Lattice Radiant Timing Constraints Methodology

Application Note

FPGA-AN-02059-1.2

July 2023
Disclaimers
Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk the responsibility entirely of the Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer’s responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice’s product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.
Contents

Acronyms in This Document .................................................................................................................. 5
1. Introduction ........................................................................................................................................ 6
  1.1. Audience ..................................................................................................................................... 6
2. Overview ............................................................................................................................................ 7
3. SDC Constraints Supported in Radiant ............................................................................................... 8
  3.1. Clock Constraints .......................................................................................................................... 8
    3.1.1. create_clock Constraint ........................................................................................................ 8
    3.1.2. create_generated_clock Constraint .................................................................................... 9
    3.1.3. set_clock_latency Constraint ............................................................................................... 11
    3.1.4. set_clock_uncertainty Constraint ....................................................................................... 12
    3.1.5. set_clock_group Constraint .............................................................................................. 13
  3.2. Input/Output Delay Constraints .................................................................................................... 14
    3.2.1. set_input_delay Constraint .................................................................................................. 14
    3.2.2. set_output_delay Constraint .............................................................................................. 15
  3.3. Timing exception constraints .......................................................................................................... 16
    3.3.1. set_false_path Constraint ..................................................................................................... 17
    3.3.2. set_max_delay Constraint .................................................................................................... 17
    3.3.3. set_min_delay Constraint .................................................................................................... 17
    3.3.4. set_multicycle_path Constraint .......................................................................................... 18
4. Constraints Entry in Radiant ................................................................................................................ 19
  4.1. Using Pre-Synthesis Timing Constraints Editor ............................................................................ 19
  4.2. Using Post-Synthesis Timing Constraints Editor .......................................................................... 20
  4.3. Using manual constraints entry ................................................................................................... 21
5. Timing Constraints effect on Implementation Process ......................................................................... 22
  5.1. Timing Constraints effect on Synthesis ....................................................................................... 22
  5.2. Timing Constraints effect on MAP ............................................................................................. 23
  5.3. Timing Constraint effect on Place and Route ............................................................................. 23
6. User constraints storage during the implementation process ............................................................. 24
7. Constraints Propagation Engine .......................................................................................................... 25
  7.1. CPE rules ..................................................................................................................................... 25
  7.2. CPE Usage and Recommendation ............................................................................................... 27
  7.3. CPE Output Files .......................................................................................................................... 27
    7.3.1. CPEreport.txt file ................................................................................................................ 27
    7.3.2. CPE generated .ldc file ....................................................................................................... 28
8. References ............................................................................................................................................ 29
9. Technical Support Assistance ................................................................................................................ 30
10. Revision History ................................................................................................................................. 31
Figures
Figure 2.1. Overview of Radiant Constraints Consumption ................................................................. 7
Figure 3.1. DUT ................................................................................................................................. 9
Figure 3.2. PLL with clocks ................................................................................................................ 10
Figure 3.3. PLL using the create_clock Constraints ........................................................................ 11
Figure 3.4. Clock Latency .................................................................................................................. 12
Figure 3.5. FPGA Driven by an External Design Outside the FPGA .................................................. 14
Figure 3.6. External Circuit is Driven by External Clocks ................................................................. 15
Figure 3.7. Clock from Inside the FPGA Drives External Clock .......................................................... 16
Figure 4.1. Pre-Synthesis Timing Constraints Editor Tools Menu ..................................................... 19
Figure 4.2. Post-synthesis Timing Constraints Editor Tools Menu .................................................... 20
Figure 4.3. To Edit a Greyed-out Constraint ...................................................................................... 20
Figure 4.4. Create a Constraints File ................................................................................................ 21
Figure 4.5. Adding LSE Design Constraints File .............................................................................. 21
Figure 5.1. Synthesis Strategy Options for Both LSE and Synplify Pro ............................................. 22
Figure 5.2. Place and Route Design Strategy Options ....................................................................... 23
Figure 7.1. Example of a CPEreport.txt file ...................................................................................... 28
Figure 7.2. Example of a CPE generated .ldc file ........................................................................... 28

Tables
Table 2.1. Supported File Formats by the Implementation Tools in Radiant ....................................... 7
Table 3.1. Clock Constraints ............................................................................................................. 8
Table 3.2. create_clock Constraint Options ...................................................................................... 8
Table 3.3. create_generated_clock Constraint Options .................................................................... 10
Table 3.4. set_clock_latency Constraint Options ............................................................................. 11
Table 3.5. set_clock_uncertainty Constraint Options ....................................................................... 13
Table 3.6. set_clock_group Constraint Options ............................................................................. 13
Table 3.7. Input/output Delay Constraints Supported in Radiant .................................................... 14
Table 3.8. set_input_delay Constraint Options ................................................................................ 14
Table 3.9. set_output_delay Constraint Options ............................................................................ 15
Table 3.10. Timing Exception Constraints ......................................................................................... 16
Table 3.11. set_false_path Constraint Options ............................................................................... 17
Table 3.12. set_max_delay Constraint options.................................................................................. 17
Table 3.13. set_min_delay Constraint Options ................................................................................ 18
Table 3.14. set_multicycle_path Constraint Options ....................................................................... 18
Table 6.1. Contents stored in UDB .................................................................................................... 24
Table 7.1. CPE Rules for Ignored Constraints ................................................................................... 26
Table 7.2. CPE Rules for Resolved Constraints ................................................................................ 26
Table 7.3. Timing Constraint Resolution Summary .......................................................................... 27
# Acronyms in This Document

A list of acronyms used in this document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDC</td>
<td>Lattice Design Constraints</td>
</tr>
<tr>
<td>SDC</td>
<td>Synopsys Design Constraints</td>
</tr>
<tr>
<td>FDC</td>
<td>FPGA Design Constraints</td>
</tr>
<tr>
<td>PDC</td>
<td>Physical Design Constraints</td>
</tr>
<tr>
<td>LSE</td>
<td>Lattice Synthesis Engine</td>
</tr>
<tr>
<td>PAR</td>
<td>Place And Route</td>
</tr>
<tr>
<td>UDB</td>
<td>Unified Database</td>
</tr>
<tr>
<td>SI</td>
<td>Signal Integrity</td>
</tr>
<tr>
<td>TCE</td>
<td>Timing Constraints Editor</td>
</tr>
<tr>
<td>CPE</td>
<td>Constraints Propagation Engine</td>
</tr>
</tbody>
</table>
1. Introduction

Lattice Radiant® software is the complete design environment for Lattice Semiconductor Field Programmable Gate Arrays (FPGAs). The software includes a comprehensive set of tools for all design tasks, including project management, design entry, simulation, synthesis, place and route, in-system logic analysis, and more.

To ensure the design meets its desired performance goals on the FPGA, it is the responsibility of the users to provide proper timing constraints to the design. The implementation tools in Radiant reads in the constraints provided and works its way to meet the performance requirements. In Radiant, timing constraints are consumed throughout the design implementation flow starting from synthesis to place and route. Physical constraints are only consumed by the back-end flow post-synthesis. Users can still enter the physical constraints using the pre-synthesis constraints file which gets properly passed down to the implementation tools during the flow.

Lattice Radiant supports Lattice Design Constraints (LDC) based on the standard Synopsys™ Design Constraints (SDC) supported by our in-house synthesis tool Lattice Synthesis Engine (LSE). Radiant also supports Synopsys Design Constraints (SDC) file format for both LSE and Synopsys™ Synplify Pro design flows and FPGA Design Constraints (FDC) for Synplify Pro flow. For post-synthesis implementation flow, Radiant supports Physical Design Constraints (PDC) for timing and physical constraints.

This document covers the in-depth timing constraints methodology that is used in Lattice Radiant software.

1.1. Audience

The intended audience for this document includes FPGA design engineers using Lattice Radiant design software. The technical guidelines assumes that the readers have some basic knowledge on the SDC constraints usage.
2. Overview

Constraints are consumed throughout the implementation flow starting from Synthesis to Place and Route. The primary goal of the tools is to meet user constraints and performance goals. Figure 2.1 shows an overview of Radiant constraints consumption throughout the implementation flow.

Radiant supports a subset of standard SDC constraints. The LDC and SDC files allow users to input pre-synthesis timing constraints along with physical constraints and synthesis attributes which will be used for synthesis optimizations and post-synthesis timing analysis. The physical constraints entered in the user LDC or SDC files are not consumed by the synthesis tools. These constraints are written into the database file (.udb) which are then consumed by MAP and PAR engine for implementation during the flow. Users can also enter physical constraints using the PDC file which will be consumed by both MAP and PAR processes. Table 2.1 shows the supported file formats by the implementation tools in Radiant.

<table>
<thead>
<tr>
<th>Constraint type</th>
<th>Implementation Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>.ldc</td>
<td>LSE</td>
</tr>
<tr>
<td>.sdc</td>
<td>LSE &amp; Synplify Pro</td>
</tr>
<tr>
<td>.fdc</td>
<td>Synplify Pro</td>
</tr>
<tr>
<td>.pdc</td>
<td>MAP &amp; PAR</td>
</tr>
</tbody>
</table>
3. SDC Constraints Supported in Radiant

The section describes the SDC constraints supported in Radiant.

3.1. Clock Constraints

Table 3.1 lists all the clock constraints supported in Radiant.

Table 3.1. Clock Constraints

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_clock</td>
<td>Defines clock constraints for the clocks used in the design</td>
</tr>
<tr>
<td>create_generated_clock</td>
<td>Defines generated clocks in the design</td>
</tr>
<tr>
<td>set_clock_latency</td>
<td>Specifies source clock latency outside the FPGA</td>
</tr>
<tr>
<td>set_clock_uncertainty</td>
<td>Used to over constrain the clock by accounting jitter</td>
</tr>
<tr>
<td>set_clock_group</td>
<td>Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during timing analysis.</td>
</tr>
</tbody>
</table>

3.1.1. create_clock Constraint

The create_clock constraint is used to define primary and virtual clocks in the design. The syntax for create_clock constraint is:

```
create_clock -period <period>
[-name <clock name>]
[-waveform <value1 value2>]
[get_ports | get_pins | get_nets <source_object>]
```

Table 3.2. create_clock Constraint Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-period</td>
<td>Used to specify clock period</td>
</tr>
<tr>
<td>-name</td>
<td>Used to refer the clock in other commands</td>
</tr>
<tr>
<td>-waveform</td>
<td>Used to adjust duty cycle and phase</td>
</tr>
</tbody>
</table>
This constraint can be defined using various ways which is described in Constraints Entry in Radiant section.

**Example:**

Let us consider the following example design. In Figure 3.1, clkA is a 100 MHz clock that drives the design.

![Diagram](image)

**Figure 3.1. DUT**

User must define a create clock constraint at this port clkA. The constraint for clkA with a frequency of 100 MHz with a duty cycle of 50% can be defined as:

```
create_clock -name clk -period 10 [get_ports ClkA]
```

In general, clocks can be of three clocks:

- **Base clock**: Defined and specified using create_clock constraint using the clock name and a clock object.
- **Virtual clock**: Defined using the create_clock constraint with a clock name and no object.
  
  **Example**: `create_clock -name virt_clk -period 10`
  
  This creates a virtual clock called "virt_clk" with a period of 10 ns and is only used for timing analysis purposes.

- **Generated clocks**: These clocks are derived from the base clocks either using PLL or user logic and are defined using the `create_generated_clock` constraint.

**3.1.1. Usage Guidelines for create_clock Constraint**

- It is always preferred to define clocks on the top-level clock ports.
- If the clocks are defined on the pins/nets, Radiant timer will not consider the IO buffer delays.

**3.1.2. create_generated_clock Constraint**

A `create_generated_clock` constraint is used to define clocks inside the design. For example, a `create_generated_clock` can be used to define PLL output clocks, clock divider output clocks etc. The syntax for `create_generated_clock` constraints is:

```
create_generated_clock -source <clock_source>
[ -divide_by <factor>]  
[ -multiply_by <factor>]  
[ -duty_cycle <value>] | [ -edges <edge specs>]  
[ -invert]  
[ -name <clock name>]  
[ get_pins | get_nets <pin/net name>]  
<target_object>
```

---

© 2023 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.
Table 3.3. `create_generated_clock` Constraint Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-source</td>
<td>Used to specify the clock source</td>
</tr>
<tr>
<td>-divide_by</td>
<td>Used to specify the frequency divide factor</td>
</tr>
<tr>
<td>-multiply_by</td>
<td>Used to specify the frequency multiply factor</td>
</tr>
<tr>
<td>-duty_cycle</td>
<td>Used to specify the duty cycle (in percentage) if frequency multiplication is used</td>
</tr>
<tr>
<td>-edges</td>
<td>Used to specify a list of positive integers that represents the edges from the source clock that are to form the edges of the generated clock</td>
</tr>
<tr>
<td>-invert</td>
<td>Inverts the generated clock signal</td>
</tr>
<tr>
<td>-add</td>
<td>Used to add the clock to the existing clock</td>
</tr>
</tbody>
</table>

**Example:** Let us consider the Figure 3.2. In this example, clk is a 100 MHz input clock to a PLL which is generating three clocks 50 MHz, 200 MHz, and 75 MHz.

![PLL Diagram](image)

**Figure 3.2.** PLL with clocks

To constrain the above example, user may use the following constraints:

- **# Define clock constraint for clk**
  ```verilog```
  create_clock -name clk -period 10 [get_ports clk]
  ```verilog```

- **# Define a 50 MHz generated_clock constraint to constrain clk1**
  ```verilog```
  create_generated_clock -name clk1 -source [get_ports clk] -divide_by 2 [get_pins clkop]
  ```verilog```

- **# Define a 200 MHz generated_clock constraint to constrain clk2**
  ```verilog```
  create_generated_clock -name clk2 -source [get_ports clk] -multiply_by 2 [get_pins clkos]
  ```verilog```

- **# Define a 75 MHz generated_clock constraint to constrain clk3**
  ```verilog```
  create_generated_clock -name clk3 -source [get_ports clk] -multiply_by 3 -divide_by 4 [get_pins clkos2]
  ```verilog```

### 3.1.2.1. Usage Guidelines

- Must define a primary clock source using a `create_clock` constrain before defining the `create_generated_clock` constraint.
- Must define a fixed relationship between the primary and secondary clocks.
- If PLL or OSC hard IP is used, Radiant timer automatically infers the generated clock constraints for the PLL output clocks. Users must define the input clock to the PLL.
3.1.2.2. PLL Constraining Guidelines

Users must follow the following guidelines to constrain the PLL clocks correctly:

- Clock should be defined on the top-level port that feeds the reference clock pin of the PLL using the create_clock constraints. If this is not followed:
  - Insertion delay will be inaccurate.
  - Relationship between PLL output clocks will be lost if clocks are defined at the output of the PLL using create_clock constraint.
- create_generated_clocks could either be defined by the user or preferably be generated by the timing engine at the output pins of the PLL.

![PLL model has delays And device constraints](image)

**Figure 3.3. PLL using the create_clock Constraints**

3.1.3. set_clock_latency Constraint

The set_clock_latency constraint is used to specify the source clock latency outside the FPGA. This constraint can also be used to specify the delay from virtual clock to actual port arrival, etc. The syntax for set_clock_latency constraint is:

```
set_clock_latency <value>
  -source
  [-rise]
  [-fall]
  [-early | -late]
  [get_clocks <clock name>]
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-source</td>
<td>Used to specify the clock source</td>
</tr>
<tr>
<td>-rise</td>
<td>Indicates that delay is to apply only to rise clock. By default, delay is applied to both rise and fall clock latency</td>
</tr>
<tr>
<td>-fall</td>
<td>Indicates that delay is to apply only to fall clock. By default, delay is applied to both rise and fall clock latency.</td>
</tr>
<tr>
<td>-early</td>
<td>Indicates that delay is to apply only to early clock source latency (Fastest path). By default, if -source is specified, delay is applied to both late and early clock source latency.</td>
</tr>
<tr>
<td>-late</td>
<td>Indicates that delay is to apply only to late clock source latency (Longest path). By default, if -source is specified, delay is applied to both late and early clock source latency.</td>
</tr>
</tbody>
</table>
Example: Let us consider the Figure 3.4. Let’s assume there is a source clock latency of 0.5 ns outside the FPGA from the clock source before the clock reaches the FPGA boundary.

![Figure 3.4. Clock Latency](image)

User can use the following constraint to specify the clock latency:

```plaintext
set_clock_latency -source 0.5 [get_clocks clk]
```

Here’s an example to source latency delay of the longest clock path using the using the -late option.

```plaintext
set_clock_latency -source -late 1.0 [get_clocks clk]
```

Here’s an example to source latency delay of the shortest clock path using the using the -early option.

```plaintext
set_clock_latency -source -early 0.5 [get_clocks clk]
```

### 3.1.3.1. Usage Guidelines

- The Min and Max delay are used to find the minimum and maximum arrival times even when there is a single path.
- For setup analysis, TA uses the late clock latency for the data arrival path and the early clock latency for the clock arrival path.
- For hold analysis, TA uses the early clock latency for the data arrival time and the late clock latency for the clock arrival time.
- For Nexus devices, maximum is only for setup calculation.

### 3.1.4. set_clock_uncertainty Constraint

The set_clock_uncertainty constraint is usually used to over constrain the design and for taking Jitter into account. Uncertainty also covers all other static/dynamic timing uncertainties too. – skew, crosstalk, SI etc. The syntax for set_clock_uncertainty constraint is:

```plaintext
set_clock_uncertainty [-setup] [-hold] [-from <clock>] [-to <clock>] <uncertainty_value> [clock_list]
```
Table 3.5. set_clock_uncertainty Constraint Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-setup</td>
<td>Indicates that uncertainty applies only to setup checks. By default, uncertainty applies to both setup and hold checks</td>
</tr>
<tr>
<td>-hold</td>
<td>Indicates that uncertainty applies only to hold checks. By default, uncertainty applies to both setup and hold checks</td>
</tr>
</tbody>
</table>

**Example:**
For Certus-NX according to the datasheet, output clock cycle-to-cycle jitter is 250ps for clocks >=200MHz. The constraint for this can be:

```
set_clock_uncertainty -setup 0.25 [get_clocks <clock>]
```

3.1.4.1. **Usage Guidelines**
Check the datasheet for jitter values to be used for the target device and use that number for clock uncertainty.

3.1.5. **set_clock_group Constraint**
The set_clock_group constraint specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during timing analysis. The syntax for set_clock_groups constraint is:

```
set_clock_groups -group <clock_list> -logically_exclusive | -physically_exclusive | -asynchronous>
```

Table 3.6. set_clock_group Constraint Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-logically exclusive</td>
<td>Both the clocks(or clock groups) can exist (can be active) simultaneously in the design and the clocks have SI impact on each other's clock domain</td>
</tr>
<tr>
<td>-physically exclusive</td>
<td>Both the clocks(or clock groups) cannot exist (cannot be active) simultaneously in the design and the clocks have no SI impact on each other's clock domain</td>
</tr>
<tr>
<td>-asynchronous</td>
<td>Specifies that the clock groups are asynchronous to each other (while the Radiant software assume all clocks defined by create_clock and create_generated_clock are synchronous)</td>
</tr>
</tbody>
</table>

**Example:**

```
set_clock_groups -asynchronous -group [get_clocks clka_port] -group [get_clocks clkb_port]
```

3.1.5.1. **Usage Guidelines**
- This constraint must be used only if the user is sure of the clock exclusiveness.
- If the clocks are not fully exclusive, consider using other clock exception constraints.
3.2. Input/Output Delay Constraints

Table 3.7 lists the input/output delay constraints supported in Radiant.

Table 3.7. Input/output Delay Constraints Supported in Radiant

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_input_delay</td>
<td>Sets input delay on input ports with respect to a clock signal</td>
</tr>
<tr>
<td>set_output_delay</td>
<td>Sets output delay on output ports with respect to a clock signal</td>
</tr>
</tbody>
</table>

3.2.1. set_input_delay Constraint

In Radiant, to constrain the input ports of the FPGA, set_input_delay constraint must be specified relative to a clock. The set_input_delay constraint is used to specify the external delay of the signal to the fabric input port. The syntax for set_input_delay constraint is:

```
set_input_delay -clock <clock_name>
    [-clock_fall]
    [-max]
    [-min]
    [-add_delay]
    <delay> <port_list>
```

Table 3.8. set_input_delay Constraint Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-clock_fall</td>
<td>Specifies that the delay is relative to the falling edge of the clock.</td>
</tr>
<tr>
<td>-min</td>
<td>Used to specify the best case for the signal at the input port. <strong>Used to perform hold checks.</strong></td>
</tr>
<tr>
<td>-max</td>
<td>Used to specify the worst case for the signal at the input port. <strong>Used to specify Setup checks.</strong></td>
</tr>
<tr>
<td>-add_delay</td>
<td>Used to define more than one input delay on a given port.</td>
</tr>
</tbody>
</table>

*Note:* When -min or -max are not specified, the same specified value is considered for both setup and hold calculation.

**Example:**

Let us consider the Figure 3.5. In this example, port A is an input signal to the FPGA which driven by an external design outside the FPGA.

![Figure 3.5. FPGA Driven by an External Design Outside the FPGA](image-url)
Assuming the input clock Clk is 100 MHz and output data from the external design takes 1 ns to reach the FPGA input port A, user can use the following constraints to constraint input port A:

```bash
create_clock -name clk -period 10 [get_ports Clk]
set_input_delay -clock clk 1 [get_ports A]
```

**Note:** If the external clock relationship is unknown, then clk must be treated as a virtual clock.

### 3.2.2. set_output_delay Constraint

In Radiant, to constrain the output ports of the FPGA, set_output_delay constraint must be specified relative to a clock. The syntax for set_output_delay constraint is:

```bash
set_output_delay -clock <clock_name>
    [-clock_fall]  
    [-max]  
    [-min]  
    [-add_delay] <delay>  
    <port_list>
```

**Table 3.9. set_output_delay Constraint Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-clock_fall</td>
<td>Specifies that the delay is relative to the falling edge of the clock</td>
</tr>
<tr>
<td>-max</td>
<td>Used to specify the worst case for the signal at the output port. Used to perform setup checks</td>
</tr>
<tr>
<td>-min</td>
<td>Used to specify the best case for the signal at the output port. Used to specify hold checks</td>
</tr>
<tr>
<td>-add_delay</td>
<td>Used to define more than one output delay on a given port</td>
</tr>
</tbody>
</table>

**Note:** When -min or -max are not specified, the same specified value is considered for both setup and hold calculation.

#### Example 1: Case when external circuit is driven by external clocks

**Things to consider:**
- Identify relationship between external clock and clock going into FPGA and set a clock latency on the clock input pin.
- Compute the external requirement by figuring out the delay to the clock pin of the external sequential element and the delay from the FPGA output to the external sequential element.
- Define the set_output_delay using the external clock and the external delays.

Consider the circuit in **Figure 3.6**.
Based on the above points, the following constraints can be applied to the above circuit:

```
create_clock –period T –name CLK1
create_clock –period T –name CLK2 [get_ports CLK]
set_clock_latency t5 [get_clocks CLK2]
set_input_delay t1 + t2 –clock [get_clocks CLK1] [all_inputs]
create_clock –period T –name CLK3
set_output_delay t3 + t4 –t6 –clock [get_clocks CLK3] [all_outputs]
```

**Example 2: Case when clock from inside the FPGA drives external clock**

Things to consider:

- Define a generated clock on the port where the clock comes out.
- Compute external requirements by finding the clock and data delays to the external sequential element.
- Define the set_output_delay constraint using the generated clock and the external delays.

Consider the circuit in Figure 3.7.

![Figure 3.7. Clock from Inside the FPGA Drives External Clock](image)

Based on the above points, the following constraints can be applied to the above circuit:

```
create_clock –period T –name CLK1
create_clock –period T –name CLK2 [get_ports CLK]
set_clock_latency t5 [get_clocks CLK2]
set_input_delay t1 + t2 –clock [get_clocks CLK1] [all_inputs]
create_generated_clock -name clk3 -source [pll/lscc_pll_inst/gen_no_refclk_mon.u_PLL.PLL_inst/CLKOS] [get_ports Clk_out] # assuming the clock was generated using a PLL
set_output_delay t3 + t4 –clock [get_clocks CLK3] [all_outputs]
```

### 3.3. Timing exception constraints

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_false_path</td>
<td>Used to specify paths that are considered false and excluded from timing analysis</td>
</tr>
<tr>
<td>set_max_delay</td>
<td>Specifies the maximum delay for the timing paths</td>
</tr>
<tr>
<td>set_min_delay</td>
<td>Specifies the minimum delay for the timing paths</td>
</tr>
<tr>
<td>set_multicycle_path</td>
<td>Defines Path that takes multiple clock cycles</td>
</tr>
</tbody>
</table>
3.3.1. set_false_path Constraint

set_false_path constraint is used to specify paths that have to be excluded from timing analysis. For example, paths that cross clock domain and the clocks are asynchronous to each other can be excluded from timing analysis by specifying set_false_path constraint between them. The syntax for set_false_path constraint is:

\[
\text{set_false_path} \quad [[\text{-from} \mid \text{rise_from} \mid \text{fall_from}<\text{object_list}>] \\
\text{-through} <\text{object_list}>] \\
[[\text{-to} \mid \text{-rise_to} \mid \text{-fall_to} <\text{object_list}>]
\]

<table>
<thead>
<tr>
<th>Table 3.11. set_false_path Options</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-from</td>
<td>Specifies start points (clocks, ports, pins, or cells) of disabled paths</td>
</tr>
<tr>
<td>-to</td>
<td>Specifies start points (clocks, ports, pins, or cells) of disabled paths</td>
</tr>
<tr>
<td>-rise_from</td>
<td>Same as -from but the path must rise from the specified object</td>
</tr>
<tr>
<td>-fall_from</td>
<td>Same as -from but the path must fall from the specified object</td>
</tr>
<tr>
<td>-rise_to</td>
<td>Same as -to but the path must rise to the specified object</td>
</tr>
<tr>
<td>-fall_to</td>
<td>Same as -to but the path must fall to the specified object</td>
</tr>
<tr>
<td>-through</td>
<td>Specifies false paths through nets</td>
</tr>
</tbody>
</table>

Example: set_false_path -from clk1 -to clk2

3.3.2. set_max_delay Constraint

This constraint is used to specify maximum delay for the timing paths. The syntax for set_max_delay constraint is:

\[
\text{set_max_delay} \quad [[\text{-from}<\text{object} \mid \text{cell_object}>] \\
\text{-through} <\text{object} \mid \text{cell_object}>] \\
[[\text{-to}<\text{object} \mid \text{cell_object}>] \\
\text{-datapath_only} \\
<\text{delay_value}>
\]

<table>
<thead>
<tr>
<th>Table 3.12. set_max_delay Options</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-from</td>
<td>Specifies start points (clocks, ports, pins, or cells) of disabled paths</td>
</tr>
<tr>
<td>-to</td>
<td>Specifies start points (clocks, ports, pins, or cells) of disabled paths</td>
</tr>
<tr>
<td>-through</td>
<td>Specifies false paths through nets</td>
</tr>
<tr>
<td>-datapath_only</td>
<td>Considers datapath only for timing calculation</td>
</tr>
</tbody>
</table>

3.3.3. set_min_delay Constraint

This constraint is used to specify minimum delay for the timing paths. The syntax for set_min_delay constraint is:

\[
\text{set_min_delay} \quad [[\text{-from}<\text{object} \mid \text{cell_object}>] \\
\text{-through} <\text{object} \mid \text{cell_object}>] \\
[[\text{-to}<\text{object} \mid \text{cell_object}>] \\
<\text{delay_value}>
\]
Table 3.13. set_min_delay Constraint Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>-from</td>
<td>Specifies start points (clocks, ports, pins, or cells) of disabled paths</td>
</tr>
<tr>
<td>-to</td>
<td>Specifies start points (clocks, ports, pins, or cells) of disabled paths</td>
</tr>
<tr>
<td>-through</td>
<td>Specifies false paths through nets</td>
</tr>
</tbody>
</table>

3.3.4. set_multicycle_path Constraint

This constraint is used to define paths that take multiple clock cycles. The syntax for set_multicycle_path constraint is

```
set_multicycle_path <ncycles>
[(-from | rise_from | fall_from)<object_list>]
[[-through <object_list>]
[(<to | -rise_to | -fall_to) <object_list>]
[-setup | -hold] [-start | -end]
<delay_value>
```

Table 3.14. set_multicycle_path Constraint Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ncycles</td>
<td>Number of cycles the datapath must have for setup check</td>
</tr>
<tr>
<td>-from</td>
<td>Specifies start points (clocks, ports, pins, or cells) of disabled paths</td>
</tr>
<tr>
<td>-to</td>
<td>Specifies start points (clocks, ports, pins, or cells) of disabled paths</td>
</tr>
<tr>
<td>-through</td>
<td>Specifies false paths through nets</td>
</tr>
</tbody>
</table>

Example: set_multicycle_path -from [get_cells {rd_grey_sync_r[*]}] 2
4. Constraints Entry in Radiant
This section describes the various ways users can enter constraints in the Lattice Radiant design software.

4.1. Using Pre-Synthesis Timing Constraints Editor
The Pre-synthesis Timing Constraints Editor (TCE) can be accessed from the Tools menu. This Pre-Synthesis TCE can be used to enter logical domain timing constraints based on the user design. When Pre-Synthesis TCE is invoked, Radiant compiles the user design and opens a window where users can constrain the design objects such as ports, pins, registers, and nets with various supported timing constraints and attributes. The entered constraints can then be saved into an LDC file for LSE flow or into an SDC file for LSE or Synplify Pro flows. Although, there may be some limitations of using the SDC file format, the advantage of using an SDC file format is it is supported by both LSE and Synplify Pro tools. Please refer to Radiant help “Unified Constraints Flow → Known Limitations” for the list of limitations.

![Figure 4.1. Pre-Synthesis Timing Constraints Editor Tools Menu](image)

TCE also has a Design Rule Check (DRC) function which lets the user know if the constraint is incorrect or if the entered design object is not valid. DRC checks are performed two ways. The first one is the Real time DRC checks. These checks are enabled by default and checks the validity of the constraint and objects in real time. The second DRC check happens when users save the constraints into an LDC or SDC file. These setting can be changed from the Tools menu: Tools → Options → Tools → Timing Constraints Editor. Here, users can change the behavior of the DRC checks.
4.2. Using Post-Synthesis Timing Constraints Editor

The Post-synthesis Timing Constraints Editor (TCE) will be available post synthesis and can be accessed from the Tools menu as shown in the figure below. The Post-Synthesis TCE can be used to enter physical domain timing constraints. When Post-Synthesis TCE is invoked, Post-Synthesis TCE reads in the post-synthesis database (*impl_1_syn.udb) file to populate all the post-synthesis netlist objects in the physical domain. The entered constraints can then be saved into a PDC file with an extension of .pdc. This PDC file will then be consumed by MAP and PAR processes.

The Post-Synthesis TCE also displays auto-derived device constraints and propagated constraints from a pre-synthesis constraints file which will be greyed out. The greyed-out constraints cannot be edited. To edit a greyed-out constraint, copy the constraint by right-clicking on the constraint and choose “copy constraint”. This pastes the constraint on the next line.

**Note:** The auto generated constraints cannot be removed from the Post-Synthesis TCE.

![Figure 4.2. Post-synthesis Timing Constraints Editor Tools Menu](image)

![Copy Constraint](image)
4.3. Using manual constraints entry

Constraints can also be entered manually using the source editor. To create a constraints file, from the File List Window, right-click on the Pre-synthesis/Post-Synthesis Constraints file → Add → New File.

![Create a Constraints File](image)

Figure 4.4. Create a Constraints File

Here, users can add LSE Design Constraints File (LDC file) if LSE flow is used, Post-synthesis Constraint Files (PDC file) or Pre-Synthesis Constraint Files (SDC Files) which supports both LSE and Synplify Pro.

![Adding LSE Design Constraints File](image)

Figure 4.5. Adding LSE Design Constraints File

Apart from this, users can also import an existing LDC, SDC or a PDC file choosing Add → Existing File option from the File List Window.
5. Timing Constraints effect on Implementation Process

Timing Constraints are used for timing driven synthesis and Place and Route (PAR). In Lattice Radiant software tool flow, Synplify Pro, LSE, and PAR engine are all timing driven by default. To maximize the effect of timing driven synthesis, make sure that the LSE strategy option “Optimization Goal” set to Timing and Synplify Pro strategy option “Area” unchecked. This section describes the effect of timing constraints on the implementation process.

5.1. Timing Constraints effect on Synthesis

In Radiant, both LSE and Synplify Pro flows are timing driven. Synthesis tools use user provided pre-synthesis timing constraints to optimize the circuit during synthesis process. The timing driven behavior is turned on by default in Radiant. This behavior can be changed to area driven using the strategy options.

Consider the below example. In this circuit, let us assume that there is a critical path in the logic driven by the signal X. Now, since the synthesis tools are timing driven, they apply timing driven optimization techniques as much as possible to help eliminate the critical paths wherever possible. This circuit preserves all the logic and eliminates the critical path.

If users do not provide any pre-synthesis constraints, synthesis uses a default of 200 MHz for Nexus and Avant devices for all the clocks in the design and will try to optimize the performance of the design for this frequency. This default frequency can be changed using the synthesis strategy options for both LSE and Synplify Pro.

Figure 5.1. Synthesis Strategy Options for Both LSE and Synplify Pro
One of the advantages of using pre-synthesis constraints file is to over constraining synthesis to meet the desired fmax after implementation. Since synthesis is timing driven, the user can constrain synthesis tightly by specifying a faster clock and relax it during PAR to meet the timing requirement.

5.2. Timing Constraints effect on MAP

In Radiant, timing constraints does not have any effect on MAP optimizations. For Nexus devices, Radiant MAPPER packs LUTs and direct flip flop loads during MAP. MAP also converts all the timing constraints into SLICE level constraints and saves it into the MAP UDB which then goes as an input to Place and Route.

5.3. Timing Constraint effect on Place and Route

Radiant Place and Route engine is timing driven by default. Place and Route engine uses timing constraints provided by the user to optimally place and route the design to meet user performance goals. The Place and Route engine uses advanced optimization techniques to try its best to correct any hold time violations from the user design. Please note that the timing constraints provided by the users also affect the Place and route run time. The tighter the constraints, the longer the runtime. The default behavior of timing driven place and route can be disabled using the Place and Route Design strategy options.

![Figure 5.2. Place and Route Design Strategy Options](image-url)
6. User constraints storage during the implementation process

The constraints provided by the users are processed in each of the implementation stages and are stored in the respective unified design database (UDB) files. Table 6.1 provides the information on the contents stored in the UDB after each of the implementation stage.

Table 6.1. Contents stored in UDB

<table>
<thead>
<tr>
<th>Processed Stage</th>
<th>Constraints Storage</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthesis</td>
<td>Post Synthesis UDB</td>
<td>Logical Netlist + Timing Constraints</td>
</tr>
<tr>
<td>MAP</td>
<td>MAP UDB</td>
<td>Physical Netlist + Timing Constraints</td>
</tr>
<tr>
<td>PAR</td>
<td>PAR UDB</td>
<td>Physical Netlist + Routing + Timing Constraints</td>
</tr>
</tbody>
</table>
7. Constraints Propagation Engine

Constraints Propagation Engine is a feature in Radiant SW tool designed to propagate sub-hierarchical constraints and to resolve conflicting constraints between user constraints and IP constraints. Constraints Propagation Engine or CPE compiles all input constraints from multiple .ldc files from IP and user constraints and creates an effective .ldc file which will be consumed by the synthesis tools. CPE executes right before synthesis begins and requires no user action. Please note that CPE only executes when a user design includes a .ipx with ldc/sdc files present. CPE flow does not support .fdc files.

7.1. CPE rules

Constraints propagation Engine is executed only when there is an IP with ldc constraints file is instantiated in the user design. CPE does not resolve conflicts within user constraints. User constraints conflicts are handled by Radiant timer. Constraints resolution rules applies only if there is a conflict between user constraints and IP constraints.

**Rule 1:** Create_clock constraint on an IP port will be ignored

- **Constraint example 1 on an IP port:** create_clock -name (clk_ip_a) –period 10 [get_ports clk]
- **Constraint example 2 on an IP port:** create_clock -name (clk_ip_b) –period 10 [get_ports ports]
- **Resolution:** IP level create clock constraint is Ignored.
- **Reason:** Clocks should always be defined on the top-level ports
  - Clocks on the input side of an IP are clocks that could potentially drive other circuits. In addition, such clocks could give rise to incorrect slack calculations at input ports, output ports and inter-clock paths if defined on the IP.
- **User Action:** Re-define the clock constraint on the top module ports.
- **Example:** create_clock -name clk -period 10 [get_clocks clk]

**Rule 2:** Input/output delay constraint on an IP port will be ignored

- **Constraint:** set_inputDelay -clock [get_clock virt_clk] 9 [get_ports in]
- **Resolution:** Ignored.
- **Reason:** IP Level input delay is not propagated.
- **User Action:** Re- define the constraint at the top-level input ports.
- **Example:** set_inputDelay -clock [get_clock virt_clk] 9 [get_ports in]

**Note:** If input/output delay constraints on IP ports come with pads (IO Buffers), only then the constraint will be propagated.

**Rule 3:** set_clock_groups constraints will be ignored

- **Constraint:** set_clock_groups [get_clocks clk] -group [get_clocks clk]
- **Resolution:** Ignored.
- **Reason:** Clock group constraints may be hazardous if these clocks are used in other parts of the design which needs to be timed.

**Rule 4:** set clock latency constraint on an IP clock will be ignored

- **Constraint:** set_clock_latency 3 -source [get_clocks clk]
- **Resolution:** Ignored
- **Reason:** Clock latency constraint is not propagated.
- **User Action:** Re- define the constraint on the top level clock.
Table 7.1 provides information on the CPE rules for ignored constraints.

**Table 7.1. CPE Rules for Ignored Constraints**

<table>
<thead>
<tr>
<th>Constraint Input</th>
<th>Source Module</th>
<th>Resolution Status</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_clock -name {clk_ip_a} –period 10 [get_ports clk]</td>
<td>IP</td>
<td>Ignored</td>
<td>Clocks on the input side of an IP are clocks that could potentially drive other circuits. In addition, such clocks could give rise to incorrect slack calculations at input ports, output ports and inter-clock paths if defined on the IP.</td>
</tr>
<tr>
<td>create_clock -name {clk_ip_b} –period 10 [get_ports clk]</td>
<td>IP</td>
<td>Ignored</td>
<td></td>
</tr>
<tr>
<td>set_clock_groups [get_clocks clk] –group [get_clocks clk2]</td>
<td>IP</td>
<td>Ignored</td>
<td>Apps working with IP team to replace all Lattice IP constraints containing set_clock_groups constraint with set_false_path constraint.</td>
</tr>
<tr>
<td>set_clock_latency 3 –source [get_clocks clk]</td>
<td>IP</td>
<td>Ignored</td>
<td>Clock latency constraint is not propagated.</td>
</tr>
</tbody>
</table>

Table 7.2 provides information on the CPE rule for resolved constraints.

**Table 7.2. CPE Rules for Resolved Constraints**

<table>
<thead>
<tr>
<th>Constraint Input</th>
<th>Source Module</th>
<th>Resolution Status</th>
<th>Constraint Output</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_clock –name {clk_top} –period 5 [get_ports clk_in]</td>
<td>Top</td>
<td>Resolved</td>
<td>create_clock –name {clk_top} –period 5 [get_ports gclk]</td>
<td>Constraint preserved</td>
</tr>
<tr>
<td>create_generated_clock –divide_by 2 –source [get_ports clkb] [get_pins b_out]</td>
<td>IP</td>
<td>Resolved</td>
<td>create_generated_clock –divide_by 2 –source [get_pins IP_B/clkb] [get_pins IP_B/b_out]</td>
<td>Propagated and name adjusted.</td>
</tr>
</tbody>
</table>
7.2. CPE Usage and Recommendation

- If the user has an IP instantiated in the design, run through the synthesis flow.
- Check IP constraints that are propagated by CPE using CPE generated Output files (next slide).
- Refer to the “Timing Constraints Resolution Summary” → User Action to “Keep the Constraint” section on Radiant help.

### Table 7.3. Timing Constraint Resolution Summary

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Constraint Input</th>
<th>Source Module</th>
<th>Resolution Status</th>
<th>Constraint Output</th>
<th>Resolution Method</th>
<th>User Action to Keep the Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>create_clock -name {clk_top} -period 5 [get_ports gclk]</td>
<td>Top</td>
<td>Resolved</td>
<td>Create_clock -name {clk_top} -period 5 [get_ports gclk]</td>
<td>Constraint preserved</td>
<td>No action needed.</td>
</tr>
<tr>
<td>2</td>
<td>create_clock -name {clk_ip_a} -period 10 [get_ports clk]</td>
<td>IP_A</td>
<td>Ignored</td>
<td>Constraint #1</td>
<td>Conflict with Constraint #1</td>
<td>Confirm satisfaction with top-level clock.</td>
</tr>
<tr>
<td>3</td>
<td>create_clock -name {clk_ip_b} -period 10 [get_ports clkb]</td>
<td>IP_B</td>
<td>Ignored</td>
<td>Defined on IP input Port</td>
<td>Ignore. Warn user</td>
<td>Redefine IP-level clock on appropriate top-level port.</td>
</tr>
<tr>
<td>4</td>
<td>create_generated_clock -divide_by 2 -source [get_ports clkb] [get_pinsb_out]</td>
<td>IP_B</td>
<td>Resolved</td>
<td>create_generated_clock -divide_by 2 -source [get_pins IP B/clkb] [get_pins IP B/b_out]</td>
<td>Propagated and name adjusted</td>
<td>No action needed.</td>
</tr>
<tr>
<td>5</td>
<td>set_input_delay -clock [get_clock sysclk] 9 [get_ports b_in]</td>
<td>IP_B</td>
<td>Ignored</td>
<td>Removed</td>
<td>IP-Level input delay not propagated</td>
<td>Redefine at top-level input port.</td>
</tr>
<tr>
<td>6</td>
<td>set_clock_groups [get_clocks_clk] -group [get_clocks clk2]</td>
<td>IP_B</td>
<td>Ignored</td>
<td>Removed</td>
<td>At least one clock is not internal</td>
<td>Translate to set_false_path and use appropriate user/custom IP-level objects.</td>
</tr>
<tr>
<td>7</td>
<td>set_max_delay -from [get_ports b_in] -to [get_ports b_out] 5</td>
<td>IP_B</td>
<td>Resolved</td>
<td>set_max_delay -from [get_pins IP_B/b_in] -to [get_pins IP_B/b_out] 5</td>
<td>Max and Min delay always propagated</td>
<td>No action needed.</td>
</tr>
</tbody>
</table>

7.3. CPE Output Files

The Constraints Propagation Engine generates some output files. This section provides information on the output files generated by CPE.

7.3.1. CPEreport.txt file

CPE generates a CPEreport.txt file which contains information on the removed and propagated constraints. This file can be found in the project implementation directory. Figure 7.1 gives an example of a CPEreport.txt file.
7.3.2. CPE generated .ldc file

CPE generates a .ldc file which is an effective constraints file after constraints propagation and resolution. This file is consumed by synthesis tools for synthesis. This file can be used to check propagated constraints. This file is also located in the project implementation directory. The file name for this .ldc file ends with *_impl_1_cpe.ldc. Figure 7.2 gives an example of a CPE generated .ldc file.

---

**Figure 7.1. Example of a CPEreport.txt file**

**Figure 7.2. Example of a CPE generated .ldc file**
References

For complete information on Lattice Radiant Project-Based Environment, Design Flow, Implementation Flow and Tasks, as well as on the Simulation Flow, refer to the Lattice Radiant software user guide.

You may also visit the following web pages at the Lattice website:

- Lattice Radiant Software
- Lattice Synthesis Engine
Technical Support Assistance
Submit a technical support case through www.latticesemi.com/techsupport.
For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.
## Revision History

### Revision 1.2, July 2023

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
</table>
| Clock Constraints   | • Changed output clock cycle-to-cycle jitter to 250ps for clocks >=200MHz.  
|                     | • Changed set clock uncertainty setup to 0.25.         |
| References          | Added this section.                                    |

### Revision 1.1, April 2023

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Updated document type from User Guide to Application Note.</td>
</tr>
</tbody>
</table>

### Revision 1.0, January 2023

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>