

```
assign_net_macros cluster_region1 keyin1intZ0Z_62 -include_driver no
```

See Also

"unassign_net_macros"

define_region

PDC command; 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	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:

ColorDecimal Value

Color	Decimal Value
🟦	16776960
🟩	65280
🟦	16711680
🟦	16760960
🟥	255
🟪	16711935
🟨	65535
🟧	33023
🟥	8421631
🟩	9568200
🟪	8323199
⬜	12632256

-route value

Specifies whether to direct the routing of all nets internal to a region to be constrained within that region. A net is internal to a region if its source and destination pins are assigned to the region. 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: 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.

Description

Unlocked macros in empty or exclusive regions are unassigned from that region. You cannot create empty regions in areas that contain locked macros.

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, the region’s Properties Window displays the overbooked resources (over 100 percent resource utilization) in red.

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

"assign_region"

undefine_region

PDC command; 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.

Exceptions

To use this command, the region must have been previously defined.

Examples

```
undefine_region cluster_region1
```

See Also

"define_region"

move_region

PDC command; 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.

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"

unassign_net_macros

PDC command; 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.

Exceptions

If the region is currently not assigned, an error message appears in the Log window if you try to unassign it.

Examples

```
unassign_net_macros cluster_region1 keyin1intZ0Z_62
```

See Also

"unassign_macro_from_region"

"assign_net_macros"

unassign_macro_from_region

PDC command; 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 wild card characters in macro names:

Wild card	**What It Does**
\	Interprets the next character as a non-special character
?	Matches any single character
*	Matches any string

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"

set_location

PDC command; 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 Chip Planner tool to determine the x and y coordinates of the location.

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_location mem_data_in\[57\] -fixed yes 7 2
```

move_block

PDC command; moves a 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.

Note: If possible, routing is preserved when you move the blocks.

```
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. Refer to the Logical Veiw of Chip Planner for the instance name to use.

-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.

Description

This command moves a 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.

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: Microsemi recommends that you move the block left or right by increments of 12. If not, placement may fail because it violates clustering constraints. Also, Microsemi recommends that you move the block up or down by increments of three.

Exceptions

- You must associate this PDC constraint file to Place and Route.
- 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) 12 columns to the right and 3 rows up:

```
move_block -inst_name instA -right 12 -up 3 -non_logic move
```

The following example moves only the logic portion of the block and unplaces the rest by 24 columns to the right and 6 rows up.

```
move_block -inst_name instA -right 24 -up 6 -non_logic unplace
```

See Also

"set_block_options"

set_port_block

PDC command; 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.

Exceptions

- You must import this PDC command as a constraint 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 creation" option in the New Project wizard, 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
```

set_block_options

PDC command; overrides the compile option for placement or routing conflicts for an instance of a block.

```
set_block_options -inst_name instance_name -placement_conflicts value \
-routing_conflicts value
```

Arguments

-inst_name instance_name

Specifies the block instance name. If you do not know the name of the instance, run a Block Report (Design > Reports > Blocks > Interface) or look at the names shown in the Block view tab of the Chip Planner.

-placement_conflicts value

Specifies what to do when the 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 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.
discard	Discards any routing from the block, even if there are no conflicts.

Description

This command enables you to override the compile option for placement or routing conflicts for an instance of a block.

Exceptions

You must put this PDC command in a constraint PDC file and associate it to Place and Route.

If placement is discarded, the routing is automatically discarded too.

Examples

This example makes the Libero Soc 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"