AN 008: Setting Trion™ Timing Constraints in the Efinity™ Software

AN008-v1.0
October 2018
www.efinixinc.com

Contents

Introduction

The Efinity® software provides a complete tool flow from RTL design to bitstream generation, and includes a built-in timing analysis tool you can use to perform static timing analysis on your designs. The software supports the Synopsys Design Constraints (SDC) format for specifying timing constraints. The Efinity® software validates the timing performance of your design's core logic using industry-standard constraint, analysis, and reporting methodology. During compilation, the Efinity® software generates a timing analysis report.

Trion™ FPGAs feature interface blocks—I/O logic and buffers, I/O banks, PLLs, etc.—that connect the core logic to the package pins. You use the Efinity® Interface Designer to configure these interface blocks for your design. After you configure these blocks, generate a constraints file. The software outputs report files for the interface blocks, which you can view in the Results tab in the Efinity® main window. To constrain timing, you use the <*design name*>**.pt_timing.rpt** report and an SDC template file <*design name*>**.pt.sdc**.

Unlike traditional FPGAs, you make timing constraints at the core level, not the interface or package level.

- For synchronous (registered) signals, the template defines clocks and sets input and output delays for your design. You simply copy and paste the relevant lines from the SDC template file to your own SDC file and adjust the timing as needed.
- For non-synchronous (unregistered) signals, you need to determine the interface timing and board timing and add those to your core settings.

This application note describes how to create timing constraints for Trion™ FPGA interface blocks.

Setting Clock Constraints

Trion™ FPGAs have interface blocks that serve as the clock source for the logic design. Clock sources can come from interface blocks like PLL or oscillators, or they can come from your board to the core through the GPIO pins.

Set Clock Constraints for Interface Blocks (Not GPIO)

1. Go to **Result** > **Interface** in the Efinity® dashboard.
2. Double-click <*design name*>**.pt.sdc** to open the SDC template file.
3. Copy the `create_clock` constraints and paste them into your constraints file.

Example SDC clock constraints:

```
create_clock -period 8.00 Fclk
create_clock -period 16.00 Sclk
```

Set Clock Constraints for GPIO Clocks

The Efinity SDC template includes a `create_clock` assignment for GPIO clocks, but you need to define the clock period.

1. Copy the assignment and paste it into your SDC file. For example, the assignment for GPIO `clk` would be `create_clock -period <USER_PERIOD> clk`.
2. Define the clock period. For example, `create_clock -period 5 clk`.

Setting Clock Uncertainty

Trion™ FPGAs have a default clock uncertainty for setup and hold analysis. For example, the T8 has 140 ps for setup and 50 ps for hold. You can modify these defaults by including the `set_clock_uncertainty` command in your SDC file.

Add 60 ps Clock Uncertainty

You want to add 60 ps to the default uncertainty for `clk`. Add this command to your SDC file:

```
set_clock_uncertainty -to clk -setup 0.06
```

The Efinity® software uses 200 ps of clock uncertainty for setup analysis.

Setting Input and Output Delay Constraints

When the interface block is synchronizing the connection to the core, the Interface Designer SDC template file includes the `set_input_delay` and `set_output_delay` SDC constraints. When it is not synchronized, you need to add external board delays to your constraint calculation.

Note: For Trion™ FPGAs, most interface connections are synchronous. The exceptions are GPIO blocks in bypass mode and LVDS blocks in x1 bypass mode.

Constraining Synchronous Inputs and Outputs

To set a constraint for synchronous inputs and outputs in your constraints file:

1. Go to **Result** > **Interface** in the Efinity® dashboard.
2. Double-click <*design name*>**.pt.sdc** to open the report.
3. Copy the `set_input_delay` and `set_output_delay` constraints and paste them into your constraints file.

Example set_output_delay Constraints

```
set_output_delay -clock Fclk -max -3.095 [get_ports {Fled[0]}]
set_output_delay -clock Fclk -min -2.944 [get_ports {Fled[0]}]
```

Constraining Unsynchronized Inputs and Outputs

Unsynchronized inputs and outputs are simple GPIO blocks in bypass mode or LVDS blocks in x1 bypass mode. For these blocks, you need to add external board delays to your SDC constraint calculation.

For blocks in bypass mode, the constraint clock is external to the FPGA:

- A receive clock is generated outside of the FPGA and is passed to the FPGA through a GPIO pin.
- A forward clock is generated by the FPGA and sent off chip though a GPIO pin in clock out mode.

Both receive and forward clocks synchronize the signal off chip.

For unsynchronized input or output signals, the GPIO block bypasses the register. `GPIO_IN` represents a combinational delay from the pad through the I/O buffer. `GPIO_OUT` represents a combinational delay to the pad through the I/O buffer from either the output or output enable signals.

A receive clock is passed to the FPGA design by configuring a GPIO in input mode and and setting the alternate connection to a global clock. `GPIO_IN_CLK` represents the combinational delay from the pad through the I/O buffer to the global clock.

A forward clock is generated by the FPGA design and sent off chip by configuring a GPIO in clkout mode. `GPIO_CLK_OUT` represents the combinational delay through the FPGA clock tree and the I/O buffer to the pad.

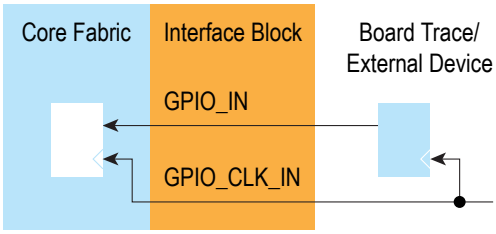
The general procedure for constraining unsynchronized inputs and outputs is:

1. Determine which mode you are constraining:
    - Input receive clock delay
    - Input forward clock delay
    - Output receive clock delay
    - Output forward clock delay
2. Find the mininum (fast) and maximum (slow) timing values in the Interface Designer report file **<*design name*>.pt_timing.rpt**.
3. Use formulas (provided in later sections) to calculate the delay.
4. Add the constraint to your SDC file.

Input Receive Clock Delay

This example shows how to set constraints for an input receive clock.

Figure 1: Receive Clock Delay (GPIO Input, Register Bypass)



The SDC constraint formulae for the receive clock delay are:

```
set_input_delay -clock <clock> -max <max calculation> <ports>
set_input_delay -clock <clock> -min <min calculation> <ports>
```

Where:
<max calculation> = *<max board constraint>* + $GPIO\_IN_{max}$ - $GPIO\_CLK\_IN_{max}$
<min calculation> = *<min board constraint>* + $GPIO\_IN_{min}$ - $GPIO\_CLK\_IN_{min}$

The following example shows how to calculate the delays and set the constraints.

Example: Constraining Input Receive Clock

You want to constrain the `din` input with respect to clock `clkin` with a max board constraint of 4 ns and a min board constraint of 2 ns. The non-registered GPIO configuration data from the Interface Designer timing report file is:

```
Non-registered GPIO Configuration:
==================================

+---------------+----------+-------------+----------+----------+
| Instance Name | Pin Name |  Parameter  | Max (ns) | Min (ns) |
+---------------+----------+-------------+----------+----------+
|     clkin     |  clkin   | GPIO_CLK_IN |  1.954   |  0.526   |
|      din      |   din    |   GPIO_IN   |  1.954   |  0.526   |
|      dout     |   dout   |   GPIO_OUT  |  4.246   |  1.081   |
+---------------+----------+-------------+----------+----------+
```

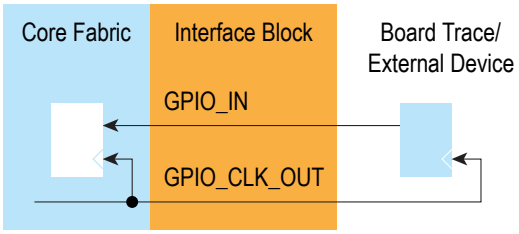
The equations are:
<max calculation> = 4 + 1.954 - 1.954 = 4
<min calculation> = 2 + 0.526 - 0.526 = 2
The resulting constraints are:

```
set_input_delay -clock clkin -max 4 din
set_input_delay -clock clkin -min 2 din
```

Input Forward Clock Delay

This example shows how to set constratints for an input forward clock.

Figure 2: Forward Clock Delay (GPIO Input, Register Bypass)



The SDC constraint formulae for the foward clock delay are:

```
set_input_delay -clock <clock> -max <max calculation> <ports>
set_input_delay -clock <clock> -min <min calculation> <ports>
```

Where:
<max calculation> = *<max board constraint>* + $GPIO\_IN_{max}$ + $GPIO\_CLK\_OUT_{max}$
<min calculation> = *<min board constraint>* + $GPIO\_IN_{min}$ + $GPIO\_CLK\_OUT_{min}$

The following example shows how to calculate the delays and set the constraints.

Example: Constraining Input Receive Clock

You want to constrain the `din` input with respect to clock `clkout` with a max board constraint of 4 ns and a min board constraint of 2 ns. The non-registered GPIO configuration data from the Interface Designer timing report file is:

```
Clkout GPIO Configuration:
==========================

+---------------+-----------+--------------+----------+----------+
| Instance Name | Clock Pin |   Parameter  | Max (ns) | Min (ns) |
+---------------+-----------+--------------+----------+----------+
|     clkout    |  pllclk0  | GPIO_CLK_OUT |  6.834   |  4.401   |
+---------------+-----------+--------------+----------+----------+

Non-registered GPIO Configuration:
==================================

+---------------+----------+-------------+----------+----------+
| Instance Name | Pin Name |  Parameter  | Max (ns) | Min (ns) |
+---------------+----------+-------------+----------+----------+
|     clkin     |  clkin   | GPIO_CLK_IN |  1.954   |  0.526   |
|      din      |   din    |   GPIO_IN   |  1.954   |  0.526   |
|      dout     |   dout   |   GPIO_OUT  |  4.246   |  1.081   |
+---------------+----------+-------------+----------+----------+
```

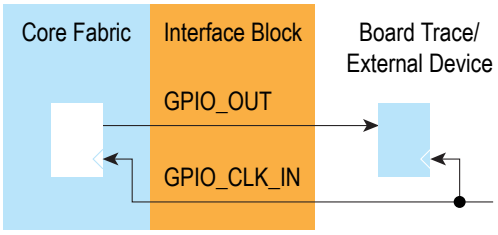
The equations are:
<max calculation> = 4 + 1.954 +6.834 = 12.788
<min calculation> = 2 + 0.526 + 4.401 = 6.927
The resulting constraints are:

```
set_input_delay -clock clkout -max 12.788 din
set_input_delay -clock clkout -min 6.927 din
```

Output Receive Clock Delay

This example shows how to set constraints for an output receive clock.

Figure 3: Receive Clock Delay (GPIO Output, Register Bypass)



The SDC constraint formulae for the receive clock delay are:

```
set_output_delay -clock <clock> -max <max calculation> <ports>
set_output_delay -clock <clock> -min <min calculation> <ports>
```

Where:
<max calculation> = *<max board constraint>* + $GPIO\_OUT_{max}$ + $GPIO\_CLK\_IN_{max}$
<min calculation> = *<min board constraint>* + $GPIO\_OUT_{min}$ + $GPIO\_CLK\_IN_{min}$

The following example shows how to calculate the delays and set the constraints.

Example: Constraining Input Receive Clock

You want to constrain the `dout` output with respect to clock `clkin` with a max board constraint of 4 ns and a min board constraint of 2 ns. The non-registered GPIO configuration data from the Interface Design report file is:

```
Non-registered GPIO Configuration:
==================================

+---------------+----------+-------------+----------+----------+
| Instance Name | Pin Name |  Parameter  | Max (ns) | Min (ns) |
+---------------+----------+-------------+----------+----------+
|     clkin     |  clkin   | GPIO_CLK_IN |  1.954   |  0.526   |
|      din      |   din    |   GPIO_IN   |  1.954   |  0.526   |
|      dout     |   dout   |  GPIO_OUT   |  4.246   |  1.081   |
+---------------+----------+-------------+----------+----------+
```

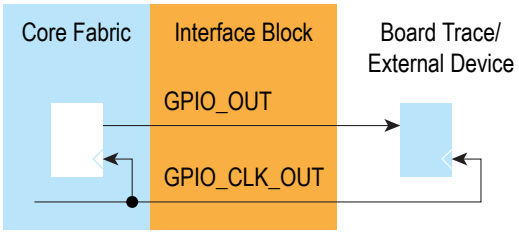
The equations are:
<max calculation> = 4 + 4.246 + 1.954 = 10.2
<min calculation> = 2 + 1.081 + 0.526 = 3.607
The resulting constraints are:

```
set_output_delay -clock clkin -max 10.2 dout
set_output_delay -clock clkin -min 3.607 dout
```

Output Forward Clock Delay

This example shows how to set constratints for an output forward clock.

Figure 4: Forward Clock Delay (GPIO Output, Register Bypass)



The SDC constraints for the forward clock delay is:

```
set_output_delay -clock <clock> -max <max calculation> <ports>
set_output_delay -clock <clock> -min <min calculation> <ports>
```

Where:
<max calculation> = *<max board constraint>* + $GPIO\_OUT_{max}$ - $GPIO\_CLK\_OUT_{max}$
<min calculation> = *<min board constraint>* + $GPIO\_OUT_{min}$ - $GPIO\_CLK\_OUT_{min}$

The following example shows how to calculate the delays and set the constraints.

Example: Constraining Input Receive Clock

You want to constrain the `dout` output with respect to clock `clkout` with a max board constraint of 4 ns and a min board constraint of 2 ns. The non-registered GPIO configuration data from the Interface Designer timing report file is:

```
Clkout GPIO Configuration:
==========================

+---------------+-----------+--------------+----------+----------+
| Instance Name | Clock Pin |   Parameter  | Max (ns) | Min (ns) |
+---------------+-----------+--------------+----------+----------+
|     clkout    |  pllclk0  | GPIO_CLK_OUT |  6.834   |  4.401   |
+---------------+-----------+--------------+----------+----------+

Non-registered GPIO Configuration:
==================================

+---------------+----------+-------------+----------+----------+
| Instance Name | Pin Name |  Parameter  | Max (ns) | Min (ns) |
+---------------+----------+-------------+----------+----------+
|     clkin     |  clkin   | GPIO_CLK_IN |  1.954   |  0.526   |
|      din      |   din    |   GPIO_IN   |  1.954   |  0.526   |
|      dout     |   dout   |   GPIO_OUT  |  4.246   |  1.081   |
+---------------+----------+-------------+----------+----------+
```

The equations are:
<max calculation> = 4 + 4.246 - 6.834 = 1.412
<min calculation> = 2 + 1.081 - 4.401 = -1.32
The resulting constraints are:

```
set_outputput_delay -clock clkout -max 1.412 dout
set_output_delay -clock clkout -min -1.32 dout
```

SDC Tips & Tricks

You can use wildcards, groups, and regular expressions to make it easier to assign constraints for interface blocks that have the same settings and board constraints.

In SDC syntax:

- # starts a comment; remaining text on this line is ignored.
- \ at the end of a line indicates that a command wraps to the next line.

Wildcard Commands

An * indicates a wildcard. Use * by itself to match all signals, or use it to create a partial wildcard. For example `clk*` would match `clk` and `clk2`.

Example: Constraining with Wildcards

You want to constrain all `Oled` signals with respect to clock `clk`. The resulting constraints are:

```
set_input_delay -max 10.214 -clock clk Oled*
set_input_delay -min 3.607 -clock clk Oled*
```

Regular Expressions

You can use regular expressions (in the Perl regular expression format) with the object specifier. You must encapsulate the object specifiers in square brackets []; arguments must be enclosed in curly braces {}.

To use Perl regular expressions, include the `-regexp` option in your command. Escape Perl regular expression characters if the provided string argument contains those characters. For example:

Simple wildcard:

```
get_pins y_r[*]~FF|D
```

Using Perl regular expressions:

```
regexp get_pins -regexp { y_r\[.*\]~FF|D }
```

Inverted Clocks

For an inverted external clock (one that uses the negative edge), include the clock_fall option in your set_input_delay or set_output_dalay command. For example:

```
set_output_delay -clock <clock> -clock_fall -max -3.1 <ports>
set_output_delay -clock <clock> -clock_fall -min -2.85 <ports>
```

Revision History

Table 1: Revision History

Date	Version	Description
October 2018	1.0	Initial release.