Using Timing Constraints in SiliconBlue Designs

This application note describes the recommended approach for specifying Synplify Pro® timing constraints for designs that target SiliconBlue® technologies. See the following for details:

Introduction

The Synplify Pro synthesis tool reads input timing constraints in Synplicity® (`.sdc`) format and forward-annotates the constraints in Synopsys® constraint format (`.scf`) to the SiliconBlue place-and-route tool.

Note that for SiliconBlue technologies, the synthesis tool does not support all the constraints and formats described in the documentation. This document briefly describes the supported constraints that are forward-annotated for the SiliconBlue tool and how to use them. For full descriptions of the constraints, see the tool documentation.

Defining Clock Constraints

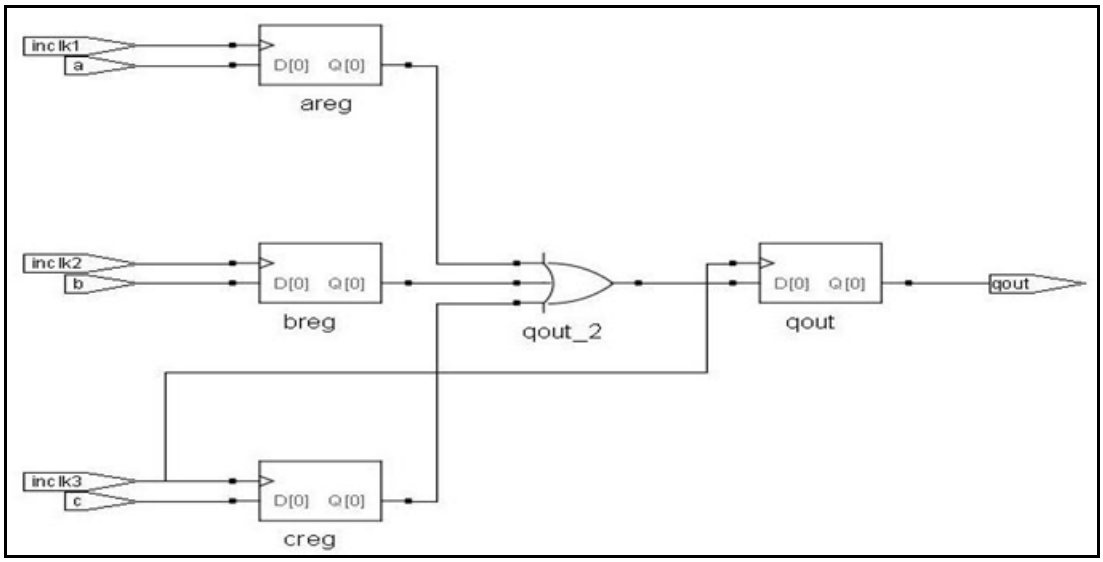
You define a clock constraint with the define_clock command, which lets you define a specific duty cycle and frequency or clock period goal. You can define clock constraints on top-level input ports, nets, or output pins of instantiated cells. See the following examples:

- Example 1: Clock Defined on a Port, on page 3
- Example 2: Clock Defined on a Net, on page 4
- Example 3: Clock Defined on a Pin, on page 5

Do not apply clock constraints to instances. If you need to apply such a constraint, apply it to the output pins or nets of the instances.

Example 1: Clock Defined on a Port

This example shows a clock defined on a port and the corresponding `.sdc` and forward-annotated `.scf` constraints.



E:/selvam/FPGA/SILICON_BLUE/constraint_usage_examples/define_clock/test.sdc

	Enabled	Clock Object	Clock Alias	Frequency (MHz)	Period (ns)	Clock Group	Rise At (ns)	Fall At (ns)	Duty Cycle (%)	Route (ns)	Virtu Clock
1	☑	inclk1	inclk1	200	5	default_clkgroup_0					☐
2	☑	p:inclk2	p:inclk2	100	10	default_clkgroup_0					☐
3	☑	p:inclk3	p:inclk3	50	20	default_clkgroup_0	10	15	25		☐

Synplicity Constraints

```
define_clock {inclk1} -name {inclk1}  -freq 200 -clockgroup default_clkgroup_0
define_clock {p:inclk2} -name {p:inclk2}  -freq 100 -clockgroup default_clkgroup_0
define_clock {p:inclk3} -name {p:inclk3}  -period 20 -clockgroup default_clkgroup_0
    -rise 10 -fall 15
```

Forward-annotated SCF Constraints

```
create_clock -period 5.000 -waveform {0.000 2.500} -name {inclk1} [get_ports {inclk1}]
create_clock -period 10.000 -waveform {0.000 5.000} -name {inclk2}
    [get_ports {inclk2}]
create_clock -period 20.000 -waveform {10.000 15.000} -name {inclk3} [get_ports
    {inclk3}]
```

If you put clocks in the same clock group, they are synchronous or related. To make the clocks asynchronous, put them in different clock groups. The synthesis tool treats all paths between clocks in different clock groups as false paths, and forward-annotates them accordingly.

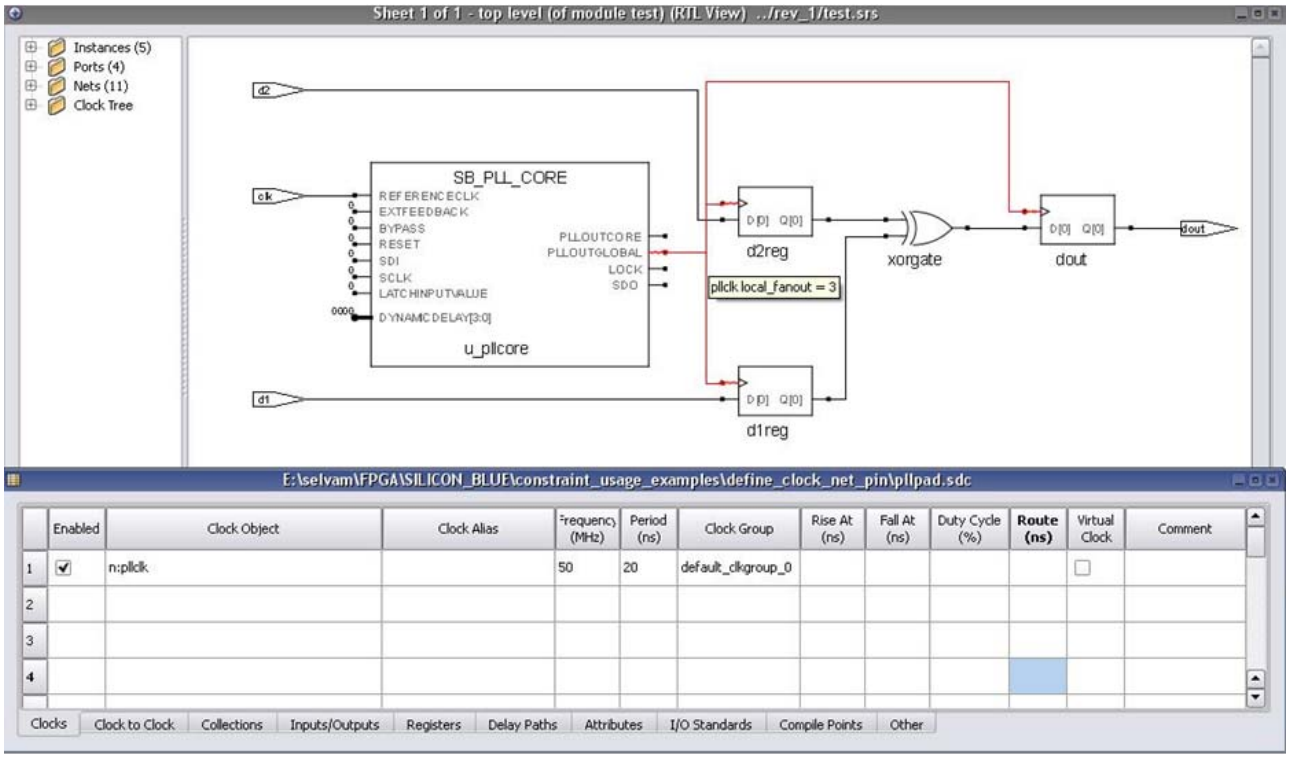
So if you want to set false paths between clocks, put them in different clock groups as shown in the following example:

Synplicity Constraints	define_clock {clk1} -freq 50 -clockgroup default_clkgroup_0 define_clock {clk2} -freq 100 -clockgroup default_clkgroup_1
Forward-annotated SCF Constraints	create_clock -period 20.000 -waveform {0.000 10.000} -name {clk1} [get_ports {clk1}] create_clock -period 10.000 -waveform {0.000 5.000} -name {clk2} [get_ports {clk2}] set_false_path -from [get_clocks {clk2}] -to [get_clocks {clk1}] set_false_path -from [get_clocks {clk1}] -to [get_clocks {clk2}]]

If your design has multiple clocks and you have not set any clock constraints, the tool automatically applies the default clock constraint and puts them in different clock groups. It treats the paths between the clocks as false paths.

Example 2: Clock Defined on a Net

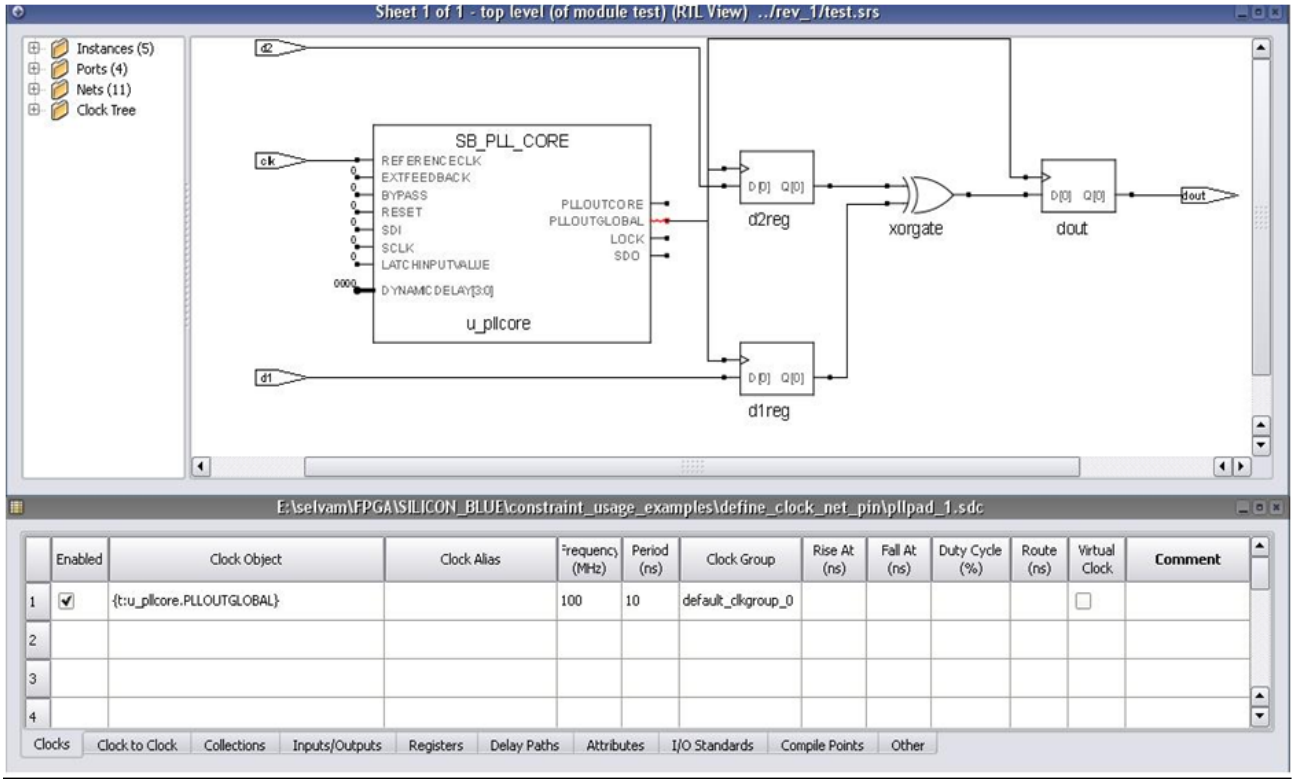
In this case, the clock constraint is applied on the pllclk net that connects the PLLOUTGLOBAL PLL output pin. You can either select the net and drag and it into the Clock Object cell in the SCOPE spreadsheet or you can type it into the SCOPE cell.



Synplicity Constraint	define_clock {n:pllclk} -freq 50 -clockgroup default_clkgroup_0
Forward-annotated SCF Constraint	create_clock -period 20.000 -waveform {0.000 10.000} -name {pllclk} [get_pins {u_pllcore/PLLOUTGLOBAL}]

Example 3: Clock Defined on a Pin

To apply a clock constraint on a pin, select the pin, copy it using Ctrl+c, and paste it in the SCOPE Clock Object field. Alternatively, drag and drop the pin into the SCOPE spreadsheet, or type the pin name in the Clock Object field.



Synplicity Constraint	define_clock {{t:u_pllcore.PLLOUTGLOBAL}} -freq 100 -clockgroup default_clkgroup_0
Forward-annotated SCF Constraint	create_clock -period 10.000 -waveform {0.000 5.000} -name {u_pllcore.PLLOUTGLOBAL} [get_pins {u_pllcore/PLLOUTGLOBAL}

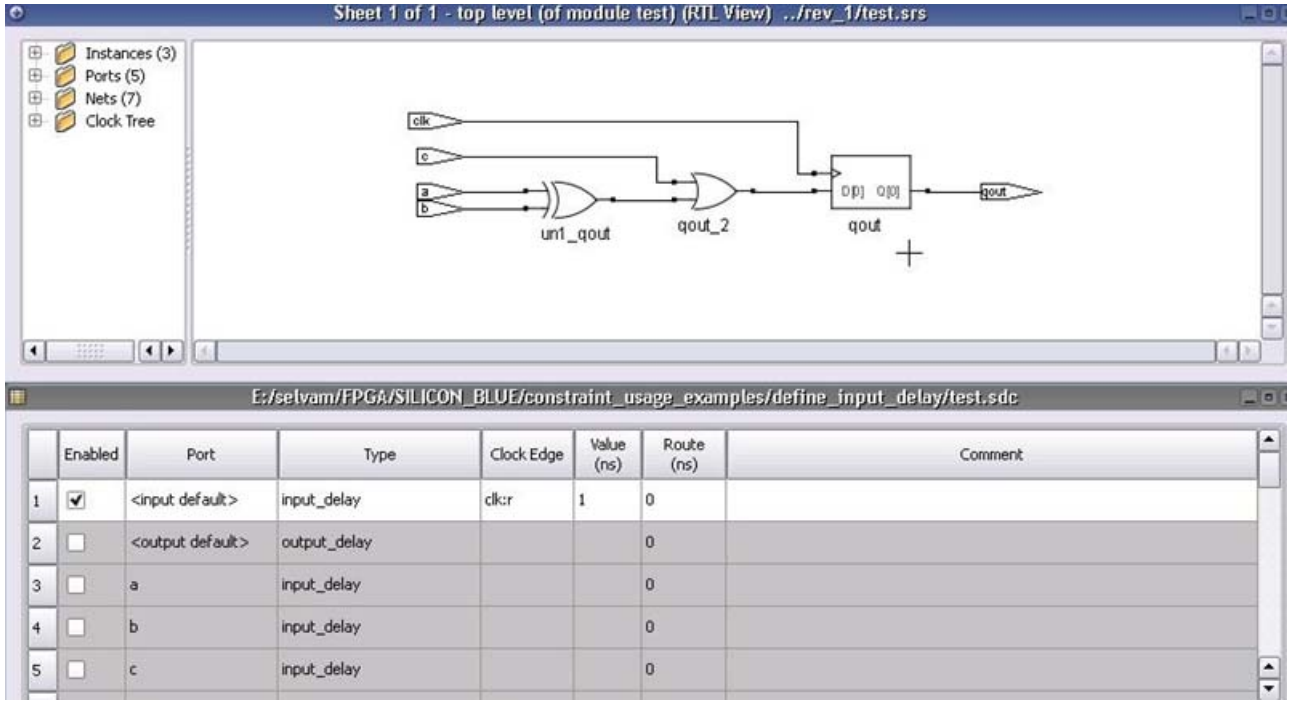
Defining Input Delays

You can specify external input delays for the top-level ports in the design. See the following examples:

- Example 1: Port Input Delay with -default Option, on page 6
- Example 2: Input delay on Individual Ports, on page 7

Example 1: Port Input Delay with -default Option

When the -default option is specified, as in the following example, the tool applies the default input delay to all input ports except clock ports.



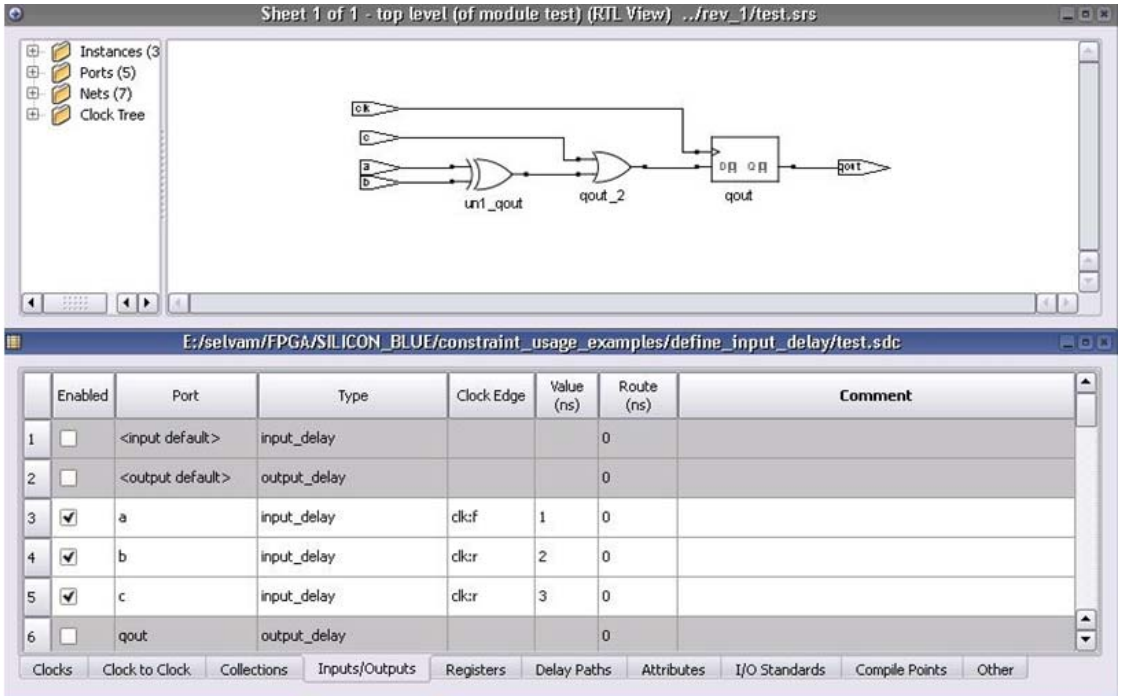
Synplicity Constraints	define_clock {clk} -name {clk} -freq 200 -clockgroup default_clkgroup_0 define_input_delay -default 1.00 -improve 0.00 -route 0.00 -ref {clk:r}
Forward-annotated SCF Constraints	create_clock -period 5.000 -waveform {0.000 2.500} -name {clk} [get_ports {clk}] set_input_delay -max 1.000 -clock [get_clocks {clk}] -add_delay [get_ports {a}] set_input_delay -max 1.000 -clock [get_clocks {clk}] -add_delay [get_ports {b}] set_input_delay -max 1.000 -clock [get_clocks {clk}] -add_delay [get_ports {c}]

Note the following:

- If you specify the -route or -improve options, the tool only uses them for synthesis. They are not forward-annotated.
- It is recommended that you use the -ref option with the clock name and the edge that triggers the event. The tool forward-annotates this information accordingly.

Example 2: Input delay on Individual Ports

In this example the input delays are applied on individual input ports, with reference to different clock edges.



Synplicity Constraints

```
define_clock   {clk} -name {clk}  -freq 200 -clockgroup default_clkgroup_0
define_input_delay {a}  1.00 -improve 0.00 -route 0.00 -ref {clk:f}
define_input_delay {b}  2.00 -improve 0.00 -route 0.00 -ref {clk:r}
define_input_delay {c}  3.00 -improve 0.00 -route 0.00 -ref {clk:r}
```

Forward-annotated SCF Constraints

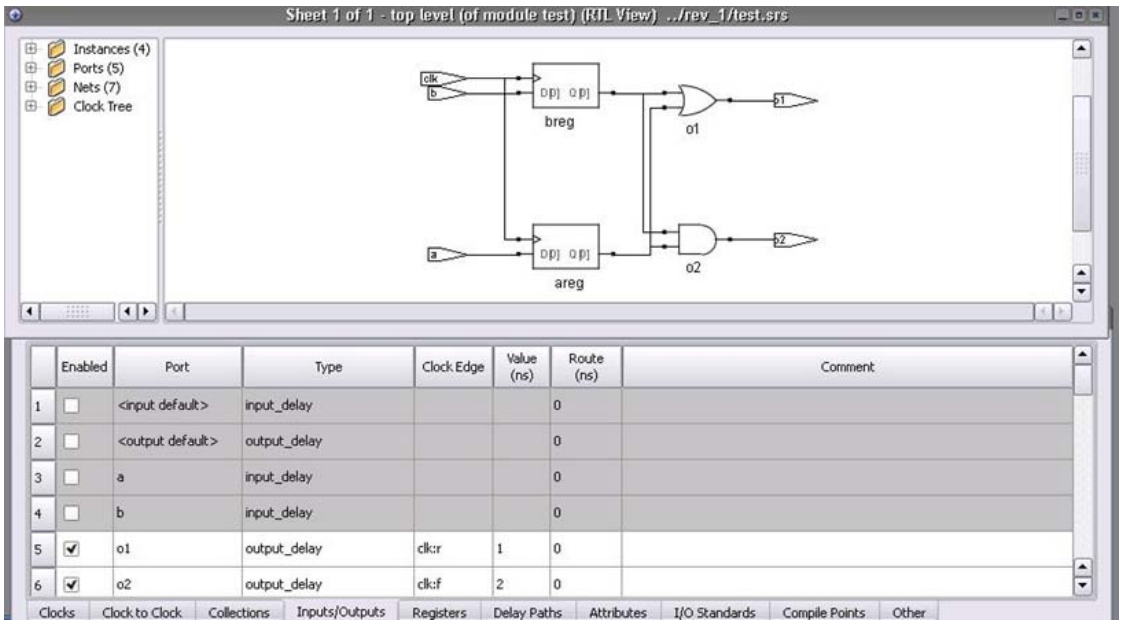
```
create_clock -period 5.000 -waveform {0.000 2.500} -name {clk} [get_ports {clk}]
set_input_delay -max 1.000 -clock [get_clocks {clk}] -clock_fall -add_delay
    [get_ports {a}]
set_input_delay -max 2.000 -clock [get_clocks {clk}] -add_delay [get_ports {b}]
set_input_delay -max 3.000 -clock [get_clocks {clk}] -add_delay [get_ports {c}]
```

Defining Output Delays

You can specify external output delays on the top-level ports of the design. Note the following:

- The tool ignores I/O paths for timing unless they are constrained by having input or output delays defined for them.
- The tool uses the define_reg_input_delay and define_reg_output_delay constraints for synthesis only, and does not forward-annotate them.

Example: Output Delay on Output Ports



Synplicity Constraints	define_clock {clk} -name {clk} -freq 100 -clockgroup default_clkgroup_0 define_output_delay {o1} 1.00 -improve 0.00 -route 0.00 -ref {clk:r} define_output_delay {o2} 2.00 -improve 0.00 -route 0.00 -ref {clk:f}
Forward-annotated SCF Constraints	create_clock -period 10.000 -waveform {0.000 5.000} -name {clk} [get_ports {clk}] set_output_delay -max 1.000 -clock [get_clocks {clk}] -add_delay [get_ports {o1}] set_output_delay -max 2.000 -clock [get_clocks {clk}] -clock_fall -add_delay [get_ports {o2}]

Defining Multicycle Paths

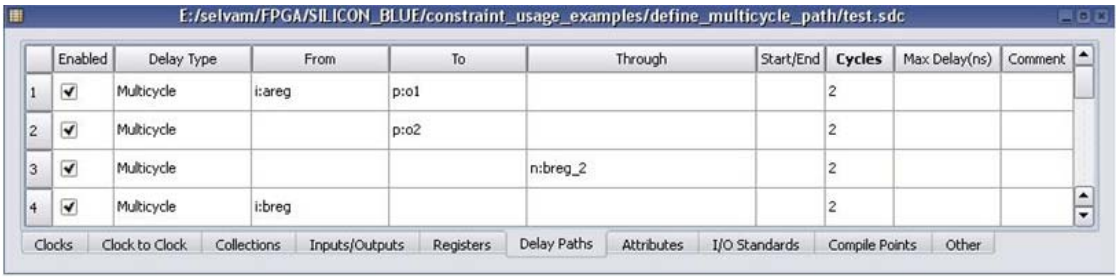
You can define a path as a timing exception because it uses multiple clock cycles. You can use the following constraints for multicycle paths:

- -from
  Registers (i:) or top-level input or bi-directional ports (p:)
- -to
  Registers (i:) or top-level output or bi-directional ports (p:)
- -through
  Combinational nets (n:) or pins on instantiated cells (t::)

Note the following:

- Do not use clocks as from/to points. It is not recommended.
- The tool uses forward-annotated multicycle path constraints only for setup.
- Use the -through option only for combinational nets and pins of the instantiated cells.
- When applying multicycle paths from or to the ports, make sure that I/O delay constraints are applied to them.

Example: Multicycle Paths



Synplicity Constraints	define_multicycle_path -from {{i:areg}} -to {{p:o1}} 2 define_multicycle_path -to {{p:o2}} 2 define_multicycle_path -through {n:breg_2} 2 define_multicycle_path -from {{i:breg}} 2
Forward-annotated SCF Constraints	set_multicycle_path 2 -setup -from [get_cells {areg}] -to [get_ports {o1}] set_multicycle_path 2 -setup -to [get_ports {o2}] set_multicycle_path 2 -setup -through [get_pins {breg_2_keep_RNO/O}] set_multicycle_path 2 -setup -from [get_cells {breg}]

Defining False Paths

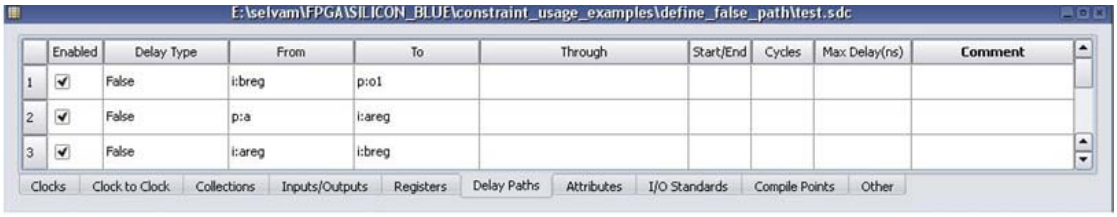
You can define paths to be ignored during timing analysis. You can define the following constraints for false paths:

- -from
  Registers (i:) or top-level input or bi-directional ports (p:)
- -to
  Registers (i:) or top-level output or bi-directional ports (p:)
- -through
  Combinational nets (n:) or pins on instantiated cells (t::)

Note the following:

- Do not use clocks as from/to points. It is not recommended.
- Use the -through option only for combinational nets and pins of the instantiated cells.
- For false path constraints on input or output ports, you must have I/O delays specified for the path from input or to the output ports. The false path constraint is only valid and forward-annotated when the delays are specified on the input and output ports as described in *Defining Input Delays*, on page 6 and *Defining Output Delays*, on page 8.

Example 1: False Path from Register to Output Port



Synplicity Constraint	define_false_path -from {{i:breg}} -to {{p:o1}}
Forward-annotated SCF Constraint	set_false_path -from [get_cells {breg}] -to [get_ports {o1}]

Example 2: False Path from Input Port to Register

Synplicity Constraint	define_false_path -from {{p:a}} -to {{i:areg}}
Forward-annotated SCF Constraint	set_false_path -from [get_ports {a}] -to [get_cells {areg}]

Example 3: False Path from Register to Register

Synplicity Constraint	define_false_path -from {{i:areg}} -to {{i:breg}}
Forward-annotated SCF Constraint	set_false_path -from [get_cells {areg}] -to [get_cells {breg}]

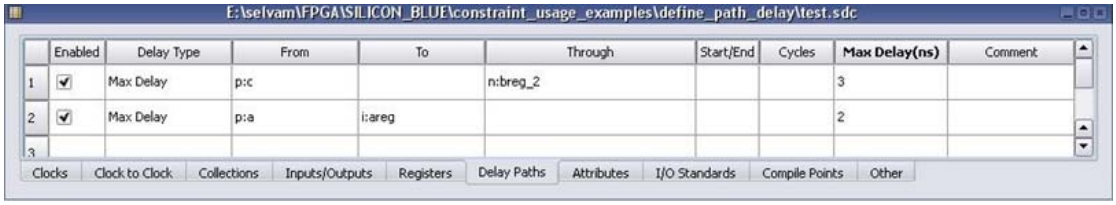
Defining Path Delays

You can define maximum delays between specific points in the design. You can specify the following point-to-point delays:

- -from
  Registers (i:) or top-level input or bi-directional ports (p:)
- -to
  Registers (i:) or top-level output or bi-directional ports (p:)
- -through
  Combinational nets (n:) or pins on instantiated cells (t::)

Do not use clocks as from/to points. It is not recommended.

Example 1: Maximum Delay from a Port Through a Net



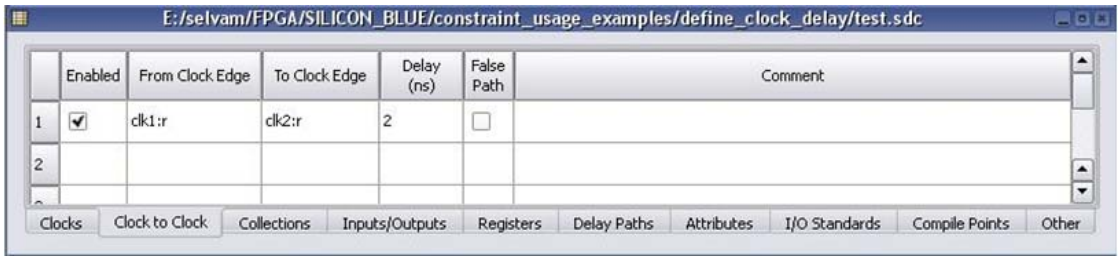
Synplicity Constraint	define_path_delay -from {{p:c}} -through {n:breg_2} -max 3
Forward-annotated SCF Constraint	set_max_delay 3.000 -from [get_ports {c}] -through [get_pins {breg_2_keep_RNO/O}]

Example 2: Maximum Delay from a Port to an Instance

Synplicity Constraint	define_path_delay -from {{p:a}} -to {{i:areg}} -max 2
Forward-annotated SCF Constraint	set_max_delay 2.000 -from [get_ports {a}] -to [get_cells {areg}]

Defining Clock Delays

You can define delays between the clocks in the design. The synthesis tools do not currently support collections for clock-to-clock delay constraints. Use the clock names as shown.



Example: Clock-to-Clock Delay

Synplicity Constraint	define_clock_delay -rise {clk1} -rise {clk2} 2
Forward-annotated SCF Constraint	set_max_delay 2.000 -rise_from [get_clocks {clk1}] -rise_to [get_clocks {clk2}]

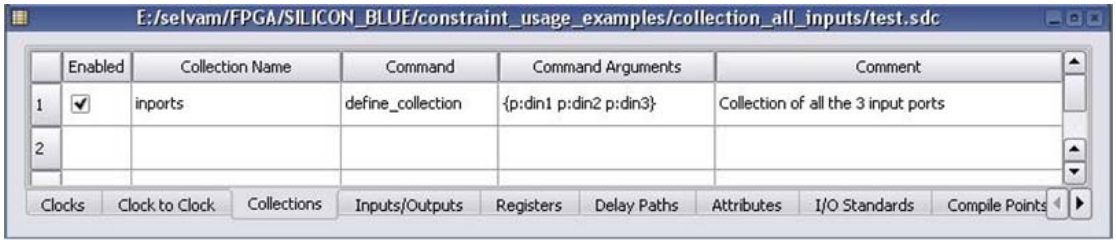
Using Collections

A collection is a group of objects. You can apply constraints to collections, thus making it easier to apply constraints. Do not use collections to apply clock-to-clock delay constraints.

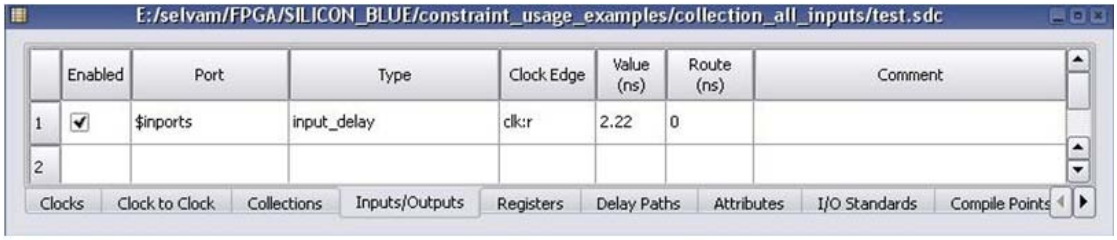
The following are some examples illustrating the use of collections. Refer to the tool documentation for more detailed information.

Example 1: Input Port Collection

In this example, you first create a collection of input ports:



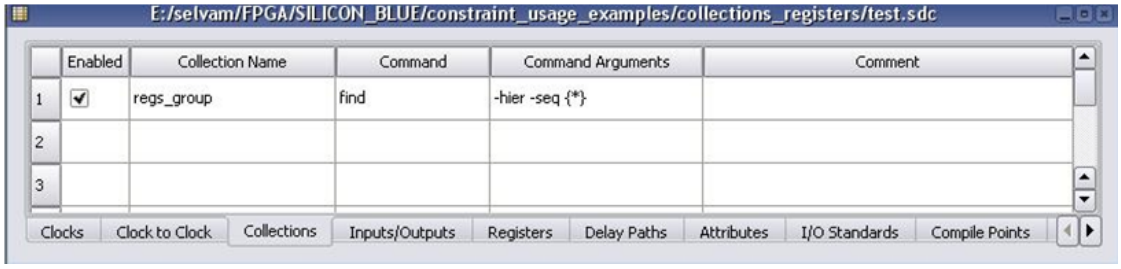
You then apply an input delay constraint to the collection. This applies the constraint to all the ports in the collection.



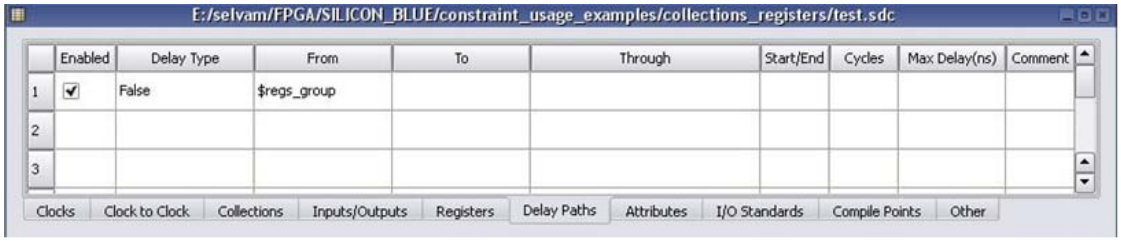
Synplicity Constraints	define_scope_collection -comment {Collection of all the 3 input ports} inports {define_collection {p:din1 p:din2 p:din3}} define_input_delay {$inports} 2.22 -improve 0.00 -route 0.00 -ref {clk:r}
Forward-annotated SCF Constraints	set_input_delay -max 2.220 -clock [get_clocks {clk}] -add_delay [get_ports {din1}] set_input_delay -max 2.220 -clock [get_clocks {clk}] -add_delay [get_ports {din2}] set_input_delay -max 2.220 -clock [get_clocks {clk}] -add_delay [get_ports {din3}]

Example 2: Register Collection Created with the find Command

In this example, you first define a collection of registers using the Tcl find command:



You then apply a false path constraint to the collection:



Synplicity Constraint	define_scope_collection regs_group {find -hier -seq {*}}
Forward-annotated SCF Constraint	set_false_path -from [get_cells {areg breg}]

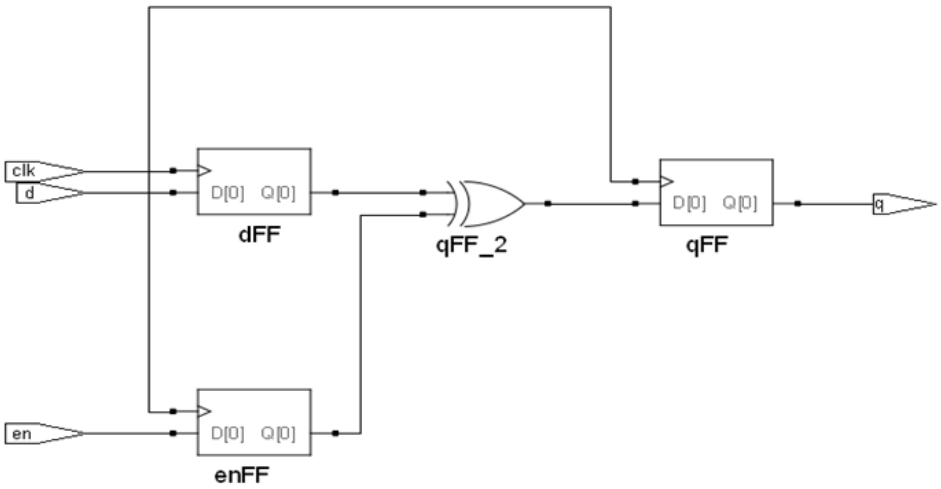
Forward-Annotation of Constraints on Merged Registers

The following examples describe forward-annotation details for constraints set on registers that get merged into SiliconBlue IOBs or RAMs.

Example 1: Register Packed into the IOB

This example has the following sdc input constraints applied:

- False path constraint from the enFF instance
- Path_delay constraint from qFF instance
- Multicycle path constraint between the dFF and qFF flip-flop instances

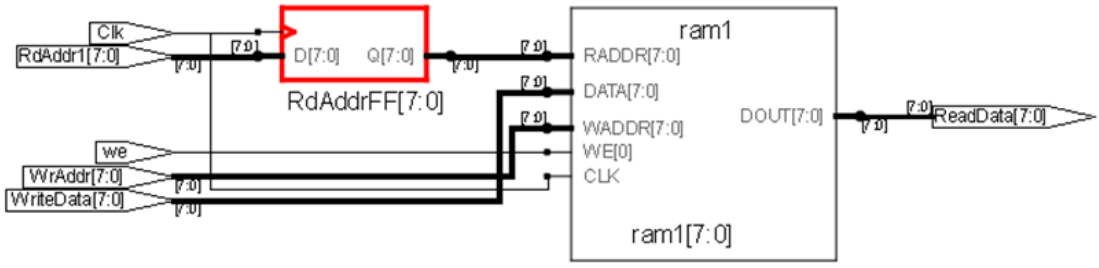


The forward-annotation results vary, depending on whether register packing is enabled (syn_useioff=1). See the following table:

Synplicity Constraints	define_clock {clk} -name {clk} -freq 100 -clockgroup default_clkgroup_0 define_input_delay -default 0.00 -ref {clk:r} define_output_delay -default 0.00 -ref {clk:r} define_false_path -from {{i:enFF}} define_multicycle_path -from {{i:dFF}} -to {{i:qFF}} 2 define_path_delay -from {{i:qFF}} -max 8
Forward-annotation (no register packing)	create_clock -period 10.000 -waveform {0.000 5.000} -name {clk} [get_ports {clk}] set_multicycle_path 2 -setup -from [get_cells {dFF}] -to [get_cells {qFF}] set_max_delay 8.000 -from [get_cells {qFF}] set_false_path -from [get_cells {enFF}] set_input_delay -max 0.000 -clock [get_clocks {clk}] -add_delay [get_ports {en}] set_input_delay -max 0.000 -clock [get_clocks {clk}] -add_delay [get_ports {d}] set_output_delay -max 0.000 -clock [get_clocks {clk}] -add_delay [get_ports {q}]
Forward-annotation (with register packing)	All constraints applied on registers that are merged into I/Os are lost. These constraints are not honored or forward-annotated. To retain the constraints, disable I/O packing on the registers where the constraints are applied. This behavior is the same as in the general synthesis flow.

Example 2: Register Merged into a RAM

In this example, a multicycle -to constraint is applied on the i:RdAddrFF instance, which is the registered read address of the RAM.



Synplicity Constraints	define_clock {Clk} -name {Clk} -freq 150 -clockgroup default_clkgroup_0 define_multicycle_path -to {{i:RdAddrFF[7:0]}} 2
Forward-annotation	The multicycle constraint is not forward-annotated because the read address registers are merged into the RAM. This behavior is the same as in our general flow.

Synopsys, Inc.
700 East Middlefield Road
Mountain View, CA 94043 USA
www.solvnet.com