Synopsis Design Constraints

This lecture explains how to specify timing constraints in the form of SDC (Synopsys Design Constraint) commands. After this lecture you should be able to use SDC commands to correctly apply the constraints described in this lecture.

SDC Timing Constraints

A standard file format, Synopsys Design Constraint (SDC), is used to specify timing and other design constraints. The constraints are specified as tcl commands. This lecture covers the most common timing constraints and how they are specified in an SDC file.

The syntax details are available in the Quartus SDC editor, the TimeQuest timing analyzer GUI and the Quartus documentation.

Collections

We often want to set timing constraints on a collection of signals such as all of the bits in a bus.

When a module is instantiated unique signal names must be generated to avoid name conflicts. These names may be difficult to determine. For example the signal name `adcspi:a0|cnt.bitcnt[0]` may specify the `0`'th bit of the `bitcnt` field of the `cnt` signal of the `a0` instance of the `adcspi` module.

SDC files can use tcl functions that look up signal names by matching netlist names with patterns that can include an asterisk as a wildcard. For example, the command `get_nodes *bitcnt[*]` would return references to the matching signal names, including the one above.

Different `get_` functions can search for different type of signals[1]: `nodes` most netlist items, `cells` (registers), `pins` (cell i/o's), `nets` (connections between pins), `ports` (top-level i/o), `clocks` (clock signals – not necessarily in your design). For example, `get_ports a*` would return references to all top-level IO signals matching the pattern "a*."

As shown below, these commands are usually placed in square brackets so the result of the command can be used as an argument in another command.

[1]From Quartus documentation

Clock Constraints

Most designs will use one (or more) clock signals. These clocks' frequencies must be specified in the SDC file so that the timing analyzer can ensure that the setup and hold time requirements of flip-flops internal to the FPGA will be met. Many designs will also derive new clock signals either by dividing a clock or by using a phase-locked loop (PLL).

Clock constraints are the only constraints required for designs with simple (unclocked) IO such as LEDs and switches.

create_clock This specifies a clock signal. The frequency is specified, it is given a name and it can be associated with a particular net in the design. For example:

```
create_clock -name clk50 -period 20 \
     [get_ports {clk50}]
```

This adds timing paths that start at the port `clk50` and end at the `D` inputs of the flip-flops to which this clock is connected.

create_generated_clock Can be used to define clocks with a fixed relationship (frequency and/or phase) to another clock. For example:

```
create_generated_clock \
  -source [get_ports clk50]\
  -divide_by 2 -name sclk_int \
  [get_nodes sclk~reg0]
```

the `-source` specifies the location of the reference clock and the target is the location of the generated clock. Propagation delays from the source are included in clock path delay calculations.

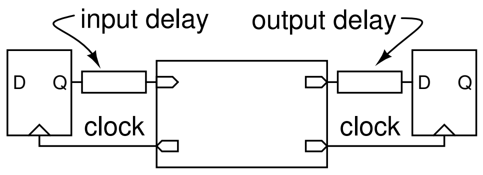
derive_pll_clocks This is an Intel-specific command that uses uses PLL instance parameters to create generated clocks corresponding to the PLL output(s). Most FPGA designs use PLLs to generate clocks and this is often the only clock constraint required.

Virtual Clocks A virtual clock is a generated clock that does not correspond to a pin or port in the design. Virtual clocks thus do not create timing paths within the design. Virtual clocks are primarily used to specify timing constraints for clocked devices outside the design. The syntax is as above but without the list of target pins.

IO Timing Constraints

Timing paths can start at input ports and/or end at output ports. In these cases the STA must be provided with the delays to and from what are, effectively, external flip-flop `D` inputs and `Q` outputs.

Since the STA does not know the external devices' setup and hold requirements, the designer must specify output delays to ensure the external device's requirements will be met. Similarly, the designer must specify input delays corresponding the external device's actual specifications to ensure the timing requirements of flip-flops driven by inputs will be met.



set_output_delay This SDC command specifies the propagation delay from an output port to an external device.

The `-max` option can be used to increase the maximum timing path delay and the `-min` option the minimum path delay. The `-max` option increases the maximum data arrival time and can be used to specify the external device's setup time. Similarly the `-min` option increases the minimum data arrival time and a negative value can be used to specify the external device's minimum hold time requirement. These values should be reduced by any external clock propagation delays.

For example:

```
set_output_delay -clock sclk \
     -max 15 [get_ports mosi]
```

adds a timing constraint based on data arrival time at `mosi` using clock `sclk` with a 15 ns subtracted from the maximum data required time (equivalent to adding it to the maximum data available time).

set_input_delay As above, but specifies the delay between an external clocked output and an input port. For example:

```
set_input_delay -clock sclk \
     -max 8 [get_ports miso]
```

adds a timing constraint to paths clocked by `sclk` and originating at `miso` with 8 ns added to the maximum data arrival time. This would correspond to the maximum t_{CO} of the external device.

Timing Exceptions

Some designs require additional constraints. See the timing analyzer documentation for more details. Some common examples include:

False Paths

Timing constraints cannot be meaningfully applied to timing paths with asynchronous launch and latch clocks. These are called "false paths" and should be identified to avoid misleading warnings.

A typical example is IO that is not clocked such as as an LED output or switch input.

The `set_false_path` command is used to remove timing paths from the timing analysis. For example:

```
set_false_path -from [get_ports key0]
```

Multicycles

In some cases the launching and latching clock edges can be separated by multiple clock cycles. The `set_multicycle_path` command can be used to allow a delay longer than one clock period.

Running STA

The SDC timing constraints are typically entered in a text file and applied to a timing-annotated netlist generated by the PAR tools. Then reports are generated showing any paths with negative slack.

It is easy to make mistakes when constraining timing. Waveform diagrams in reports can be used to verify that the clocks have been properly specified. Tables showing the details of how the data and clock arrival times are computed can be used to verify that the appropriate paths are being constrained.