



Quartus® II Software Design Series: Timing Analysis



Objectives

- Build SDC files for constraining PLD designs
- Verify timing on simple & complex designs using TimeQuest TA

Timing Analysis Agenda

- TimeQuest basics ←
- Timing constraints
- Example

TimeQuest GUI

Menu access to all TimeQuest features

Report Pane

Tasks Pane

View Pane

Console Pane

The screenshot shows the TimeQuest Timing Analyzer interface. The menu bar at the top includes File, Edit, View, Netlist, Constraints, Reports, Script, Tools, Window, and Help. The Reports menu is highlighted, showing options like TimeQuest Timing Analyzer Summary, SDC File List, Summary (Setup), Summary (Hold), and Report Timing. The Report pane displays a table of paths with slack values. The Tasks pane lists various reports and macros. The View pane shows a detailed timing diagram for a specific path, including launch clock, setup relationship, latch clock, data arrival, clock delay, data delay, and slack. The Console pane shows the command history and output.

Slack	From Node	To Node	Launch Clock	Latch Clock
1 1.045	inst2[15]	q[15]	clk1	clk1
2 1.045	inst2[13]	q[13]	clk1	clk1
3 1.045	inst2[9]	q[9]	clk1	clk1
4 1.045	inst2[1]	q[1]	clk1	clk1
5 1.045	inst2[2]	q[2]	clk1	clk1

Total	Incr	RF	Type	Fanout	Location	Element
1 0.000	0.000					...ge tir
2 1.297	1.297	R				...k del
3 1.547	0.250		uTco	1	Unassigned	inst2[1
4 1.547	0.000	RR	CELL	1	Unassigned	...lregc
5 1.547	0.000	RR	IC	1	Unassigned	...l)dat
6 4.255	2.708	RR	CELL	0	Unassigned	q[15]

Total	Incr	RF	Type	Fanout	Location	Element
1 6.000	6.000					...ge tir
2 6.000	0.000	R				...k del
3 5.300	-0.700	R	oExt	0	Unassigned	q[15]

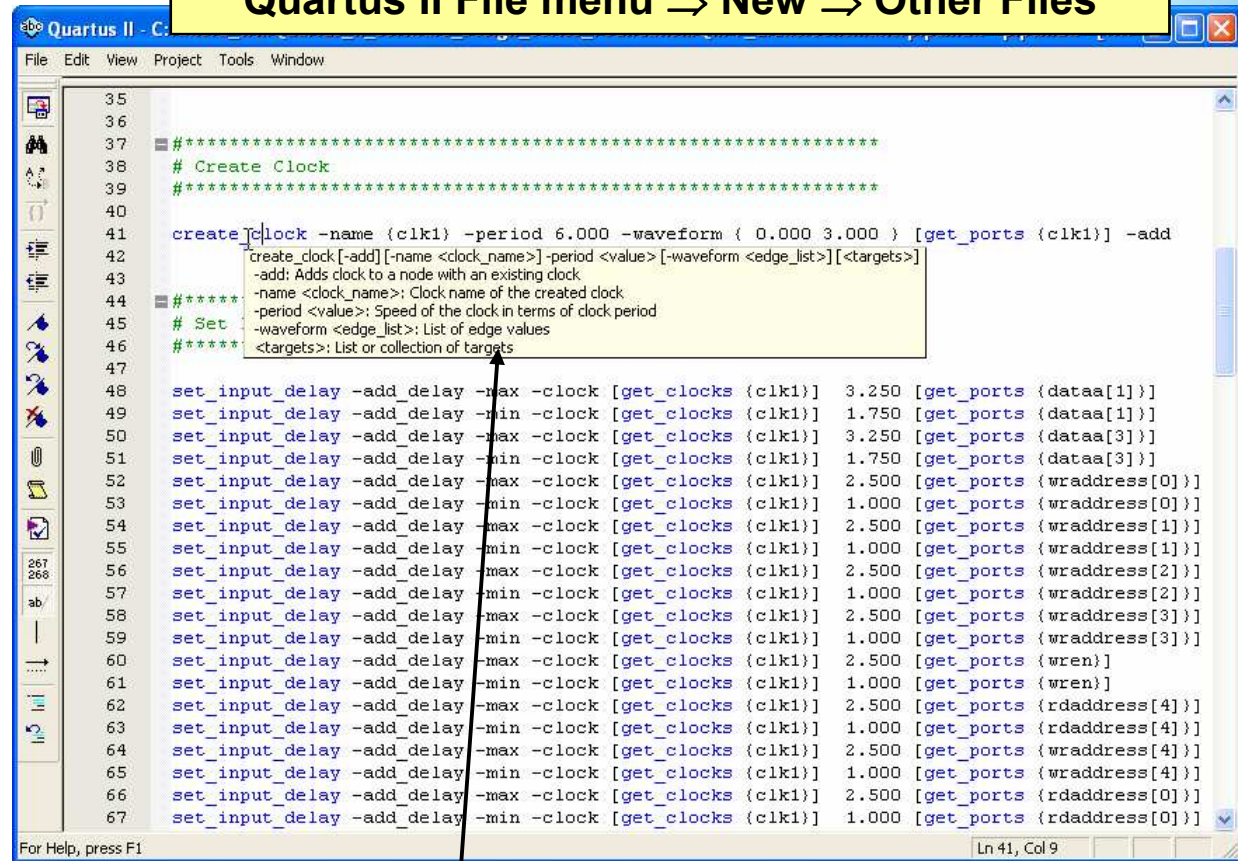
```

22: read_sdc "C:/altera_trn/Quartus_II_Software_Design_Series_Foundation/QIIF7_2/Solutions/Final_projects/Schematic/pipemult.sdc"
23: Info: Reading SDC File: 'C:/altera_trn/Quartus_II_Software_Design_Series_Foundation/QIIF7_2/Solutions/Final_projects/Schematic/pipemult.sdc'
24: update_timing_netlist;
25: create_timing_summary -setup -panel_name "summary (Setup)"
26: create_timing_summary -hold -panel_name "summary (Hold)"
27: report_timing -to_clock clk1 -setup -npaths 10 -detail path_only -panel_name {Report Timing}
28: Info: Report Timing: Found 10 setup paths (0 violated). Worst case slack is 1.045
33: 10 1.045
34:
  
```


SDC File Editor = Quartus II Text Editor

- Use Quartus II editor to create and/or edit SDC
- SDC editing unique features (for .sdc files)
 - Access to GUI dialog boxes for constraint entry (**Edit ⇒ Insert Constraint**)
 - Syntax coloring
 - Tooltip syntax help

TimeQuest File menu ⇒ New/Open SDC File
Quartus II File menu ⇒ New ⇒ Other Files



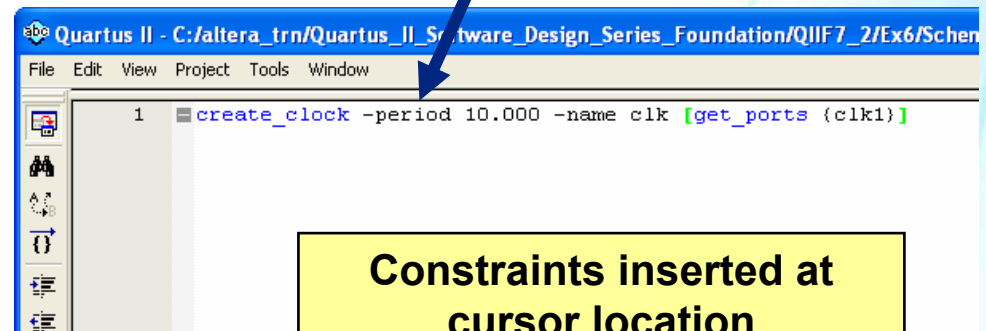
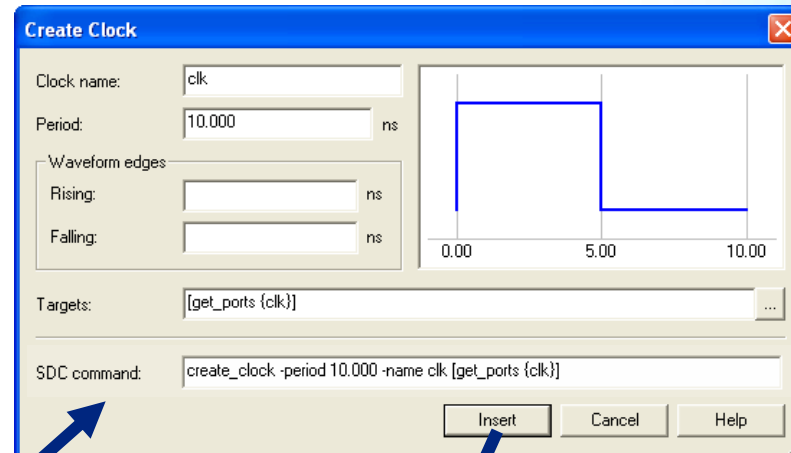
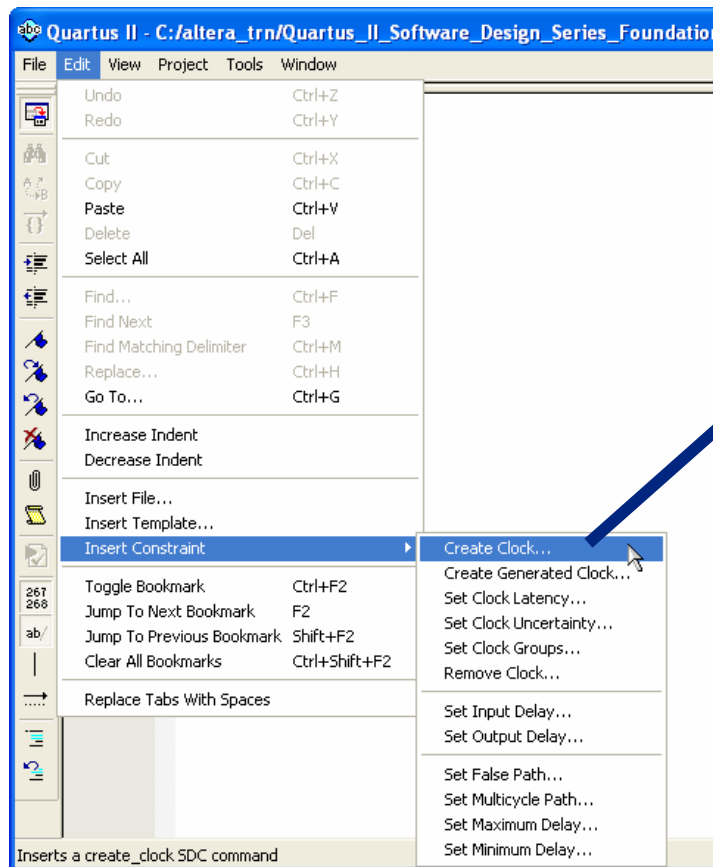
The screenshot shows the Quartus II SDC File Editor window. The menu bar includes File, Edit, View, Project, Tools, and Window. The code is syntax-highlighted. A tooltip is displayed over the 'create_clock' command on line 41, providing details about its arguments and usage. The code includes comments and various SDC commands like 'create_clock', 'set_input_delay', and 'set_output_delay'.

```
35
36
37
38 # Create Clock
39
40
41 create_clock -name {clk1} -period 6.000 -waveform { 0.000 3.000 } [get_ports {clk1}] -add
42
43
44
45 # Set
46
47
48 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 3.250 [get_ports {dataa[1]}]
49 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.750 [get_ports {dataa[1]}]
50 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 3.250 [get_ports {dataa[3]}]
51 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.750 [get_ports {dataa[3]}]
52 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 2.500 [get_ports {waddress[0]}]
53 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.000 [get_ports {waddress[0]}]
54 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 2.500 [get_ports {waddress[1]}]
55 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.000 [get_ports {waddress[1]}]
56 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 2.500 [get_ports {waddress[2]}]
57 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.000 [get_ports {waddress[2]}]
58 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 2.500 [get_ports {waddress[3]}]
59 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.000 [get_ports {waddress[3]}]
60 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 2.500 [get_ports {wren}]
61 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.000 [get_ports {wren}]
62 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 2.500 [get_ports {rdaddress[4]}]
63 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.000 [get_ports {rdaddress[4]}]
64 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 2.500 [get_ports {waddress[4]}]
65 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.000 [get_ports {waddress[4]}]
66 set_input_delay -add_delay -max -clock [get_clocks {clk1}] 2.500 [get_ports {rdaddress[0]}]
67 set_input_delay -add_delay -min -clock [get_clocks {clk1}] 1.000 [get_ports {rdaddress[0]}]
```

Place cursor over
command to see tooltip

SDC File Editor (cont.)

**Construct an SDC file using
TimeQuest graphical
constraint creation tools**

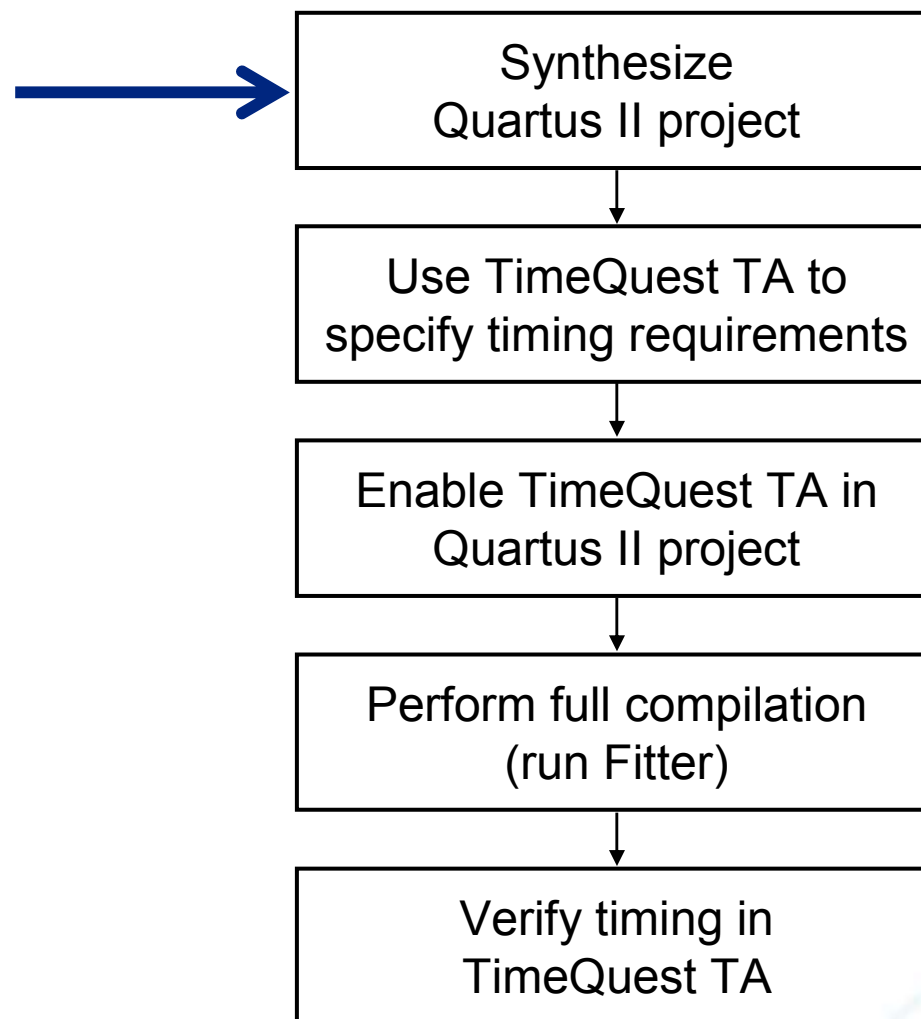


**Constraints inserted at
cursor location**

Basic Steps to Using TimeQuest TA

1. Generate timing netlist
2. Enter SDC constraints
 - a. Create and/or read in SDC file (recommended method)
 - or**
 - b. Constrain design directly in console
3. Update timing netlist
4. Generate timing reports
5. Save timing constraints (optional)

Using TimeQuest TA in Quartus II Flow



Timing Analysis Agenda

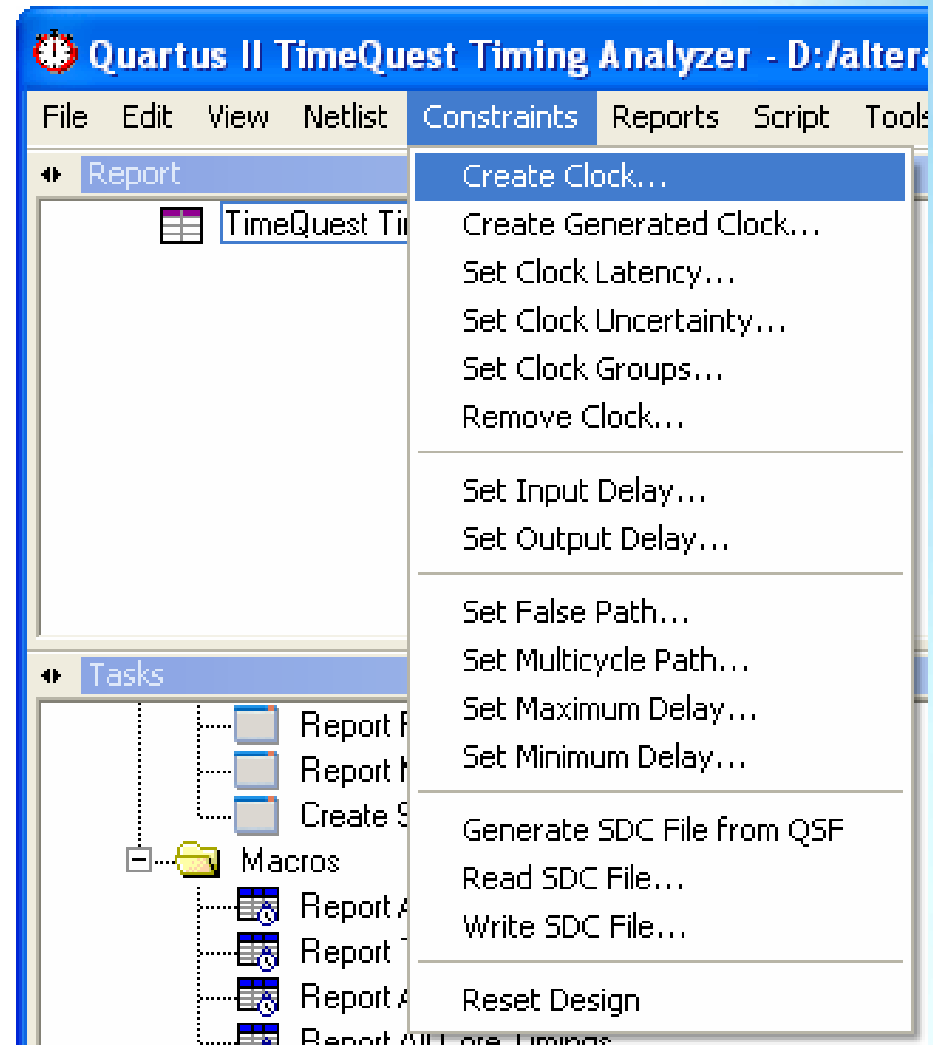
- TimeQuest basics
- Timing constraints ←
- Example

Importance of Constraining

- Timing analysis tells how a circuit **WILL** behave
- Providing timing constraints tells tools how you **WANT** the design to behave
 - Constraints paint picture of how design should operate
 - Based on design specs & specs from other devices on PCB
 - Provide goals for fitter to target during compilation
 - Provide values to which to compare timing results
- TimeQuest TA performs limited analysis without timing constraints

Timing Requirements: Enter Constraints

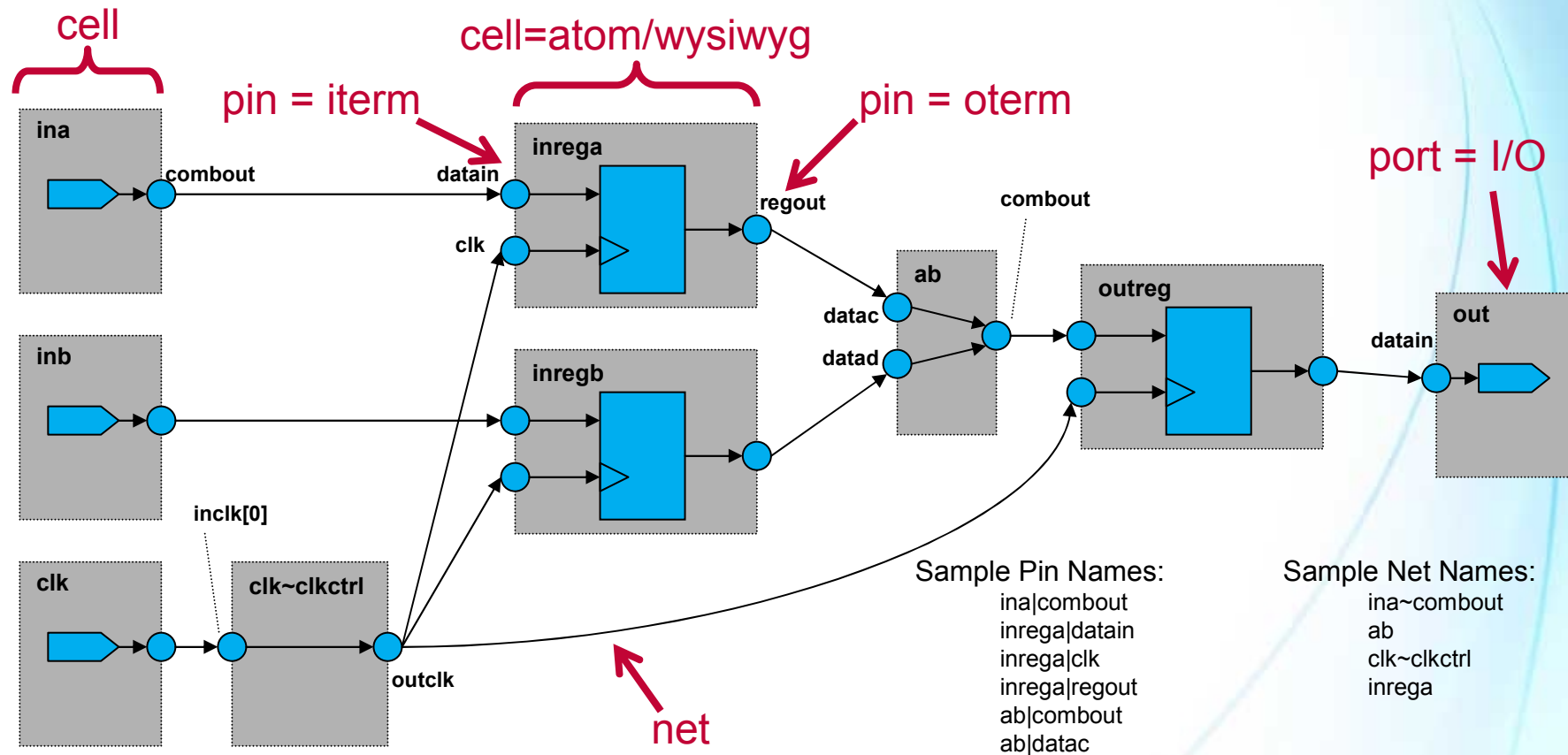
- All constraints discussed can be easily accessed in TimeQuest GUI
 - **Constraints** menu of TimeQuest
 - **Edit ⇒ Insert Constraint** menu of SDC File Editor



SDC Netlist Terminology

Term	Definition
Cell	Device building blocks (e.g. look-up tables, registers, embedded multipliers, memory blocks, I/O elements, PLLs, etc.)
Pin	Input or outputs of cells
Net	Connections between pins
Port	Top-level inputs and outputs (e.g. device pins)

SDC Netlist Example



- Paths defined in constraints by targeted endpoints (pins or ports)

Collections

- Searches and returns from the design netlist with a list of names meeting criteria
- Used in SDC commands
 - Some collections searched automatically during a command's usage and may not need to be specified
- Examples
 - `get_ports`
 - `get_pins`
 - `get_clocks`
 - `all_clocks`
 - `all_registers`
 - `all_inputs`
 - `all_outputs`

See “TimeQuest Timing Analyzer” chapter of the Quartus II Software Handbook (Volume 3) for a complete list & description of each

SDC Timing Constraints

- Clocks ←
- I/O
- False paths
- Multicycle paths

What are clocks in SDC?

- Defined, repeating signal characteristics applied to a point anywhere in the design
 - Internal: applied to a specific node being used as a clock in design (port or pin)
 - “Virtual”: No real source in, or direct interaction with design
 - Example: Clocks on external devices that feed or are fed by the FPGA design, required for I/O analysis
- Name clocks after node to which they are applied or something more meaningful
- Similar to clock settings in older Quartus II timing engine (Classic timing analyzer)

Clocks in SDC (cont.)

■ Two types

- Clock
 - Absolute or base clock
- Generated clock
 - Timing derived from another clock in design
 - Must have defined relation with source clock
 - Apply to output of logic function that modifies clock input
 - PLLs, clock dividers, output clocks, ripple clocks, etc.
 - Clock inversions automatically detected unless derived from more complex logic structure

■ ***All clocks are related by default***

- Cross-domain transfers analyzed

Clock Constraints

- Create clock
- Create generated clock
- PLL clocks
- Automatic clock detection & creation
- Default constraints
- Clock latency
- Clock uncertainty
- Common clock path pessimism removal

Creating a Clock

- Command: `create_clock`

- Options

- `[-name <clock_name>]`

- `-period <time>`

- `[-waveform {<rise_time> <fall_time>}]`

- `[<targets>]`

- `[-add]`

[] = optional

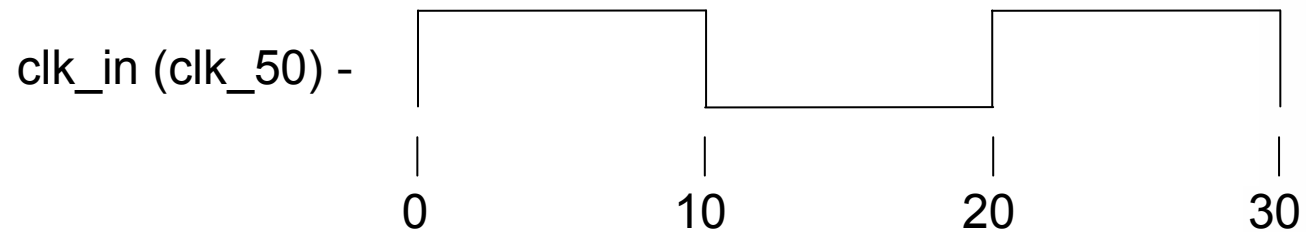
Note: In general, the more options added to a constraint command, the more specific the constraint is. When options are not specified, the constraint is more generalized and pertains to more of the target.

create_clock Notes

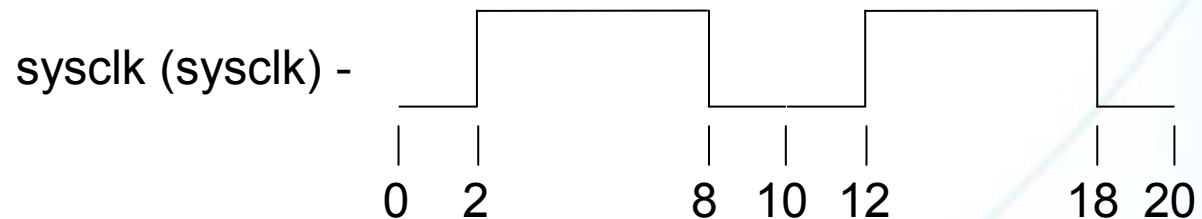
- `-name`: Assigns name to the clock to be used in other commands & reports when referring to clock
 - Optional; defaults to target name if not specified
- `-waveform`: Indicates clock offset or non-50% duty cycle clocks
 - 50% duty cycle is assumed unless otherwise indicated
- `-add`: Adds clock to node with existing clock
 - Without `-add`, warning given and subsequent clock constraints ignored
- `<targets>`: Target ports or pins for clock setting
 - Virtual clock created if no target specified

create_clock Examples

```
create_clock -period 20.0 -name clk_50 [get_ports clk_in]
```



```
create_clock -period 10.0 -waveform {2.0 8.0} [get_ports sysclk]
```



Create Clock using GUI

TimeQuest main: Constraints ⇒ Create Clock
SDC Editor: Edit ⇒ Insert Constraint ⇒ Create Clock

Create Clock


Clock name:

Period: ns

Waveform edges

Rising: ns

Falling: ns

Targets: 

SDC command:

**Edit any field
(change values; use wildcards
in targets or command)**

**Name Finder
(next slide)**

Name Finder

Clicking on Browse button opens Name Finder allowing you to search netlist for node names (similar to Quartus II Node Finder)

Name Finder

Collection: **get_ports** Filter: *

Options

- ☐ Case-insensitive
- ☐ Hierarchical
- ☐ Compatibility mode
- ☐ No duplicates

Matches

List

67 matches found

- clk_in_100mhz
- clkout
- din_a[0]
- din_a[1]
- din_a[2]
- din_a[3]
- din_a[4]
- din_a[5]
- din_a[6]
- din_a[7]
- din_b[0]
- din_b[1]
- din_b[2]
- din_b[3]

Navigation: > >> < <<

9 selected names

- clk_in_100mhz
- din_a[0]
- din_a[1]
- din_a[2]
- din_a[3]
- din_a[4]
- din_a[5]
- din_a[6]
- din_a[7]

SDC command: [get_ports {clk_in_100mhz din_a[0] din_a[1] din_a[2] din_a[3] din_a[4] din_a[5] din_a[6] din_a[7]}]

OK Cancel Help

Options available depend on selected collection

Select collection to search

Edit command here or final command to use wildcards

Creating a Generated Clock

- Command: `create_generated_clock`

- Options

```
[-name <clock_name>]
-source <master_pin>
[-master_clock <clock_name>]
[-divide_by <factor>]
[-multiply_by <factor>]
[-duty_cycle <percent>]
[-invert]
[-phase <degrees>]
[-edges <edge_list>]
[-edge_shift <shift_list>]
[<targets>]
[-add]
```


create_generated_clock Notes

- `-source`: Specifies the node in design from which generated clock is derived
 - Ex. Placing source before vs. after an inverter would yield different results
- `-master_clock`: Used if multiple clocks exist at source due to `-add` option
- `-edges`: Relates rising/falling edges of generated clock to rising/falling edges of source based on numbered edges
- `-edge_shift`: Relates edges based on amount of time shifted (requires `-edges`)

Create Generated Clock using GUI

Create Generated Clock

Clock name:

Source:

Relationship to source

☒ Based on frequency

Divide by:

Multiply by:

Duty cycle:

Phase:

Offset:

☐ Based on waveform

Edge list:

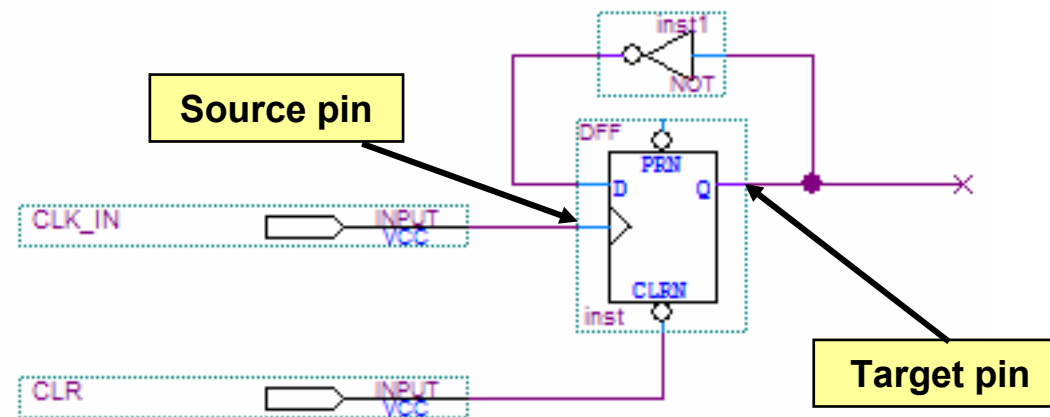
Edge shift list: ns ns ns

☐ Invert waveform

Targets:

SDC command:

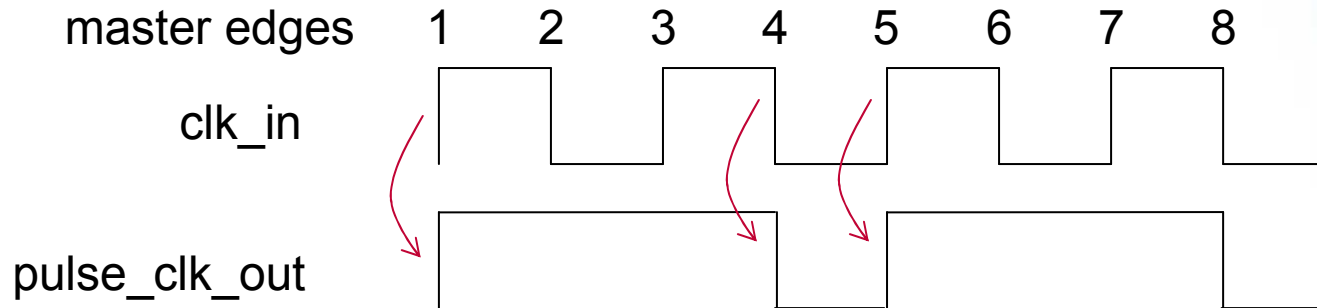
Generated Clock Example 1



```
create_clock -period 10 [get_ports clk_in]
```

```
create_generated_clock -name clk_div \  
    -source [get_pins inst|clk] \  
    -divide_by 2 \  
    [get_pins inst|regout]
```

Generated Clock Example 2

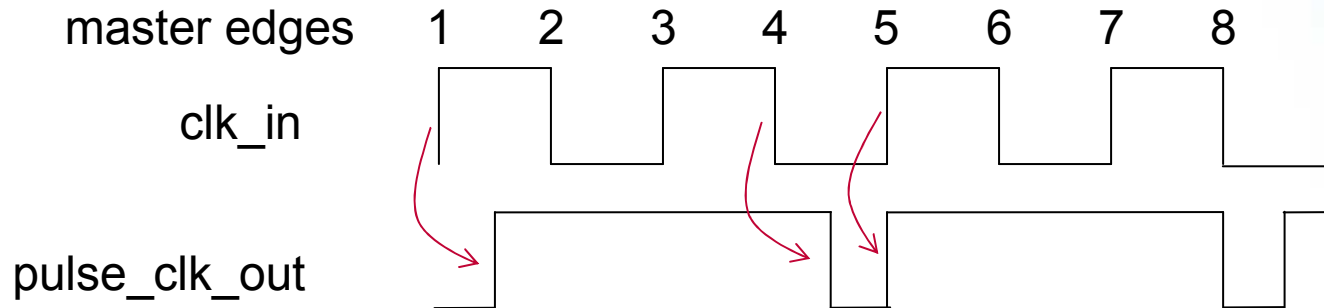


```
create_clock -period 10 [get_ports clk_in]
```

```
create_generated_clock -name pulse_clk_out -source clk_in \  
    -edges {1 4 5} \  
    [get_pins pulse_logic|out]
```

Master edges are numbered 1..<n>. In the edge list, the first
number corresponds to the first rising edge of the generated
clock. The second number is the first falling edge. The third
number is the second rising edge. Thus, a clock is created that
is half the period of the source with a 75% duty cycle.

Generated Clock Example 3



```
create_clock -period 10 [get_ports clk_in]
```

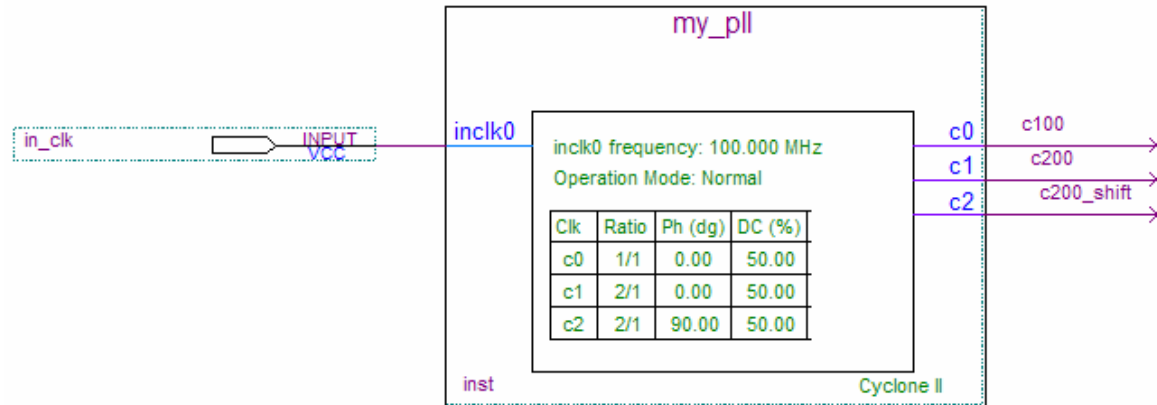
```
create_generated_clock -name pulse_clk_out -source clk_in \  
    -edges {1 4 5} -edge_shift {2.5 2.5 0} \  
    [get_pins pulse_logic|out]
```

```
# Same as example 2 except -edge_shift shifts each edge indicated  
# amount of time
```

PLL Clocks (Altera SDC Extension)

- **Command:** `derive_pll_clocks`
 - `[-use_tan_name]`: names clock after design net name from Classic timing analyzer settings instead of the default PLL output SDC pin name
 - `[-create_base_clocks]`: generates `create_clock` constraint(s) for PLL input clocks
- Create generated clocks on all PLL outputs
 - Based on input clock & PLL settings
- Requires defining PLL input as clock unless `-create_base_clocks` is used
- Automatically updates generated clocks on PLL outputs as changes made to PLL design
- `write_sdc -expand` expands constraint into standard `create_clock` and `create_generated_clock` commands
- *Not in GUI; must be entered in SDC manually*

derive_pll_clocks Example



Using generated clock commands

```
create_clock -period 10.0 [get_ports in_clk]
create_generated_clock -name c100 \
    -source [get_pins {inst|altpll_component|pll|inclk[0]}] \
    -divide_by 1 \
    [get_pins {inst|altpll_component|pll|clk[0]}]
create_generated_clock -name c200 \
    -source [get_pins {inst|altpll_component|pll|inclk[0]}] \
    -multiply_by 2 \
    [get_pins {inst|altpll_component|pll|clk[1]}]
create_generated_clock -name c200_shift \
    -source [get_pins {inst|altpll_component|pll|inclk[0]}] \
    -multiply_by 2 \
    -phase 90 \
    [get_pins {inst|altpll_component|pll|clk[2]}]
```

Using derive_pll_command

```
create_clock -period 10.0 \
    [get_ports in_clk]
derive_pll_clocks

# or simply:

derive_pll_clocks \
    -create_base_clocks

# Note the clock names for
# the generated clocks
# will be the names of
# the PLL output pins
```

Automatic Clock Detection & Creation

- Command: `derive_clocks`
 - `[-period]`: same use as with `create_clock`
 - `[-waveform]`: same use as with `create_clock`
 - No target required
- Automatically create clocks on clock pins in design that don't already have clocks defined
- Does not work with PLL outputs (use `derive_pll_clocks`)
- SDC extension expanded with `write_sdc -expand`
- *Not in GUI*
- *Not recommended for final timing sign-off*

Default Clock Constraints

- Remember, all clocks must be constrained to analyze design with timing analysis
- If no clock constraints exist, default constraints created through two commands

```
derive_clocks -period 1.0
```

```
derive_pll_clocks
```

- Default constraints not applied if at least one clock constraint exists

- *Not in GUI*

- *Not recommended for final timing sign-off*

Non-Ideal Clock Constraints

- So far, all clocks have been ideal
 - Nice square waves
 - No accounting for delays outside of FPGA
- Add extra constraints to define realistic, non-ideal clocks
- Three special constraints
 - `set_clock_latency`
 - `set_clock_uncertainty`
 - `derive_clock_uncertainty`

Clock Latency

- Two types of latency
 - Source: From clock source to input port (board latency)
 - Network: From input port to destination register clock pin
- Network latency handled and understood by timing analysis automatically
- Need to model source latency
 - TimeQuest TA knows nothing about delays external to device
- Provide a more realistic picture of external clock behavior
- Example
 - External feedback clock: need to specify delay from clock output I/O to clock input I/O
- Clocks created with `create_clock` have default source latency of 0

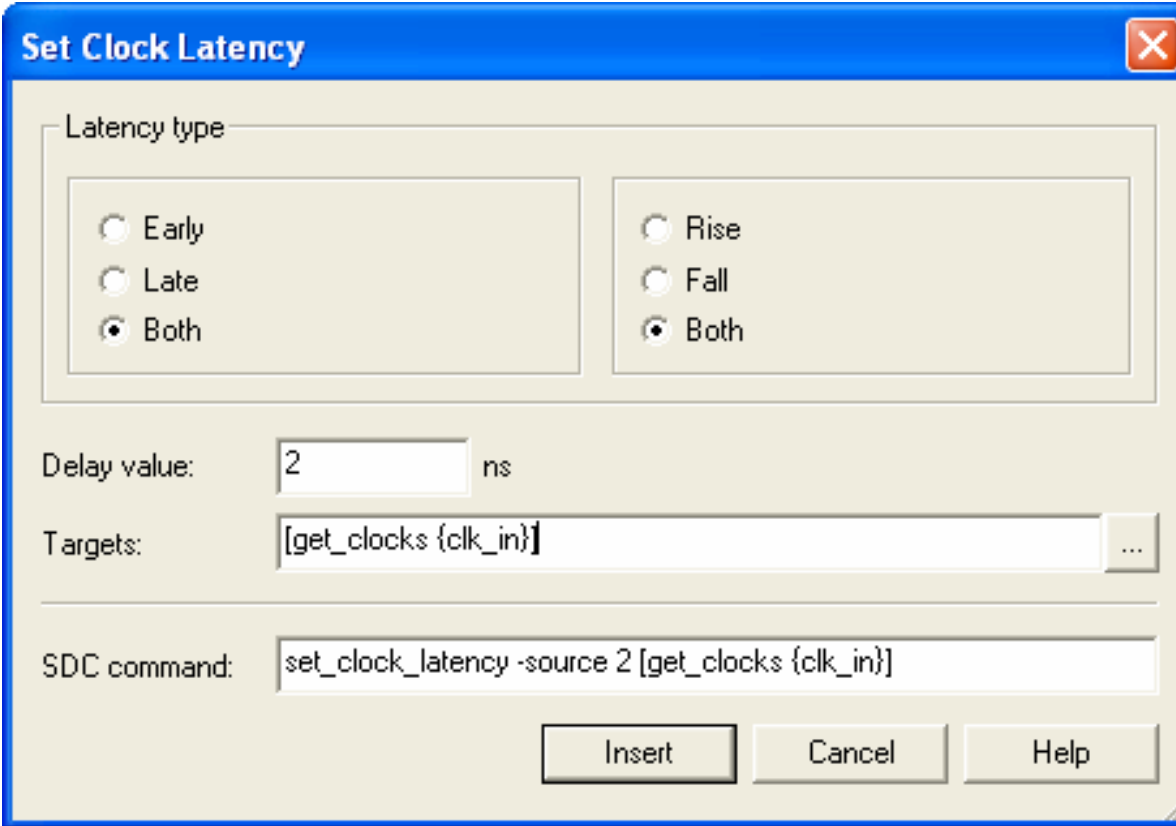
Clock Latency (cont.)

- Command: `set_clock_latency`
- Specify source latency on external path(s) to device
- Options
 - `-source`
 - `[-clock <clock_list>]`
 - `[-early | -late]`
 - `[-fall | -rise]`
 - `<delay>`
 - `<targets>`

set_clock_latency Notes

- `-source`: required argument for constraint (no options)
- `-fall` | `-rise`: latency applied on only falling or rising edge of clock
- `-early` | `-late`: latency on shortest/longest external path
 - Used by timing analyzer as part of definition of data/clock arrival paths for setup/hold analyses

Clock Latency (GUI)



The image shows a 'Set Clock Latency' dialog box with a blue title bar and a close button. It contains two groups of radio buttons for 'Latency type', a 'Delay value' field, a 'Targets' field, and an 'SDC command' field. At the bottom are 'Insert', 'Cancel', and 'Help' buttons.

Set Clock Latency

Latency type

☐ Early ☐ Rise
☐ Late ☐ Fall
☒ Both ☒ Both

Delay value: ns

Targets: ...

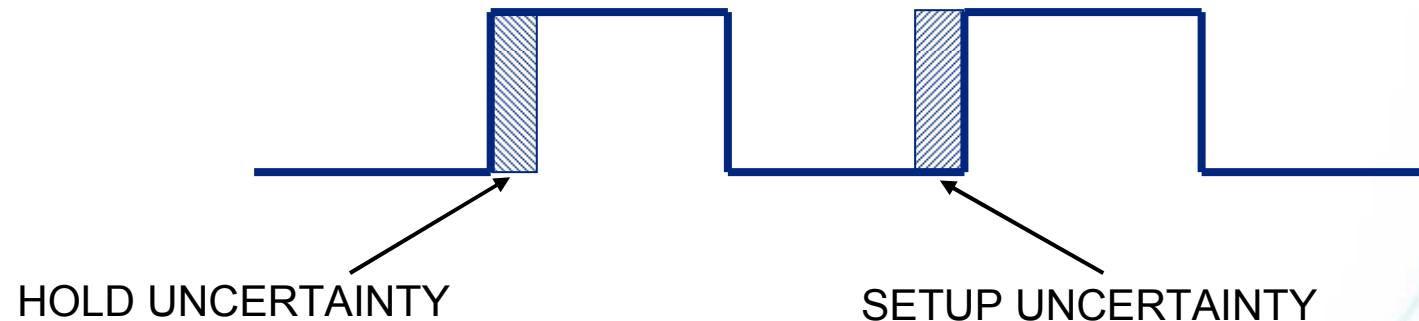
SDC command:

Clock Uncertainty

- Command: `set_clock_uncertainty`
- Use to model jitter, guard band, or skew
 - Allows generation of clocks that are non-ideal
- Options
 - `[-setup | -hold]`
 - `[-fall_from <fall_from_clock>]`
 - `[-fall_to <fall_to_clock>]`
 - `[-from <from_clock>]`
 - `[-rise_from <rise_from_clock>]`
 - `[-rise_to <rise_to_clock>]`
 - `[-to <to_clock>]`
 - `<value>`

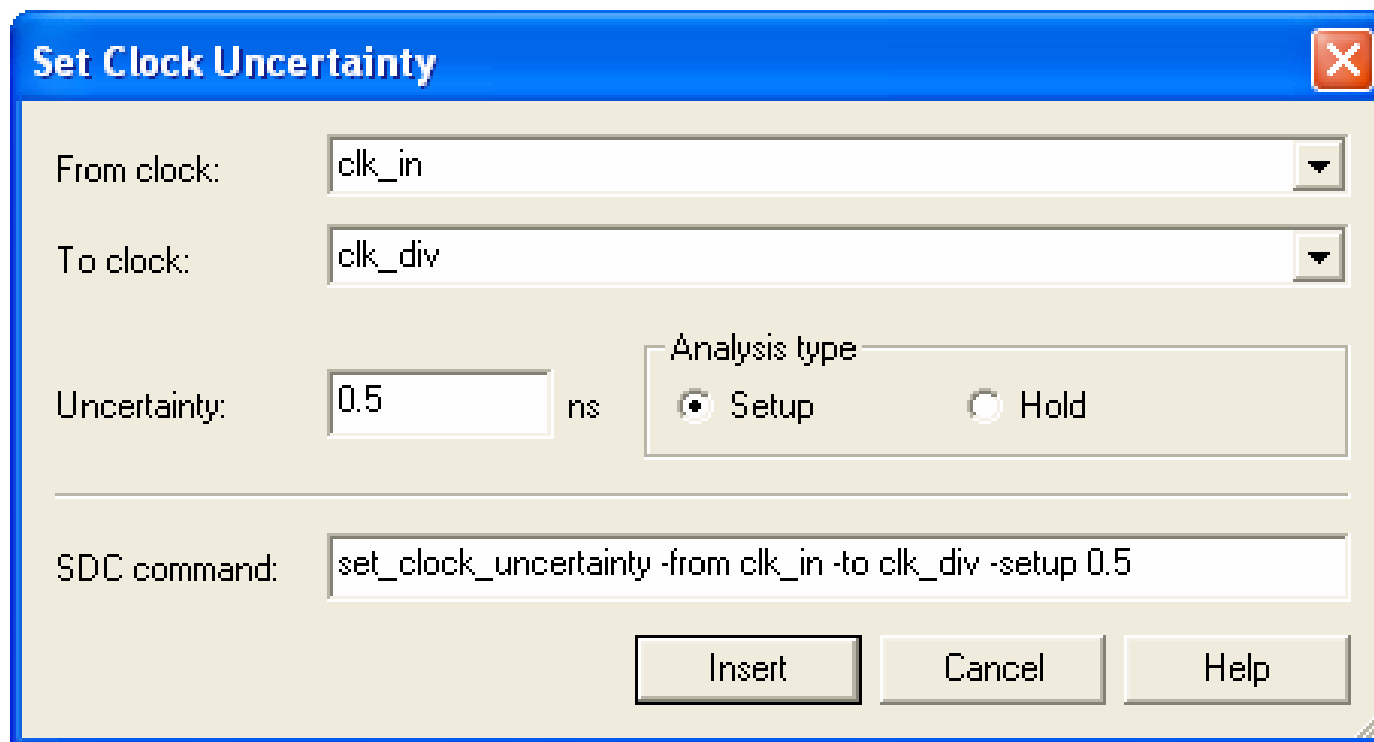
Clock Uncertainty

- Setup uncertainty decreases setup required time
- Hold uncertainty increases hold required time



Ex. To add a 0.5-ns guardband around clock, use 250 ps of setup uncertainty and 250 ps of hold uncertainty.

Clock Uncertainty (GUI)



The image shows a 'Set Clock Uncertainty' dialog box with a blue title bar and a close button. It contains several input fields and a section for analysis type.

From clock:

To clock:

Uncertainty: ns

Analysis type:
☒ Setup ☐ Hold

SDC command:

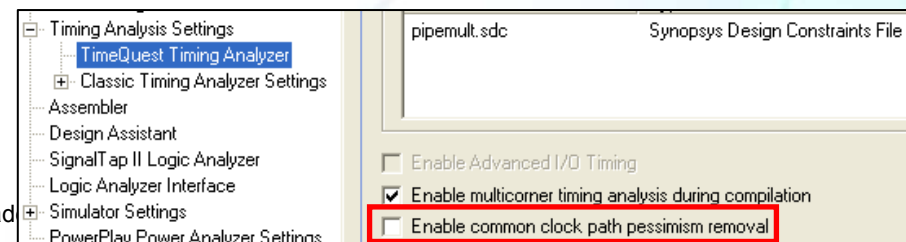
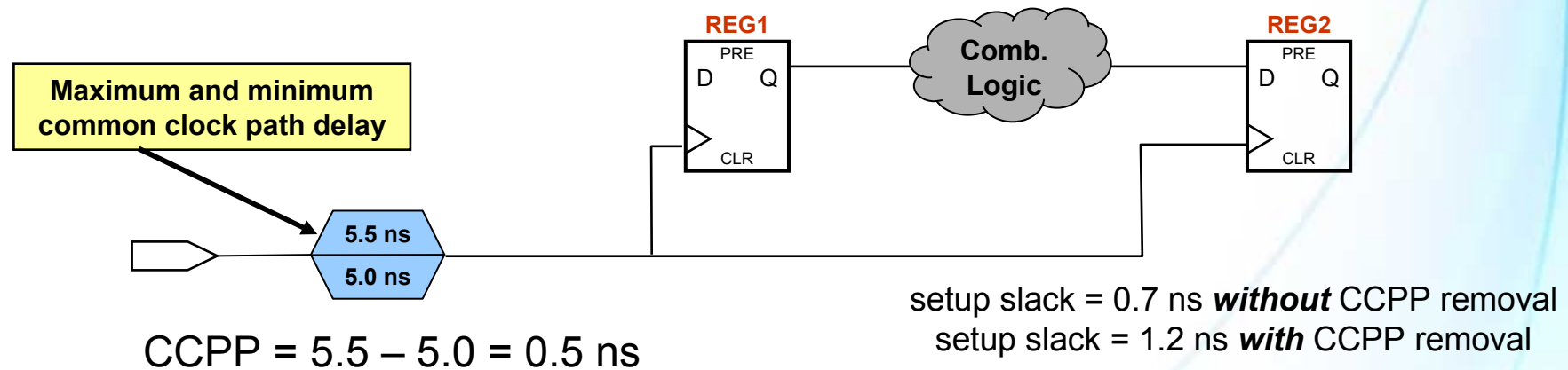
Buttons: Insert, Cancel, Help

Automatically Derive Uncertainty

- Command: `derive_clock_uncertainty`
- Automatically derive clock uncertainties in supported devices
 - Cyclone III, Stratix II, HardCopy® II, Stratix III, and new devices
- Uncertainties created manually with `set_clock_uncertainty` have higher precedence
- Options
 - `[-overwrite]`: overwrites any existing uncertainty constraints
 - `[-add]`: adds derived uncertainties to existing constraints
- SDC extension expanded with `write_sdc -expand`
- *Not in GUI*

Common Clock Path Pessimism Removal

- Remove clock delay pessimism to account for min/max delays on common clock paths (Cyclone III, Stratix III and newer devices)
 - Ex: Max delay for data arrival time; min delay for data required time
- Also used to improve minimum required clock pulse widths
- Enable for fitter and for timing analysis
 - TimeQuest Timing Analyzer settings in Quartus II software
 - `enable_ccpp_removal` in TimeQuest script or console



Checking Clock Constraints

- Nodes used as clocks but not defined with SDC clock constraint considered unconstrained
- Solution
 - Use Unconstrained Paths Report to find unconstrained clocks
 - Quartus II Compilation Report timing summary
 - Run `report_ucp` command
 - Choose **Report Unconstrained Paths** (Tasks Pane or Reports menu)
 - Use Clock Report to verify clocks are constrained correctly

Unconstrained Path Report

Quartus II TimeQuest Timing Analyzer - C:/altera_trn/Quartus_II_Software_Design_Ser

File Edit View Netlist Constraints Reports Script Tools Window Help

Report

- TimeQuest Timing Analyzer Summary
- Unconstrained Paths
 - Unconstrained Paths Summary
 - Clock Status Summary
- Setup Analysis
- Hold Analysis

Property	Setup	Hold
1 Illegal Clocks	0	0
2 Unconstrained Clocks	1	1
3 Unconstrained Input Ports	34	34
4 Unconstrained Input Port Paths	51	51
5 Unconstrained Output Ports	32	32
6 Unconstrained Output Port Paths	32	32

Unconstrained Paths Summary Report indicates how many clock nodes are unconstrained (along with other unconstrained paths)

Clock Status Summary Report lists each clock found and whether it was constrained

Quartus II TimeQuest Timing Analyzer - C:/altera_trn/Quartus_II_Software_D

File Edit View Netlist Constraints Reports Script Tools Window Help

Report

- TimeQuest Timing Analyzer Summary
- Unconstrained Paths
 - Unconstrained Paths Summary
 - Clock Status Summary
- Setup Analysis
- Hold Analysis

Clock	Type	Status
1 clk_test	Base	Constrained
2 clk_in_100mhz	Base	Unconstrained

Report Clocks (report_clocks)


- List details about the properties of constrained clocks

Clock properties

	Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle	Divide by	Multiply by	Phase	Inverted	Master	Source	Targets
1	c100	Generated	10.000	100.0 MHz	0.000	5.000	50.00	1	1		false	clk_in_100mhz	inst1 altpll_component pll inclk[0]	{ inst1 altpll_component pll clk[0] }
2	c100_out	Generated	10.000	100.0 MHz	-3.610	1.390	50.00	1	1	-130.0	false	clk_in_100mhz	inst1 altpll_component pll inclk[0]	{ inst1 altpll_component pll clk[2] }
3	c200	Generated	5.000	200.0 MHz	0.000	2.500	50.00	1	2		false	clk_in_100mhz	inst1 altpll_component pll inclk[0]	{ inst1 altpll_component pll clk[1] }
4	clk_in_100mhz	Base	10.000	100.0 MHz	0.000	5.000								{ clk_in_100mhz }
5	clkout	Generated	10.000	100.0 MHz	6.390	11.390		1	1		false	c100_out	inst1 altpll_component pll clk[2]	{ clkout }

**Clock names
(-name argument
or default name)**

SDC Timing Constraints

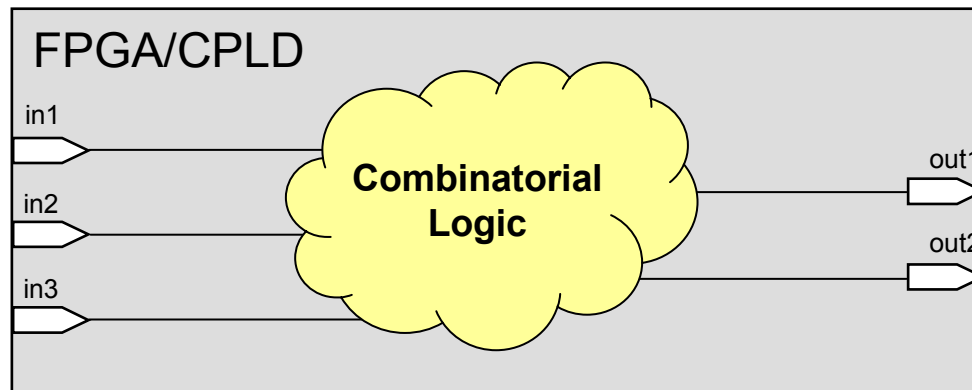
- Clocks
- I/O 
- False paths
- Multicycle paths

I/O Constraints

- Combinatorial I/O interface
- Synchronous I/O interface
- Source synchronous interface

Combinatorial Interface

- All paths from IN to OUT need to be constrained
- Use `set_max_delay` & `set_min_delay` commands
 - Specify an absolute maximum & minimum delay between points

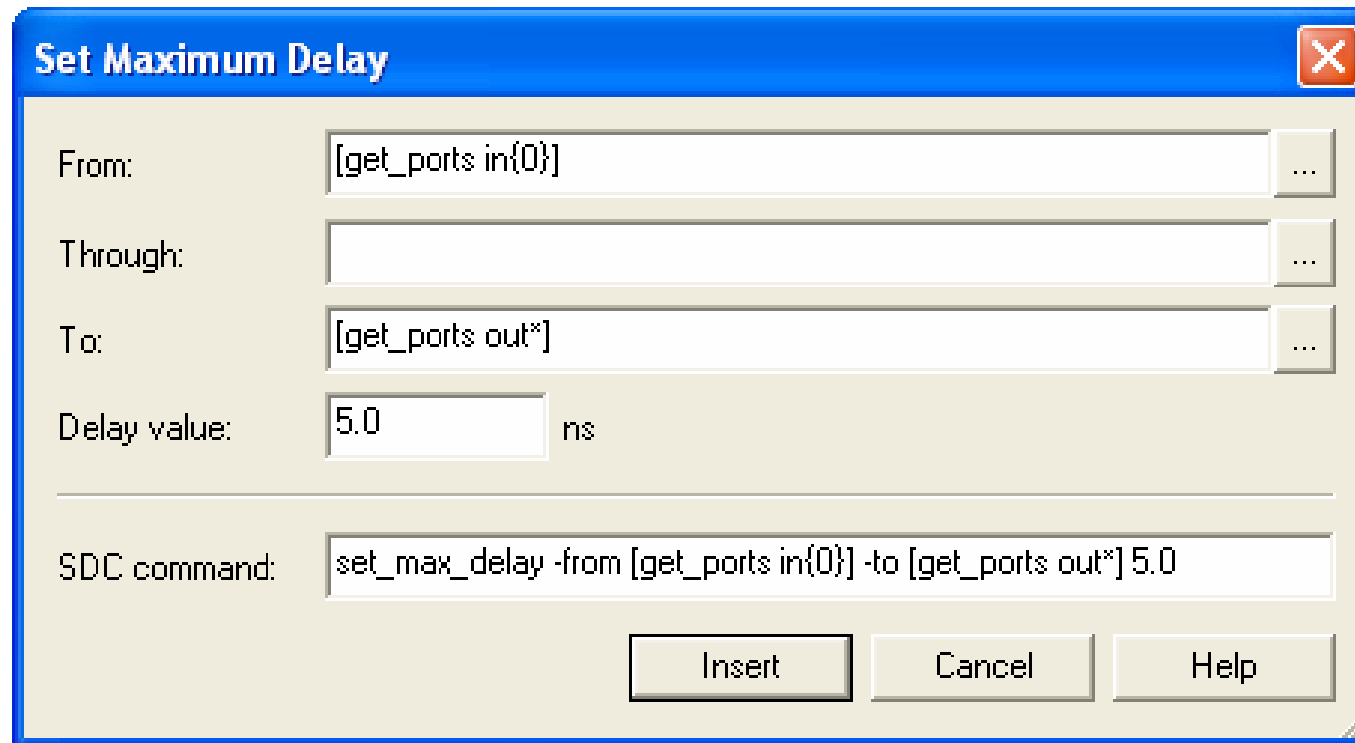


- Options
 - `[-from <names>]`
 - `[-to <names>]`
 - `[-fall_from <clocks>]`
 - `[-rise_from <clocks>]`
 - `[-fall_to <clocks>]`
 - `[-rise_to <clocks>]`
 - `[-through]`
 - `<delay>`

set_max_delay & set_min_delay Notes

- `-from` & `-to`: Use to indicate source & destination nodes for constraints
- `-through`: Use to indicate the constraint should only be applied to path(s) going through a particular node name

set_max_delay & set_min_delay (GUI)



The image shows a GUI dialog box titled "Set Maximum Delay". It has a blue title bar with a close button (X) in the top right corner. The dialog contains several input fields and buttons. The "From:" field is set to "[get_ports in{0}]", the "Through:" field is empty, and the "To:" field is set to "[get_ports out*]". The "Delay value:" is set to "5.0" with a unit of "ns". At the bottom, there is a text box for the "SDC command:" which contains the command "set_max_delay -from [get_ports in{0}] -to [get_ports out*] 5.0". Below the text box are three buttons: "Insert", "Cancel", and "Help".

Set Maximum Delay

From: [get_ports in{0}] ...

Through: ...

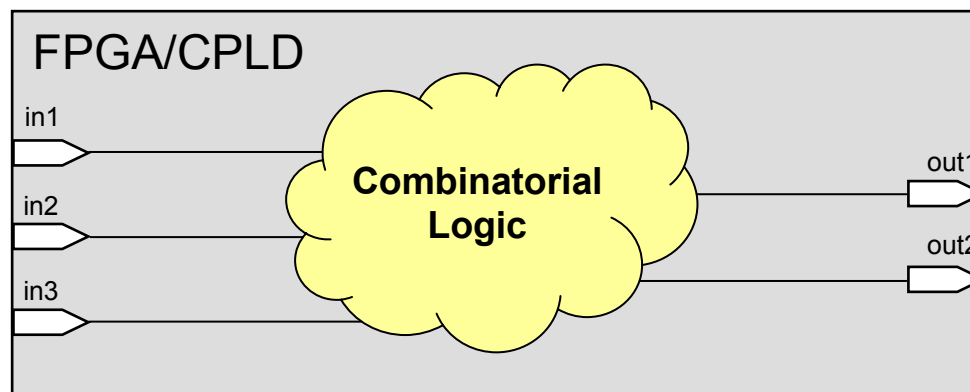
To: [get_ports out*] ...

Delay value: 5.0 ns

SDC command: set_max_delay -from [get_ports in{0}] -to [get_ports out*] 5.0

Insert Cancel Help

Combinatorial Interface Example

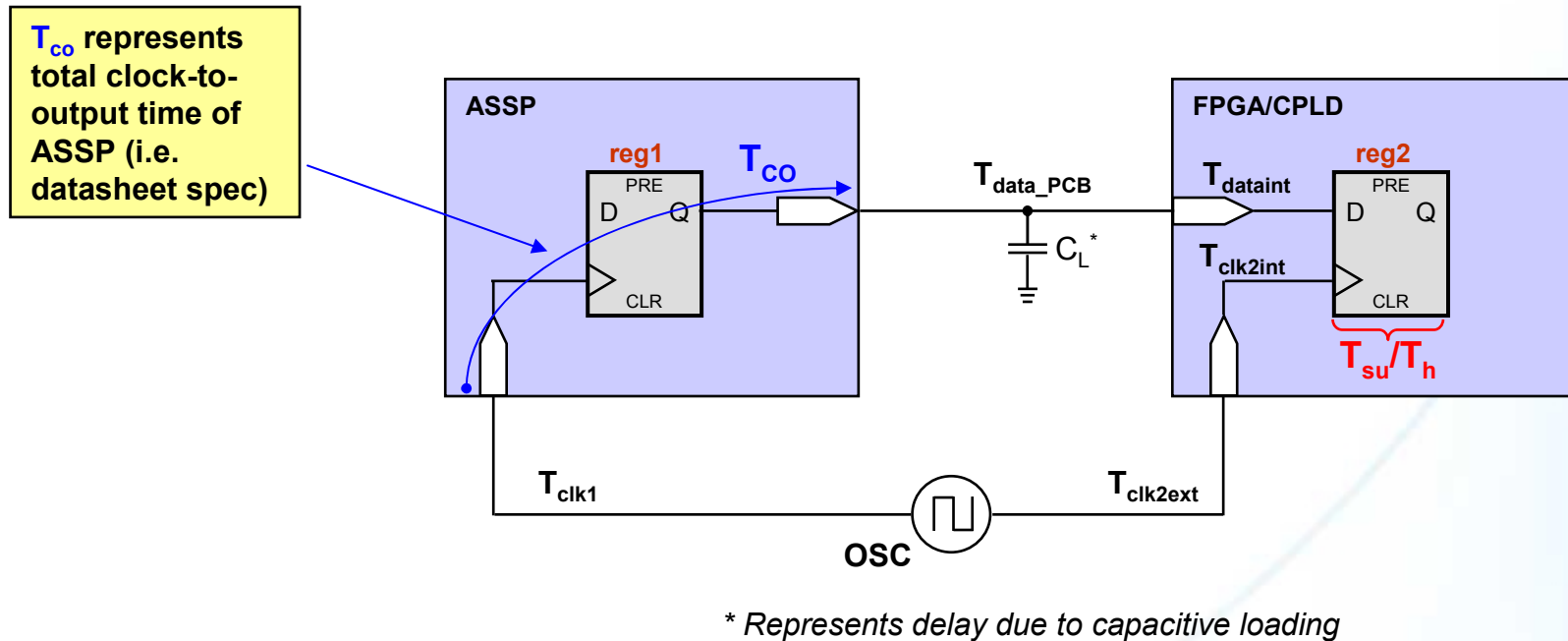


```
set_max_delay -from [get_ports in1] -to [get_ports out*] 5.0
set_max_delay -from [get_ports in2] -to [get_ports out*] 7.5
set_max_delay -from [get_ports in3] -to [get_ports out*] 9.0
```

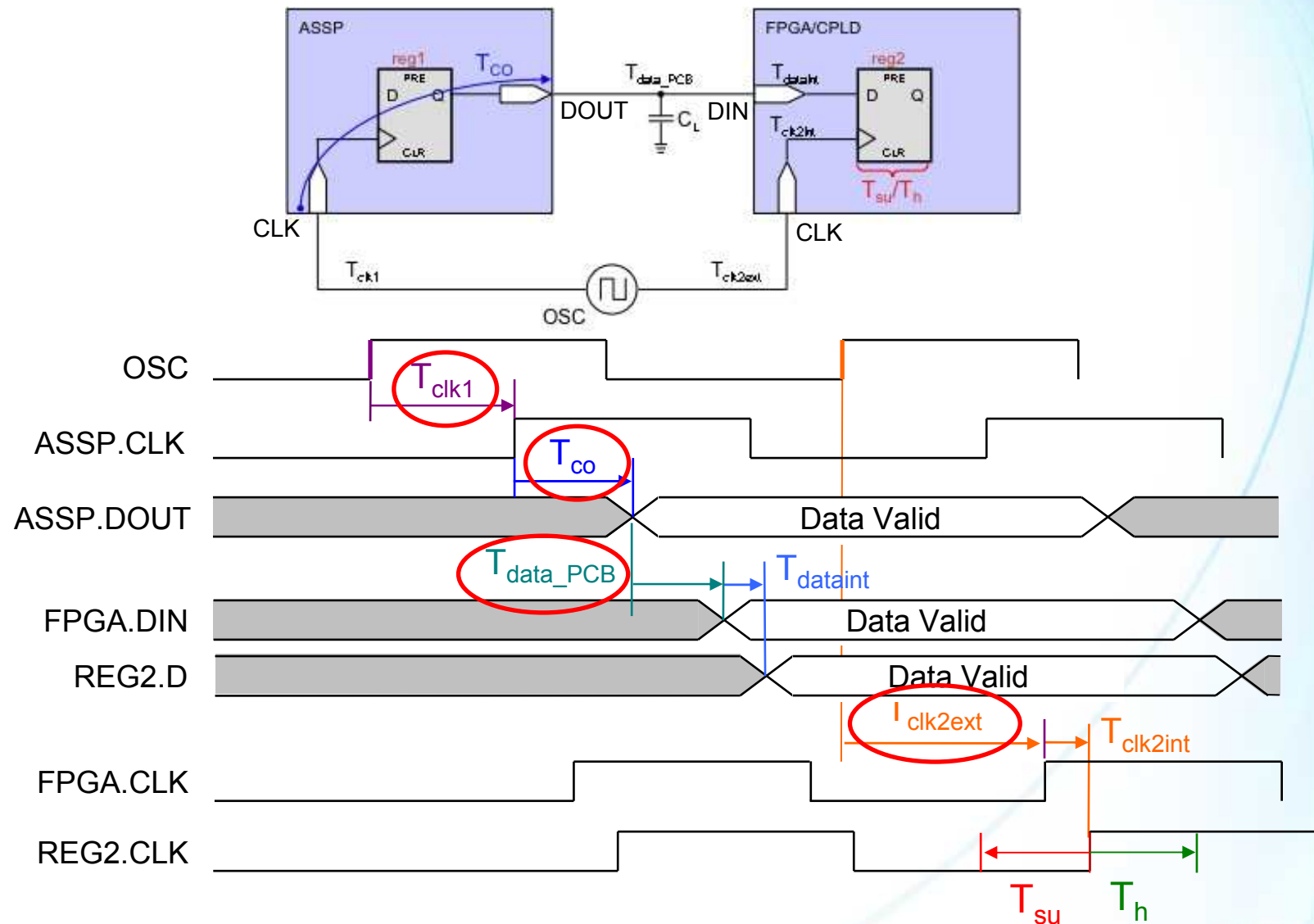
```
set_min_delay -from [get_ports in1] -to [get_ports out*] 1.0
set_min_delay -from [get_ports in2] -to [get_ports out*] 2.0
set_min_delay -from [get_ports in3] -to [get_ports out*] 3.0
```

Synchronous Inputs

- Need to specify timing relationship from ASSP to FPGA/CPLD to guarantee setup/hold in FPGA/CPLD



Synchronous Inputs



Constraining Synchronous Inputs

- Use `set_input_delay (-max option)` command to constrain **input setup time** (maximum time to arrive and still meet T_{su})

- Calculated input delay value represents all delays external to device

input delay max	= Board Delay (max) - Board clock skew (min) + $T_{co(max)}$
	= $(T_{data_PCB(max)} + T_{CL}) - (T_{clk2ext(min)} - T_{clk1(max)}) + T_{co(max)}$
data arrival time	= launch edge + input delay max + $T_{dataint}$
data required time	= latch edge + $T_{clk2int}$ - T_{su}
slack	= required time - data arrival time

- Use `set_input_delay (-min option)` command to constrain **input hold time** (minimum time to stay active and still meet T_h)

- Calculated input delay value represents all delays external to device

input delay min	= Board Delay (min) - Board clock skew (max) + $T_{co(min)}$
	= $(T_{data_PCB(min)} + T_{CL}) - (T_{clk2ext(max)} - T_{clk1(min)}) + T_{co(min)}$
data arrival time	= launch edge + input delay min + $T_{dataint}$
required time	= latch edge + $T_{clk2int}$ + T_h
slack	= data arrival time - data required time

set_input_delay Command

- Constrains input pins by specifying *external* device timing parameters
- Options

```
-clock <clock_name>  
[-clock_fall]  
[-rise | -fall]  
[-max | -min]  
[-add_delay]  
[-reference_pin <target>]  
[-source_latency_included]  
<delay value>  
<targets>
```


set_input_delay Notes

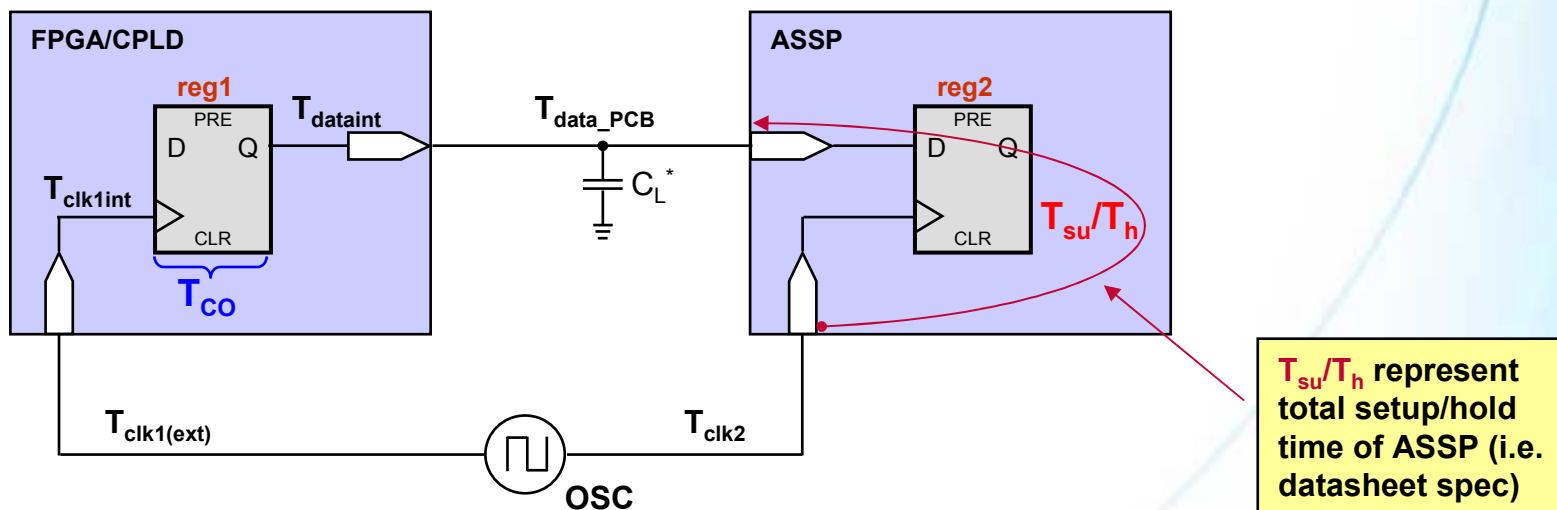
- `-clock`: Specifies the clock driving the source (external) register
 - Used to determine launch edge vs. latch edge relationship
- `-clock_fall`: Use to specify input signal was launched by a falling edge clock transition
- `-rise` | `-fall`: Use to indicate whether input delay value is for a rising or falling edge transaction
- `-add_delay`: Use to specify multiple constraints on single input
 - Only one **set** of max/min & rise/fall constraints allowed on an input pin
 - Ex. Constraining one input port driving two registers in different clock domains would require the `-add_delay` option

set_input_delay Notes

- `-reference_pin`: Use to specify that delays are with respect to some other port or pin in the design
 - Example: Feedback clock: Input delay is relative to an output port being fed by a clock
- `-source_latency_included`: input delay value specified includes clock source latency normally added automatically
 - Tells TimeQuest to ignore any clock latency constraints applied to source clock
- To fully constrain, must specify both `-max` & `-min`
 - Each will default to the value of the other setting if only one assigned (same with rise/fall)
 - Warning message if one or the other not specified

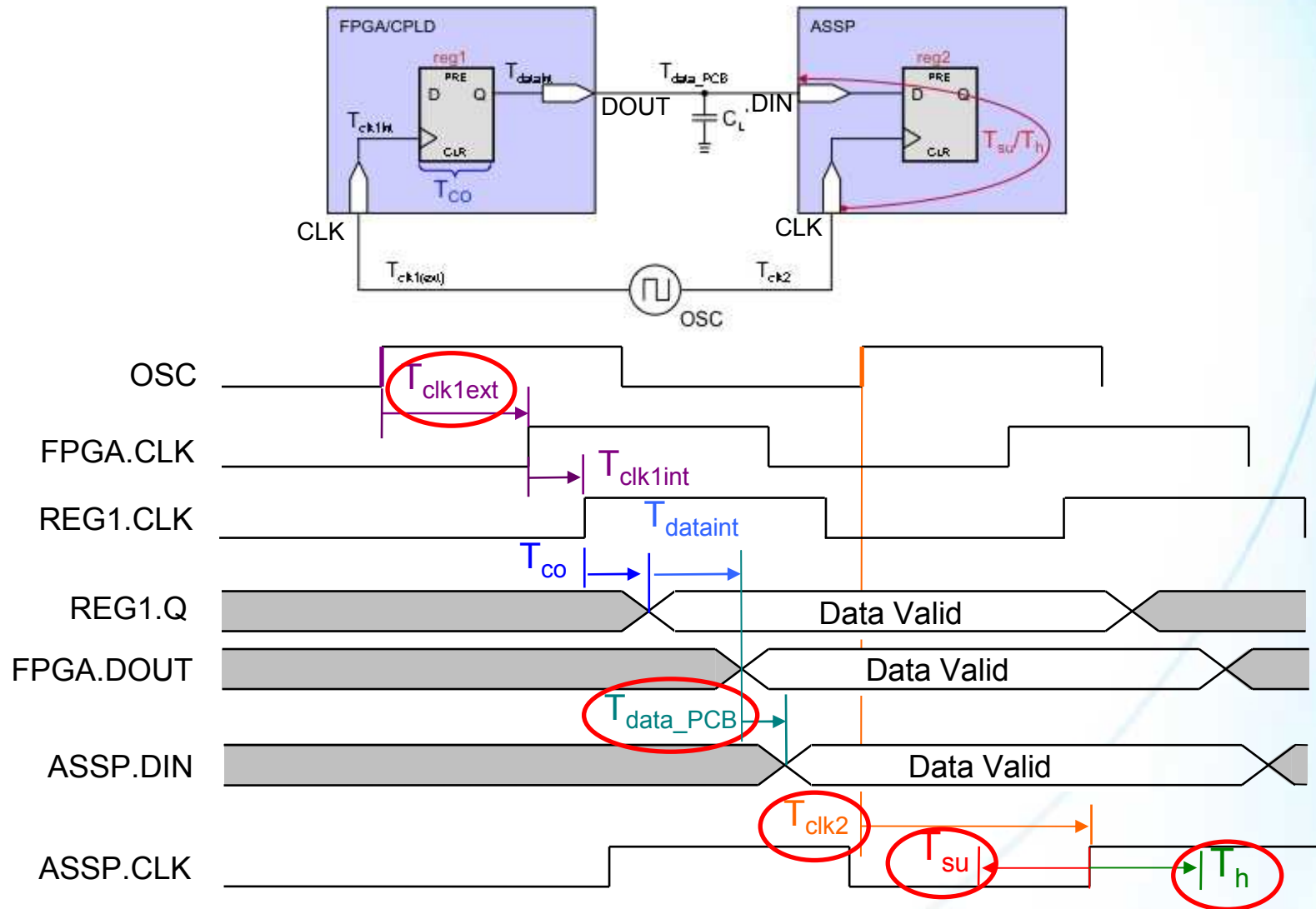
Synchronous Outputs

- Need to specify timing relationship from FPGA/CPLD to ASSP to guarantee clock-to-output times in FPGA/CPLD



* Represents delay due to capacitive loading

Synchronous Outputs



Constraining Synchronous Outputs

- Use `set_output_delay (-max option)` command to constrain **maximum clock-to-output** (maximum time to arrive and still meet ASSP's T_{su})

- Calculated output delay value represents all delays external to device

output delay max	= Board Delay (max) - Board clock skew (min) + T_{su}
	$= (T_{data_PCB(max)} + T_{CL}) - (T_{clk2(min)} - T_{clk1ext(max)}) + T_{su}$
data arrival time	= launch edge + $T_{clk1int}$ + $T_{co(max)}$ + $T_{dataint}$
data required time	= latch edge - output delay max
slack	= data required time - data arrival time

- Use `set_output_delay (-min option)` command to constrain **minimum clock-to-output** (minimum time to stay active and still meet ASSP's T_h)

- Calculated output delay value represents all delays external to device

output delay min	= Board Delay (min) - Board clock skew (max) - T_h
	$= (T_{data_PCB(min)} + T_{CL}) - (T_{clk2(max)} - T_{clk1ext(min)}) - T_h$
data arrival time	= launch edge + $T_{clk1int}$ + $T_{co(min)}$ + $T_{dataint}$
data required time	= latch edge - output delay min
slack	= data arrival time - data required time

set_output_delay Command

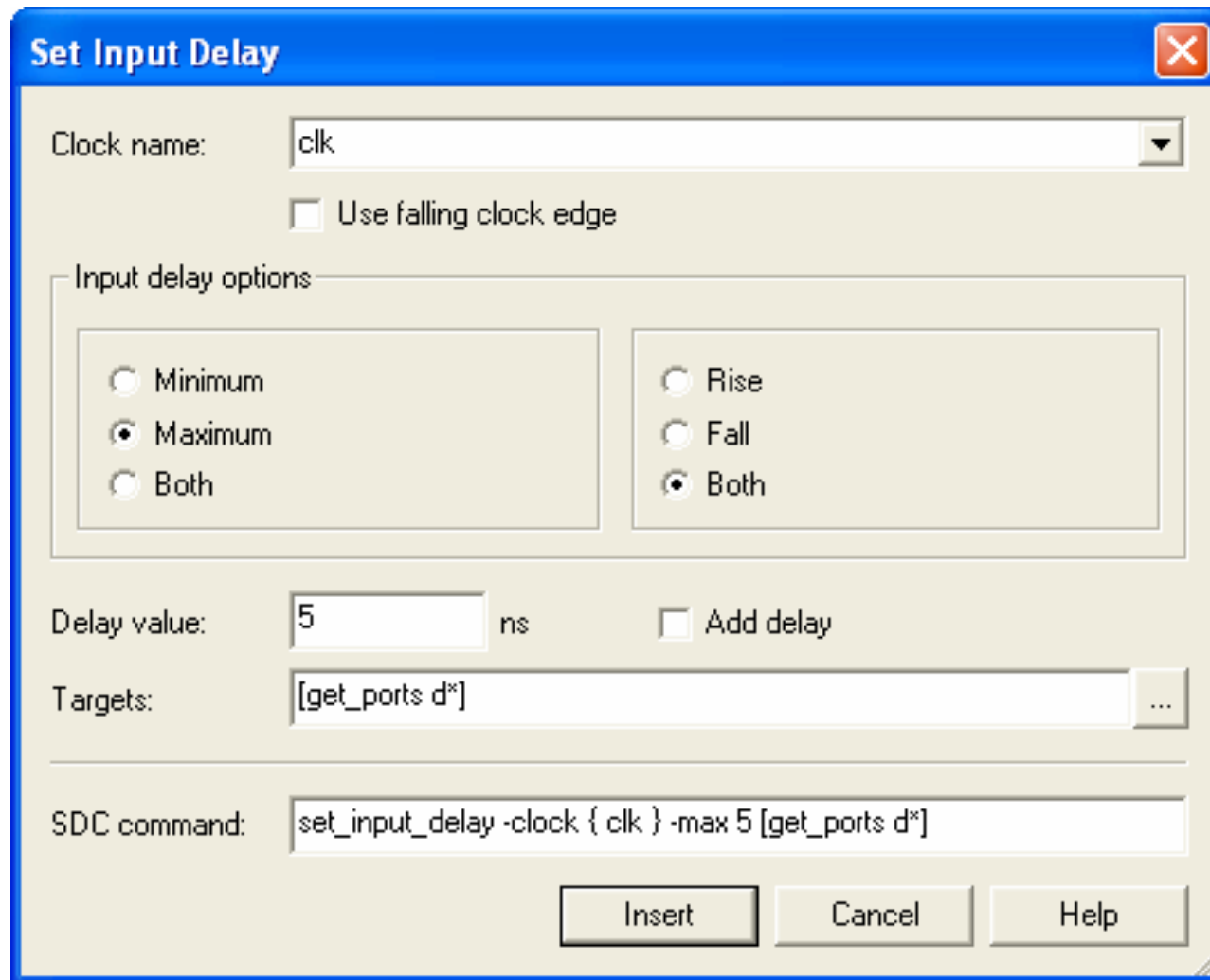
- Constrains output pins by specifying external device timing parameters
- Options

```
-clock <clock_name>  
[-clock_fall]  
[-rise | -fall]  
[-max | -min]  
[-add_delay]  
[-reference_pin <target>]  
<delay value>  
<targets>
```

set_output_delay Notes

- Same notes as `set_input_delay` command

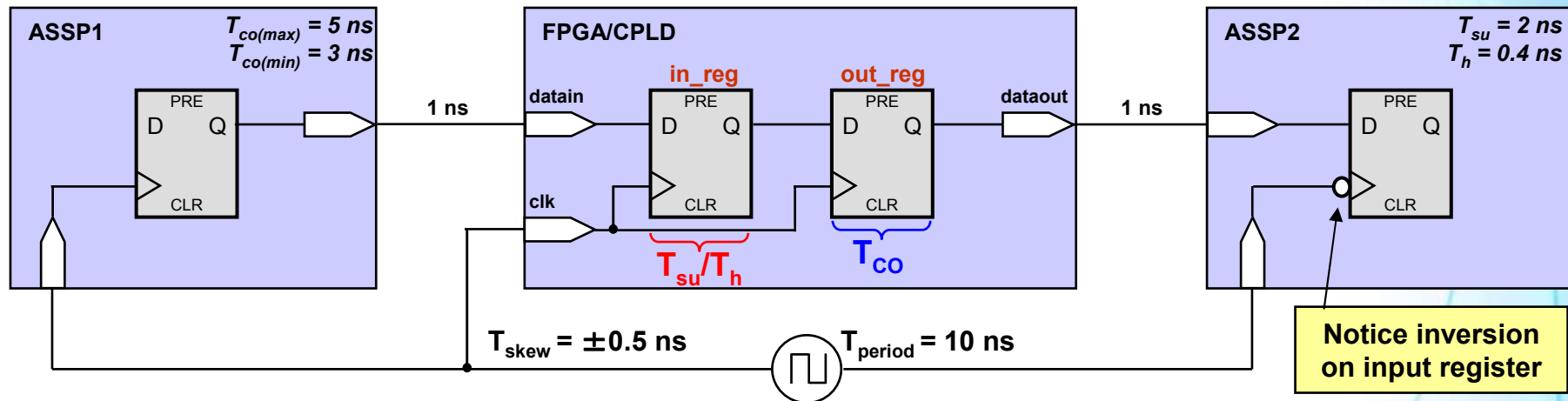
Input/Output Delays (GUI)



The image shows a 'Set Input Delay' dialog box with a blue title bar and a close button. It contains the following fields and controls:

- Clock name:** A dropdown menu with 'clk' selected.
- Use falling clock edge:** An unchecked checkbox.
- Input delay options:** A group box containing two columns of radio buttons:
 - Left column: ☐ Minimum, ☒ Maximum, ☐ Both.
 - Right column: ☐ Rise, ☐ Fall, ☒ Both.
- Delay value:** A text field with '5' and a unit dropdown set to 'ns'.
- Add delay:** An unchecked checkbox.
- Targets:** A text field with '[get_ports d*]' and a browse button ('...').
- SDC command:** A text field containing the command: `set_input_delay -clock { clk } -max 5 [get_ports d*]`.
- Buttons:** 'Insert', 'Cancel', and 'Help' at the bottom right.

Synchronous I/O Example



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

```
set_input_delay -clock clk -max [expr 1 - (-0.5) + 5] [get_ports datain]
```

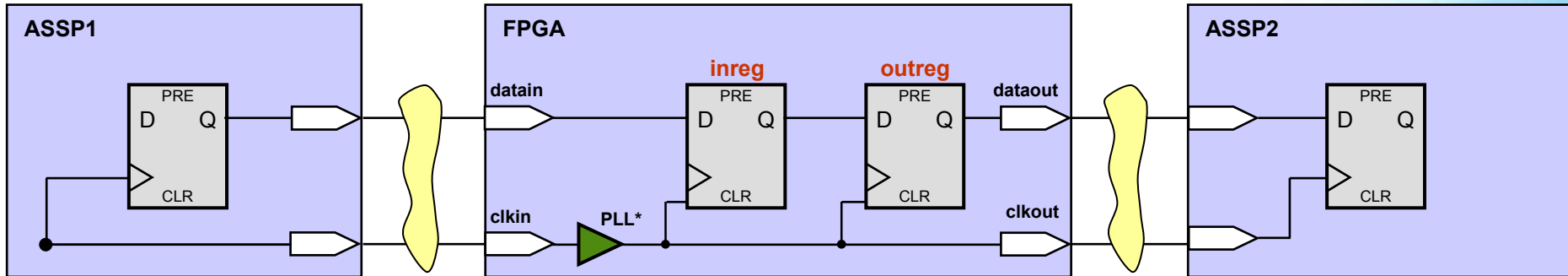
```
set_input_delay -clock clk -min [expr 1 - 0.5 + 3] [get_ports datain]
```

```
set_output_delay -clock clk -max [expr 1 - (-0.5) + 2] \
    -clock_fall [get_ports dataout]
```

```
set_output_delay -clock clk -min [expr 1 - 0.5 - 0.4] \
    -clock_fall [get_ports dataout]
```

Note: *expr* in these constraints is used to simply calculate the value of the equation broken down into the 3 parts defined by the input/output delay equations

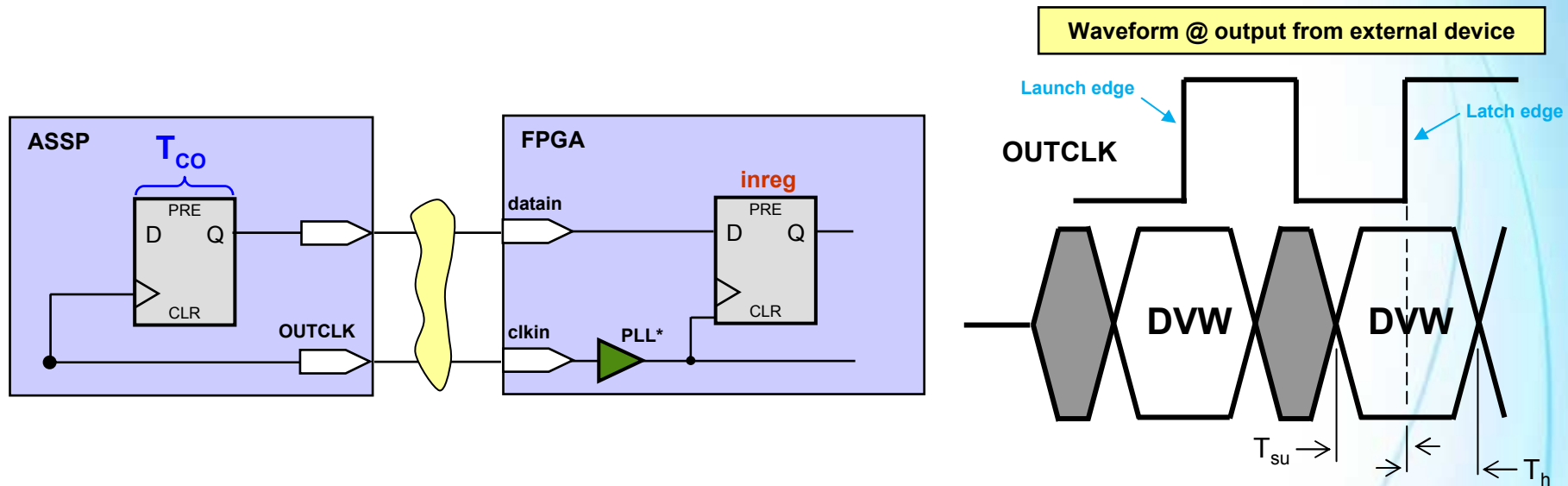
Source-Synchronous Interfaces



- Both data & clock transmitted by host device with designated phase relationship (e.g. edge or center-aligned)
 - No clock tree skew included in calculation
 - Target device uses transmitted clock to sample incoming data
- Data & clock routed identically to maintain phase relationship at destination device
 - Board delay not included in external delay calculations
 - Clock trace delay (data required time) & Data trace delay (data arrival time) are equal and offset
 - Enables higher interface speeds (compared to using system clock)

* The PLL in this example, represented by a single symbol, is actually generating multiple outputs clocks

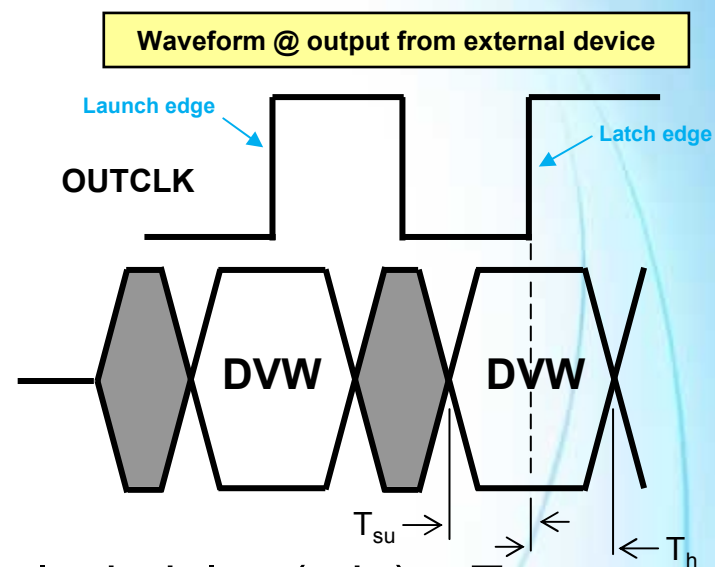
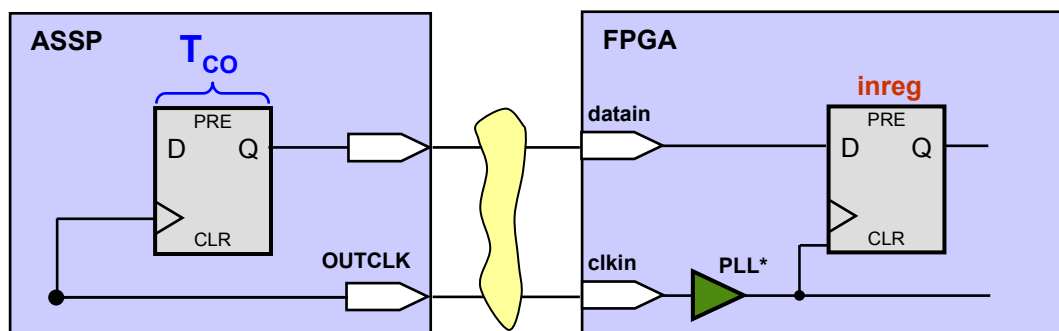
SDR Source-Synchronous Input (Center-Aligned)



- Total setup/hold relationship of FPGA to clock (clkkin) already defined by output waveform of external device
 - T_{su} is start of DVW
 - T_h is end of DVW
- Must derive `set_input_delay` values from T_{su} & T_h

* The PLL in this example is used to maintain the input clock to data relationship

SDR Source-Synchronous Input (Center-Aligned)



$$\begin{aligned} \text{input delay max} &= \cancel{\text{board delay (max)}} - \cancel{\text{clock delay (min)}} + T_{co(max)} \\ &= T_{co(max)} \end{aligned}$$

$$\text{setup slack} = \text{data required time} - \text{data arrival time}$$

If setup slack = 0 (start of DVW):

$$\text{data arrival time} = \text{data required time}$$

$$\text{latch edge} - T_{su} = \text{launch edge} + \text{input delay max}$$

so

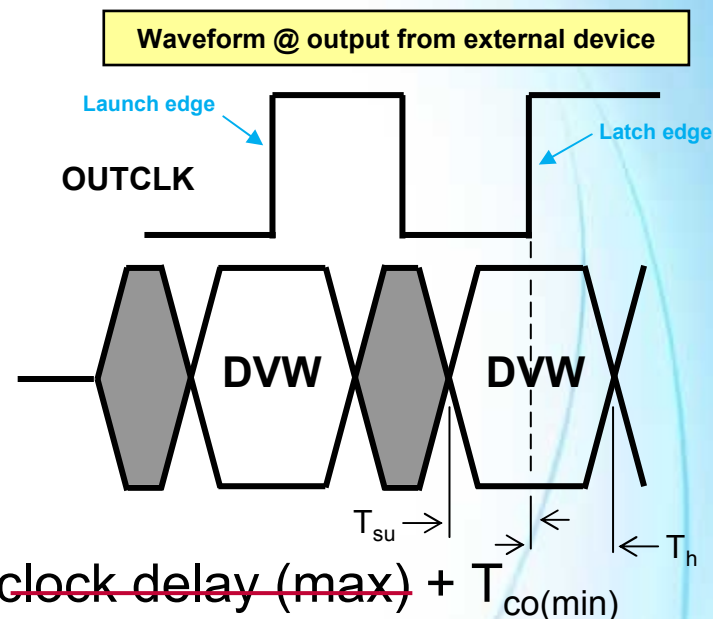
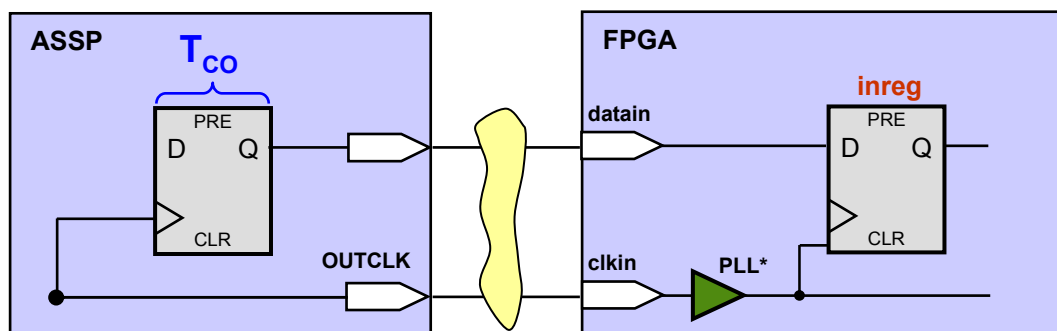
$$\text{input delay max} = (\text{latch edge} - \text{launch edge})^* - T_{su} \quad \text{*Typically 1 clock period for SDR}$$

Note: In reality for high-speed designs, there would be some max/min board & clock delay that would need to be figured into the analysis.

© 2009 Altera Corporation

Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation

SDR Source-Synchronous Input (Center-Aligned)



input delay min

$$= \text{board delay (min)} - \text{clock delay (max)} + T_{\text{co(min)}}$$

$$= T_{\text{co(min)}}$$

hold slack

= data arrival time - data required time

If hold slack = 0 (end of DVW):

data required time = data arrival time

$$\text{latch edge} + T_h = \text{launch edge} + \text{input delay min}$$

For hold analysis, latch and launch edges cancel out, so

input delay min = T_h

Note: In reality for high-speed designs, there would be some max/min board & clock delay that would need to be figured into the analysis.

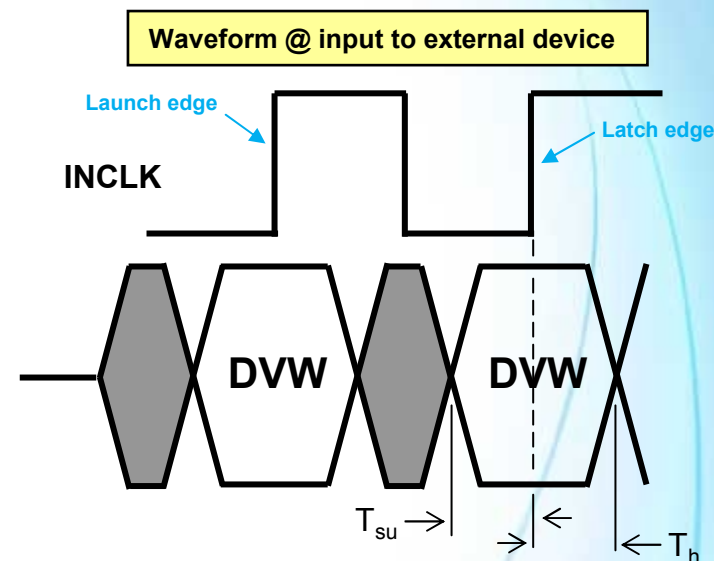
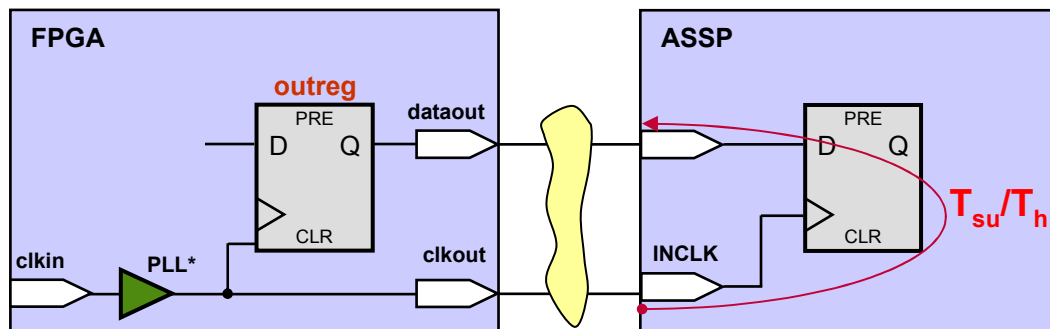
© 2009 Altera Corporation

Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation

Using SDC with Source-Sync Input

- Create clock on clock input port
- Use `set_input_delay` command with reference to clock input
 - Same as with synchronous input
 - Do not include board delay parameters in value

SDR Source-Synchronous Output (Center-Aligned)



* The PLL in this example is used to shift output clock to establish an output clock to data relationship

$$\begin{aligned} \text{output delay max} &= \cancel{\text{board delay (max)}} - \cancel{\text{clock delay (min)}} + T_{su} \\ &= T_{su} \end{aligned}$$

$$\begin{aligned} \text{output delay min} &= \cancel{\text{board delay (min)}} - \cancel{\text{clock delay (max)}} - T_h \\ &= -T_h \end{aligned}$$

Notice output delay minimum is negative

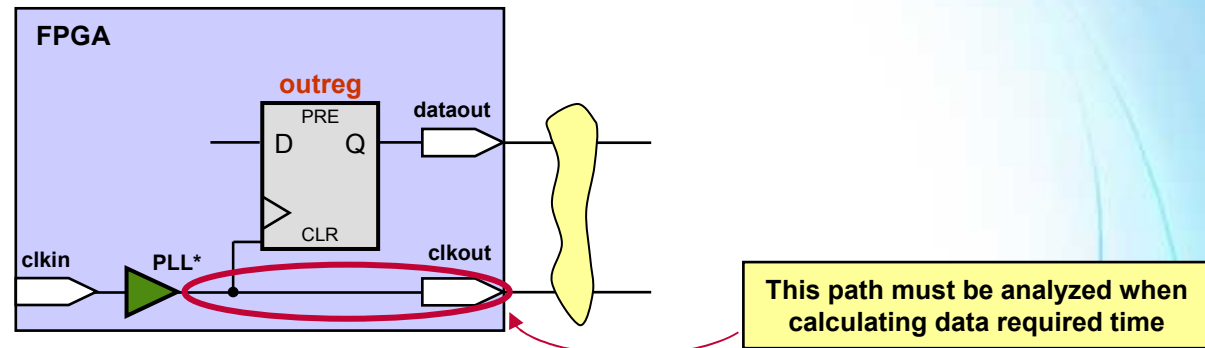
Notes:

- 1) In reality for high-speed designs, there would be some max/min board & clock delay that would need to be figured into the analysis.
- 2) The PLL in this example is used to shift output clock to establish an output clock to data relationship

© 2009 Altera Corporation

Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation

Using SDC with Source-Synch Output



- Must tell TimeQuest to analyze path from clock source to output clock port during analysis
- Use `set_output_delay` command on **dataout** with reference to generated clock on output port
 - Create generated clock on output clock port (source is PLL output pin)
 - Use `-clock` argument in output delay assignment to associate output clock to output data bus
- Path from PLL output pin to output port still considered unconstrained (clock path viewed as a data path by timing analyzer)
 - Constrain path from PLL pin to output port with false path (described later), `set_min/max_delay`, or `set_output_delay`

Constraining Source-Sync Output Example

```
create_clock 5 -name clkin \  
    [get_ports clkin]  
create_generated_clock -name pllclk divide_by 1 \  
    -source [get_ports clkin]  
    [get_pins inst|altpll_component|pll|clk[0]]
```

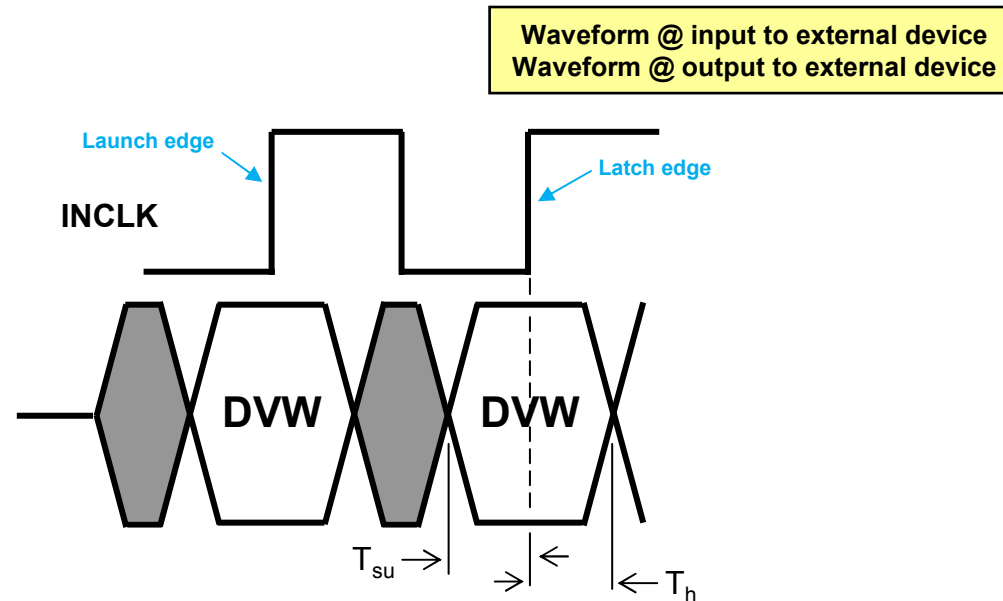
```
# Place clock on external clock output  
create_generated_clock -name clkout \  
    -source [get_pins inst|altpll_component|pll|clk[0]] \  
    -divide_by 1 \  
    [get_ports clkout]
```

```
# Constrain dataout with an external tsu of 0.5 ns
```

```
# and th of 0.5 ns using clkout as clock
```

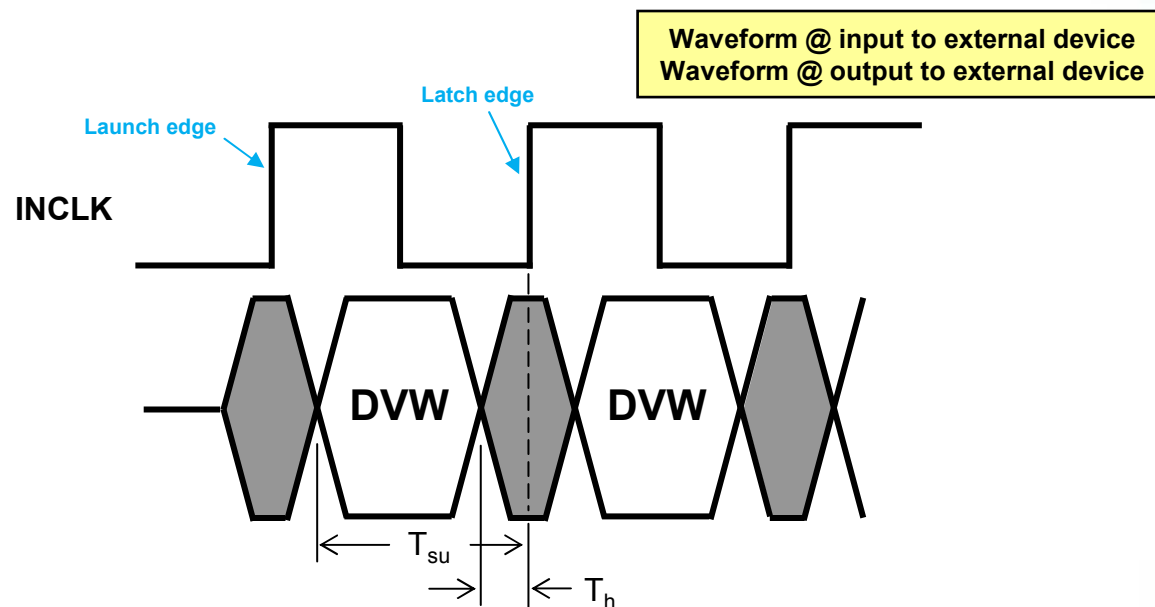
```
set_output_delay -clock [get_clocks clkout] \  
    -max 0.500 \  
    [get_ports dataout]  
set_output_delay -clock [get_clocks clkout] \  
    -min -0.500 \  
    [get_ports dataout]
```

Source Synchronous Summary (Center-Aligned)



	Maximum	Minimum
Input delay setting (ns)	(latch edge – launch edge) - T_{su}	T_h
Output delay setting (ns)	T_{su}	$-T_h$

Source Synchronous (Edge-Aligned)



	Maximum	Minimum
Input delay setting (ns)	(latch edge – launch edge) - T_{su}	$-T_h$
Output delay setting (ns)	T_{su}	T_h

Checking I/O Constraints

- Helpful TimeQuest reports to run to verify constraints
- Report SDC
- Report Unconstrained Paths (again)
- Report Ignored Constraints

Report SDC (report_sdc)

- List SDC constraints applied to netlist

The screenshot displays the SDC report window with four main sections:

- Create Clock:** Shows a single entry for the base clock.
- Create Generated Clock:** Shows multiple entries for generated clocks.
- Set Input Delay:** Lists input delays for various ports.
- Set Output Delay:** Lists output delays for various ports.

Annotations highlight specific sections:

- Base clock:** Points to the 'Create Clock' table.
- Generated clocks:** Points to the 'Create Generated Clock' table.
- Input delays:** Points to the 'Set Input Delay' table.
- Output delays:** Points to the 'Set Output Delay' table.

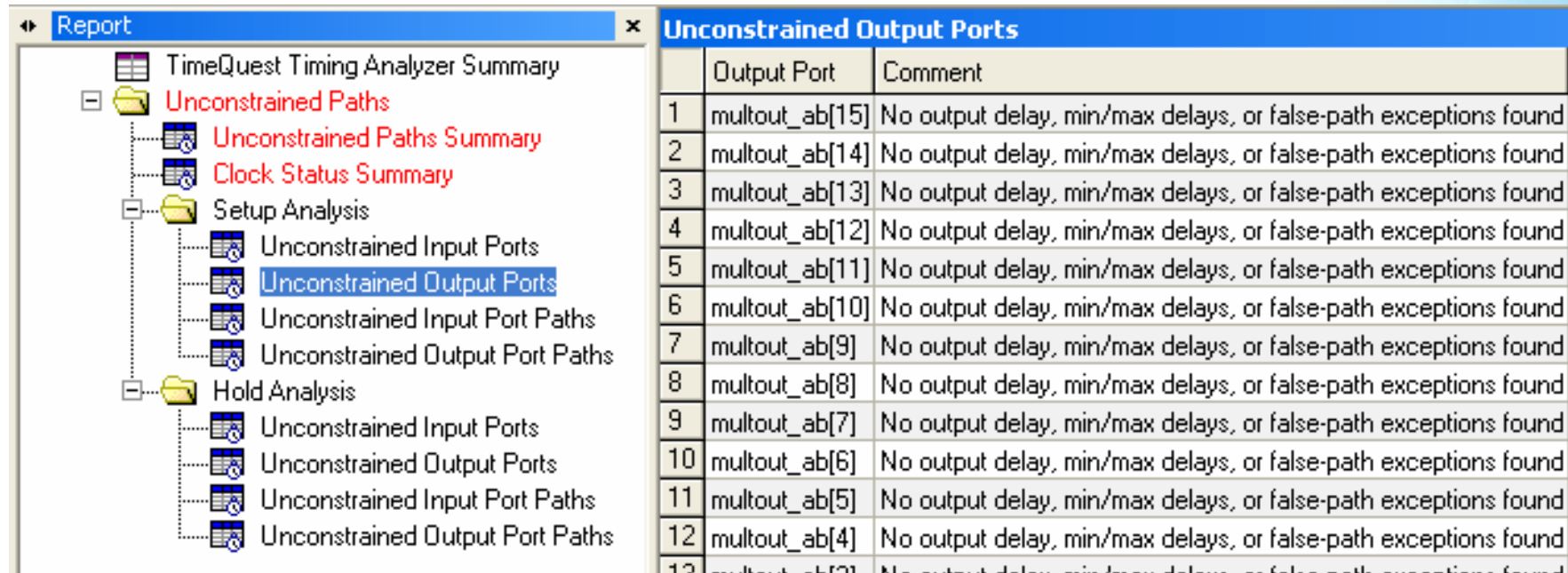
SDC Command	Name	Period	Waveform	Targets
1 create_clock	clk_in_100mhz	10.000	{ 0.000 5.000 }	[get_ports {clk_in_100mhz}]

SDC Command	Name	Source	Duty Cycle	Multiply
1 create_generated_clock	c100	[get_pins {inst1 altpll_component pllinc1k[0]}]	50.000	1
2 create_generated_clock	c200	[get_pins {inst1 altpll_component pllinc1k[0]}]	50.000	2
3 create_generated_clock	c100_out	[get_pins {inst1 altpll_component pllinc1k[0]}]	50.000	1
4 create_generated_clock	clkout	[get_pins {inst1 altpll_component pllinc1k[0]}]	50.000	1

SDC Command	Flags	Clock Name	Delay	Ports
1 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_a[0]]
2 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_a[0]]
3 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_b[0]]
4 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_b[0]]
5 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_x[0]]
6 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_x[0]]
7 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_y[0]]
8 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_y[0]]
9 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_a[1]]
10 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_a[1]]
11 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_b[1]]
12 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_b[1]]
13 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_x[1]]
14 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_x[1]]
15 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_y[1]]
16 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_y[1]]
17 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_a[2]]
18 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_a[2]]
19 set_input_delay	-max	[get_clocks clk_in_100mhz]	4.500	[get_ports din_b[2]]
20 set_input_delay	-min	[get_clocks clk_in_100mhz]	1.000	[get_ports din_b[2]]

SDC Command	Flags	Clock Name	Delay	Ports	Source Latency
1 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[15]]	
2 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[15]]	
3 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[14]]	
4 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[14]]	
5 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[13]]	
6 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[13]]	
7 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[12]]	
8 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[12]]	
9 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[11]]	
10 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[11]]	
11 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[10]]	
12 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[10]]	
13 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[9]]	
14 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[9]]	
15 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[8]]	
16 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[8]]	
17 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[7]]	
18 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[7]]	
19 set_output_delay	-max	[get_clocks clkout]	0.500	[get_ports multout_ab[6]]	
20 set_output_delay	-min	[get_clocks clkout]	-0.500	[get_ports multout_ab[6]]	

Report Unconstrained Paths (report_ucp)



The screenshot displays the 'Report' window in Quartus II. The left pane shows a tree view of the report structure. The 'Unconstrained Paths' folder is expanded, showing sub-items: 'Unconstrained Paths Summary', 'Clock Status Summary', 'Setup Analysis', and 'Hold Analysis'. The 'Setup Analysis' folder is further expanded, showing 'Unconstrained Input Ports', 'Unconstrained Output Ports' (which is selected), 'Unconstrained Input Port Paths', and 'Unconstrained Output Port Paths'. The right pane shows a table titled 'Unconstrained Output Ports' with 12 rows, each listing an output port and a comment.

	Output Port	Comment
1	multout_ab[15]	No output delay, min/max delays, or false-path exceptions found
2	multout_ab[14]	No output delay, min/max delays, or false-path exceptions found
3	multout_ab[13]	No output delay, min/max delays, or false-path exceptions found
4	multout_ab[12]	No output delay, min/max delays, or false-path exceptions found
5	multout_ab[11]	No output delay, min/max delays, or false-path exceptions found
6	multout_ab[10]	No output delay, min/max delays, or false-path exceptions found
7	multout_ab[9]	No output delay, min/max delays, or false-path exceptions found
8	multout_ab[8]	No output delay, min/max delays, or false-path exceptions found
9	multout_ab[7]	No output delay, min/max delays, or false-path exceptions found
10	multout_ab[6]	No output delay, min/max delays, or false-path exceptions found
11	multout_ab[5]	No output delay, min/max delays, or false-path exceptions found
12	multout_ab[4]	No output delay, min/max delays, or false-path exceptions found

- Same report as before used for unconstrained clocks (Clock Status Summary report)
- Setup and Hold Analysis folders list unconstrained I/O ports and paths

Verifying Clocks & I/O Timing

- Use Setup & Hold Summary reports to check worst slack for each clock

“Did I make it or did I not make it?”

- Positive slack displayed in **black**, negative in **red**
- Obtaining summary reports
 - Use `create_timing_summary` Tcl command
 - TimeQuest folder of Compilation Report
 - Run Report Setup Summary & Report Hold Summary reports from Tasks pane or Reports menu
- For detailed slack/path analysis
 - Run Report Timing from Tasks pane or Constraints menu
 - Use `report_timing` command

SDC Timing Constraints

- Clocks
- I/O
- False paths ←
- Multicycle paths

Timing Exceptions: False Paths

- Logic-based
 - Paths not relevant during normal circuit operation
 - e.g. Test logic, static or quasi-static registers
- Timing-based
 - Paths intentionally not analyzed by designer
 - e.g. Bridging asynchronous clock domains using synchronizer circuits
- Must be marked by constraint to tell TimeQuest to ignore them

Two Methods to Create False Paths

■ `set_false_path` command

- Use when particular nodes are involved
- Examples
 - All paths from an input pin to a set of registers
 - All paths from a register to another clock domain

■ `set_clock_groups` command

- Use when just clock domains are involved

set_false_path Command

- Indicates paths that should be ignored during fitting and timing analysis

- Options

- `[-fall_from <clocks>]`

- `[-rise_from <clocks>]`

- `[-from <names>]`

- `[-through <names>]`

- `[-to <names>]`

- `[-fall_to <clocks>]`

- `[-rise_to <clocks>]`

- `[-setup]`

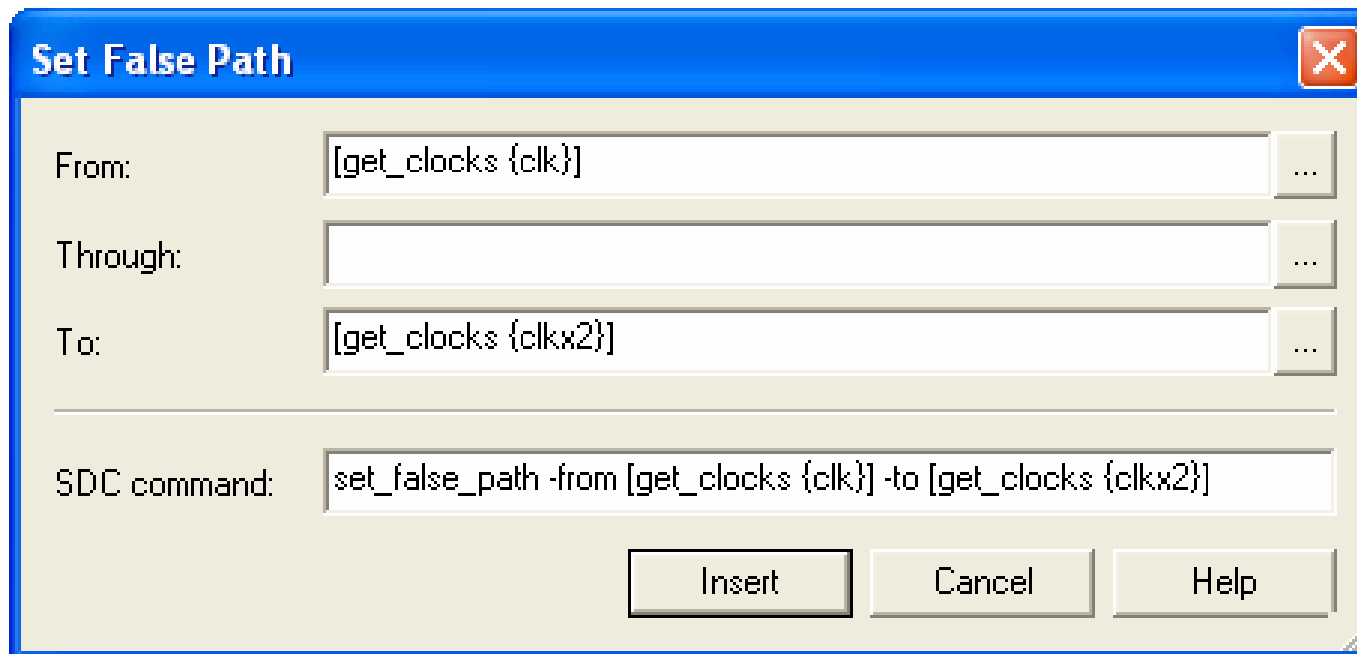
- `[-hold]`

- `<targets>`

set_false_path Notes

- `-from` & `-to`: Use to specify source & target nodes
 - Target nodes can be clocks, registers, ports, pins or cells
 - For registers, `-from` should be source register clock pin
 - Specify a clock name to constrain all paths going into or out of its domain
 - Constrains both rising and falling edge clock transitions
 - More efficient than specifying individual nodes
- `-rise_from` & `-fall_from`: Use to indicate clocks for the source node & whether constraint is for a rising or falling edge clock transition; *not in GUI*
- `-rise_to` & `-fall_to`: Use to indicate clocks for destination node & direction of transition; *not in GUI*
- `-setup` & `-hold`: Use to apply false paths to only setup/recovery or hold/removal analysis; *not in GUI*

Set False Path (GUI)



The image shows a GUI dialog box titled "Set False Path". It has a blue title bar with a close button (X) in the top right corner. The dialog contains three input fields: "From:" with the text "[get_clocks {clk}]", "Through:" which is empty, and "To:" with the text "[get_clocks {clkx2}]". Each of these three fields has a small button with three dots to its right. Below these fields is a horizontal line, and then an "SDC command:" field containing the text "set_false_path -from [get_clocks {clk}] -to [get_clocks {clkx2}]". At the bottom of the dialog are three buttons: "Insert", "Cancel", and "Help".

Set False Path

From: [get_clocks {clk}] ...

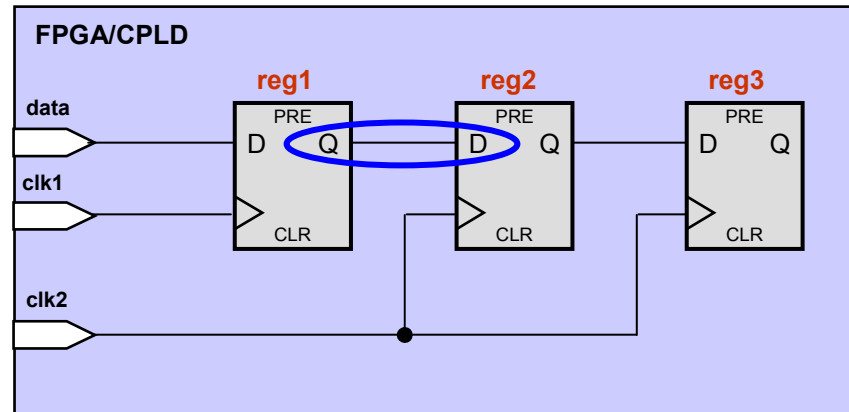
Through: ...

To: [get_clocks {clkx2}] ...

SDC command: set_false_path -from [get_clocks {clk}] -to [get_clocks {clkx2}]

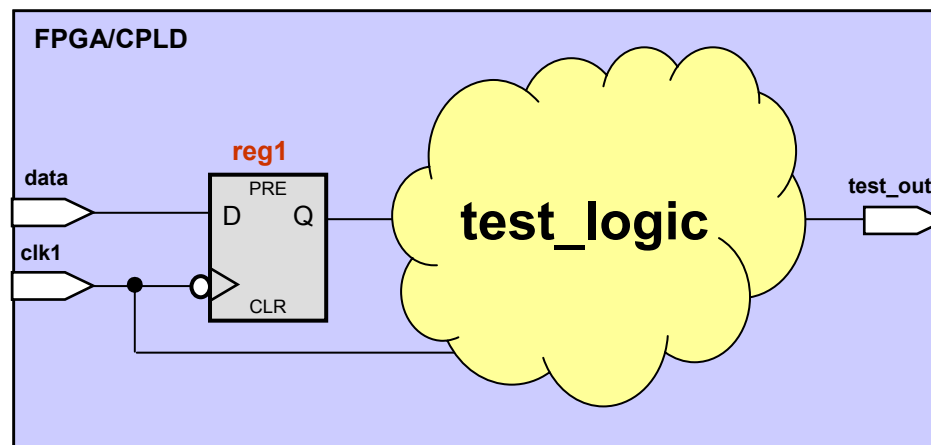
Insert Cancel Help

False Path Example 1



```
set_false_path -from [get_pins reg1|clk] \  
               -to [get_pins reg2|datain]
```

False Path Example 2



```
set_false_path -fall_from clk1 \  
               -to [get_pins test_logic|*|datain]
```

```
set_false_path -from [get_pins test_logic|*|clk] \  
               -to [get_pins test_logic|*|datain]
```

```
set_false_path -from [get_pins test_logic|*|clk] \  
               -to [get_ports test_out]
```

set_clock_groups Command

- Tells fitter and timing analyzer to ignore **ALL** paths between specified clock domains
 - Great for clock muxes
 - Equivalent to setting false paths (`-from` & `-to`) on all paths between domains

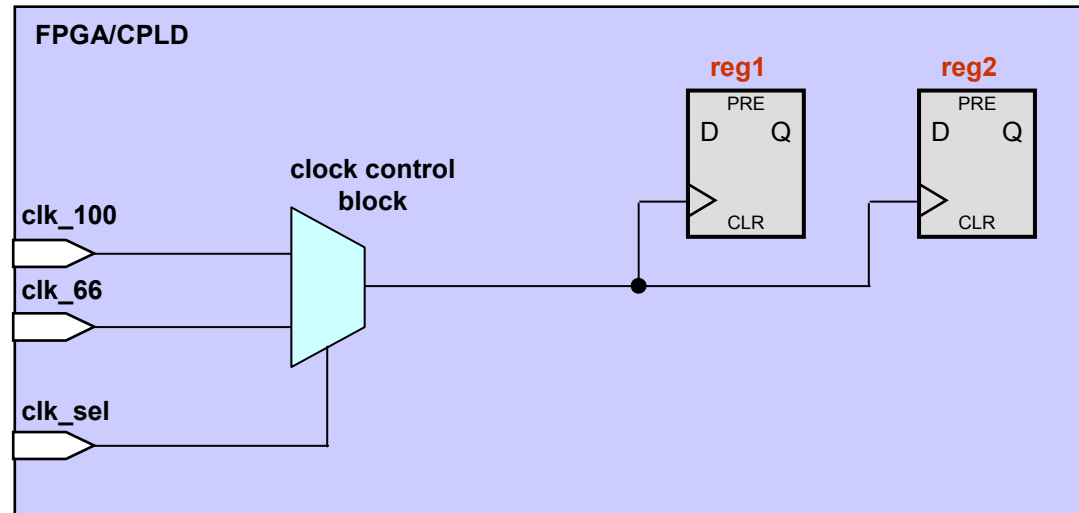
- Options

```
[-asynchronous | -exclusive]  
-group <clock name>  
-group <clock_name>  
[-group <clock name>]...
```


set_clock_groups Notes

- -group: each group of clock names is asynchronous to other clock groups
 - e.g. `set_clock_group -group {clkA clkB} \`
`-group {clkC clkD}`
- -asynchronous: no phase relationship, but clocks active at the same time
- -exclusive: clocks *not* active at the same time
 - Example: clock muxes

Clock Mux Example 1



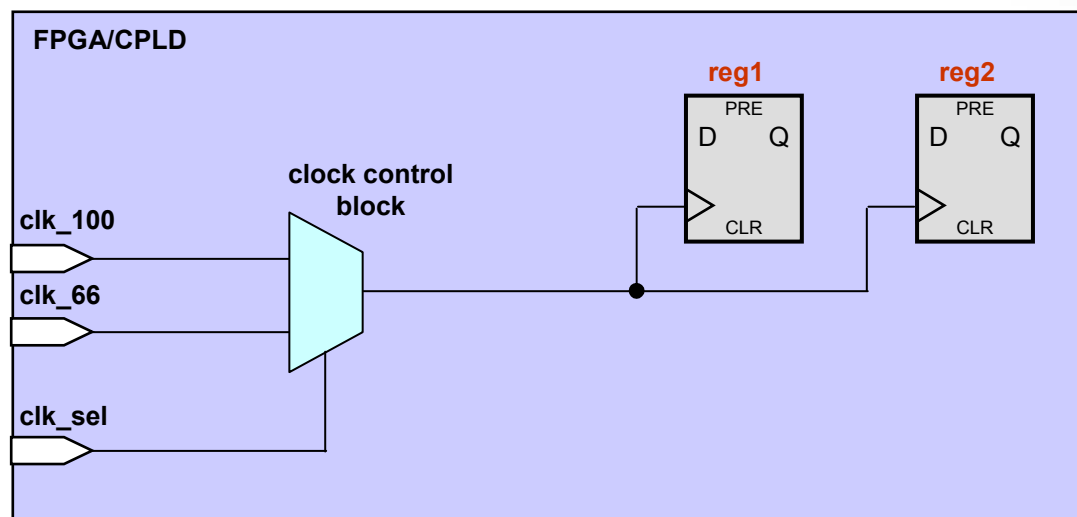
```
create_clock -period 10.0 [get_ports clk_100]
```

```
create_clock -period 15.0 [get_ports clk_66]
```

```
set_clock_groups -exclusive -group {clk_100} -group {clk_66}
```

```
# Since clocks are muxed, TimeQuest should not analyze  
# cross-domain paths as only one clock will be driving the  
# registers at any one time.
```

Clock Mux Example 1 (Alternative)

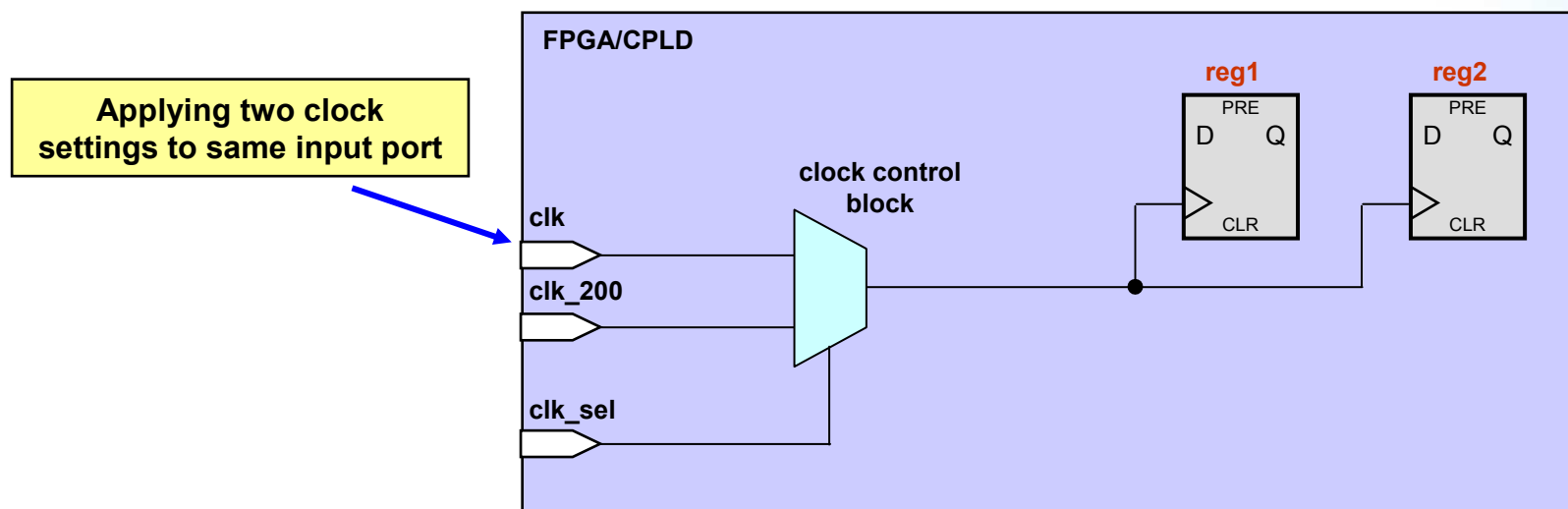


```
create_clock -period 10.0 [get_ports clk_100]
create_clock -period 15.0 [get_ports clk_66]
```

```
set_false_paths -from [get_clocks clk_100] -to [get_clocks clk_66]
set_false_paths -from [get_clocks clk_66] -to [get_clocks clk_100]
```

```
# For an equivalent constraint using false paths, you must
# consider paths going both directions
```

Clock Mux Example 2

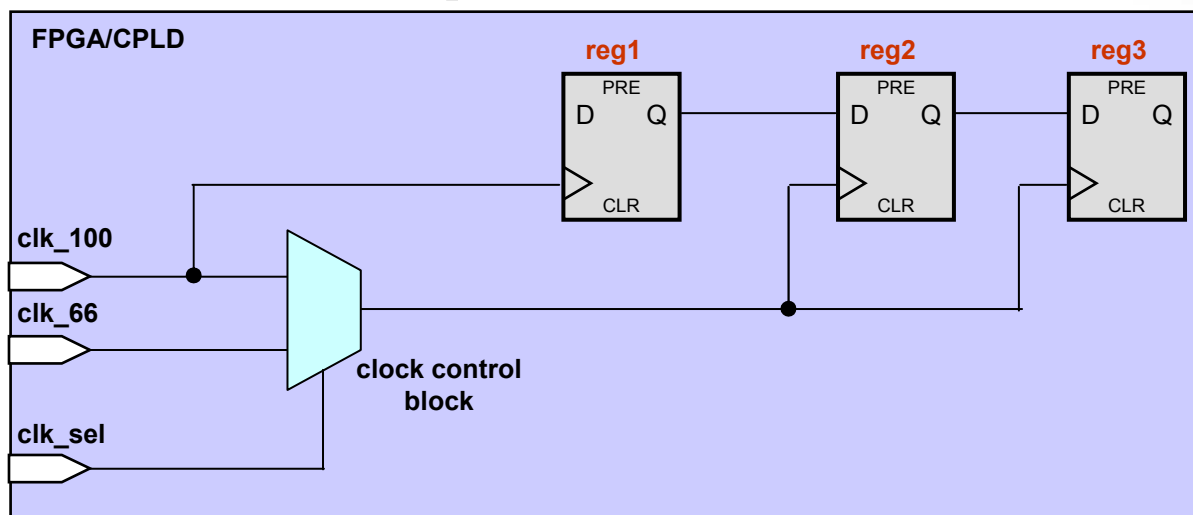


```
create_clock -name clk_100 -period 10.0 [get_ports clk]
create_clock -name clk_66 -period 15.0 [get_ports clk] -add
create_clock -period 5.0 [get_ports clk_200]
```

```
set_clock_groups -exclusive -group {clk_100} \
    -group {clk_66} -group {clk_200}
```

```
# As before, never will more than one clock be driving all
# registers
```

Clock Mux Example 3



```
create_clock -period 10.0 [get_ports clk_100]
```

```
create_clock -period 15.0 [get_ports clk_66]
```

```
create_generated_clock -name clkmux_100 -source clk_100 \  
    [get_pins clkmux|clkout]
```

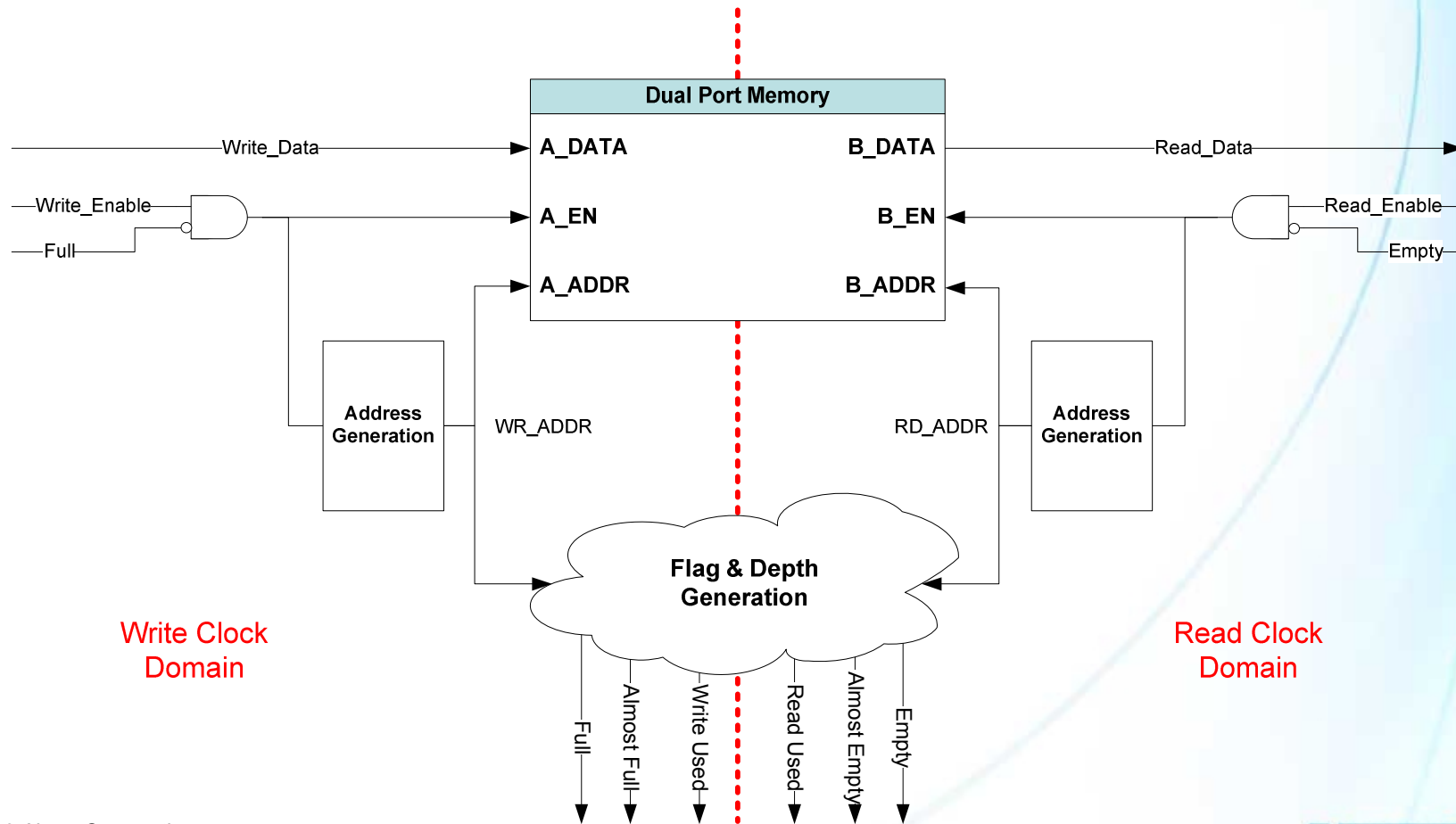
```
create_generated_clock -name clkmux_66 -source clk_66 \  
    [get_pins clkmux|clkout] -add
```

```
set_clock_groups -exclusive -group {clkmux_100} -group {clkmux_66}
```

```
# Since clk_100 is also feeding into the core, now you need to make generated  
# clocks on the mux outputs and use them for the clock groups
```

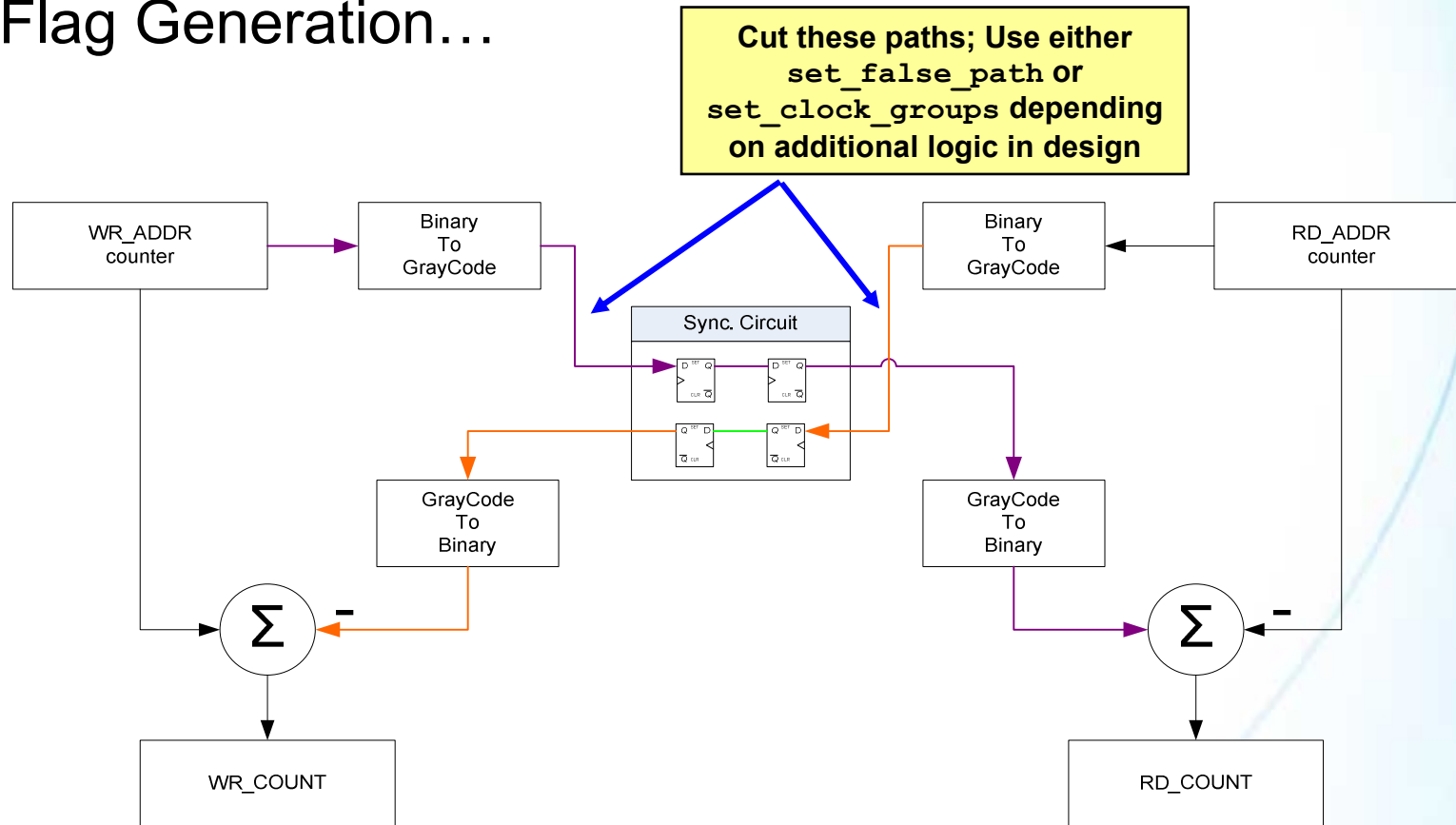
Real World Example: Memory FIFO

- FIFO bridging two clock domains; Flags indicate status of FIFO



False Paths on FIFO

■ Flag Generation...



Verifying False Paths & Groups

■ False paths

- Perform report timing on specified paths to ensure no results are returned
- Create false paths report
 - `report_timing -false_path`
 - **Tasks** pane or **Reports** menu: **Report False Path**

■ Clock groups

- Check clock transfers to ensure no paths are returned
 - `report_clk_transfers`
 - **Tasks** pane or **Reports** menu: **Report Clock Transfers**

SDC Timing Constraints

- Clocks
- I/O
- False paths
- Multicycle paths ←

Timing Exceptions: Multicycle Paths

- Paths requiring more than one cycle for data to propagate
- Causes timing analyzer to select another latch or launch edge
- Designer specifies number of cycles to move edge
- Logic *must* be designed to work this way
 - Constraint informs timing analysis how logic is supposed to function

Other Instances to Use Multicycle Paths

- Design does not require single cycle to transfer data (non-critical paths)
 - Otherwise needlessly over-constrain paths
- Clocks are integer multiples of each other with or without offset
 - Demonstrated in Exercise 4
- Clock enables ensuring register(s) not sampling data every clock edge

Multicycle Types (1)

■ Destination

- Constraint based on destination clock edges
- Moves latch edge backward (later in time) to relax required setup/hold time
- Used in most multicycle situations

■ Source

- Constraint based on source clock edges
- Moves launch edge forward (earlier in time) to relax required setup/hold time
- Useful when source clock is at higher frequency than destination

Multicycle Types (2)

■ Setup

- Increases the number of cycles for setup analysis
- Default is 1

■ Hold

- Increases the number of cycles for hold analysis
- Default is 0

***Notes:**

- 1) *Subtract 1 from the Classic Timing Analyzer hold multicycle value to convert to SDC*
- 2) *TimeQuest TA also supports negative multicycles*

set_multicycle_path Command

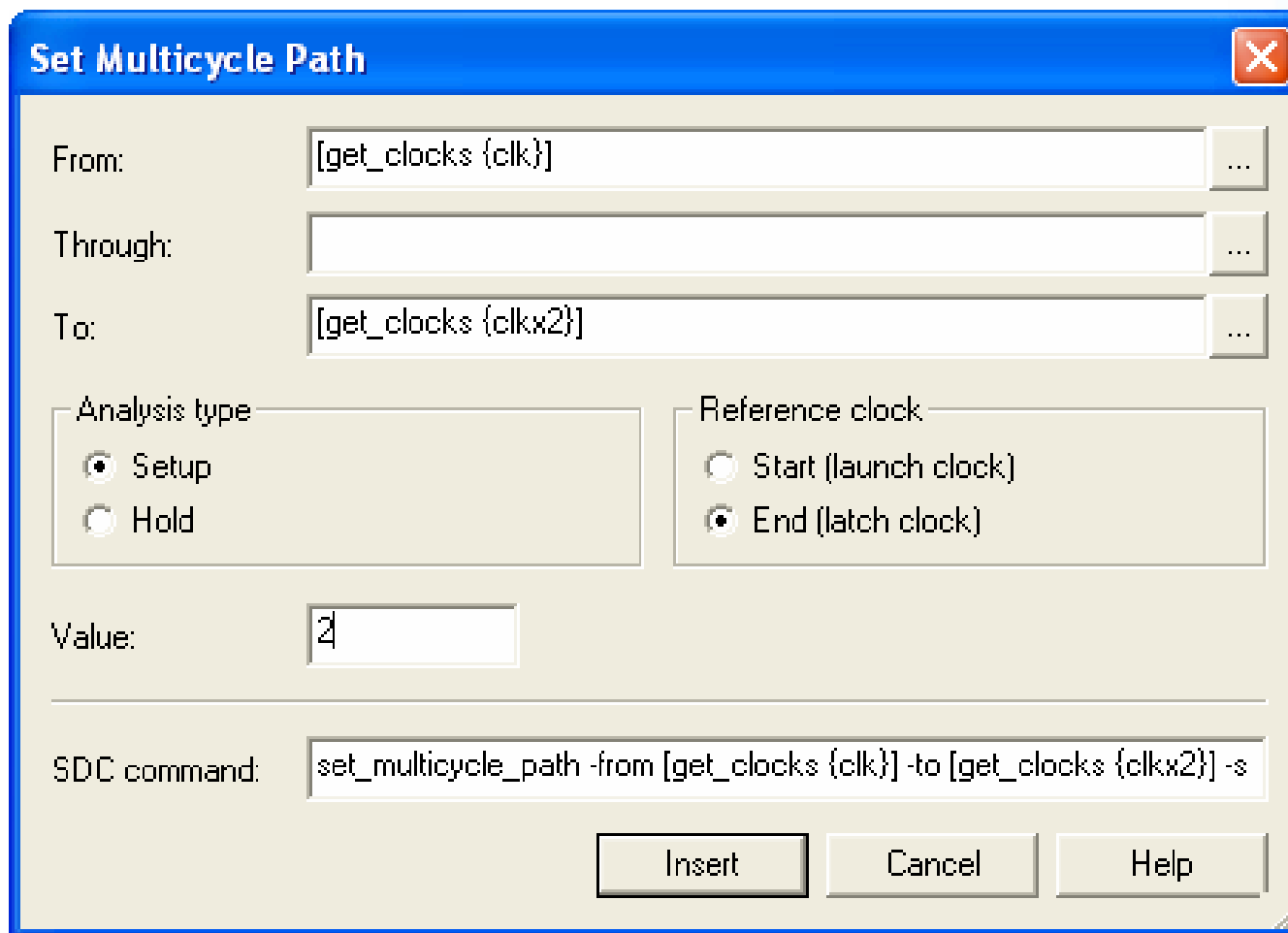
- Indicates by how many cycles the required time (setup or hold) should be extended from defaults
- Options

```
[-start | -end]  
[-setup | -hold]  
[-fall_from <clocks>]  
[-rise_from <clocks>]  
[-from <names>]  
[-through <names>]  
[-to <names>]  
[-fall_to <clocks>]  
[-rise_to <clocks>]  
<targets>  
<value>
```

set_multicycle_path Notes

- `-start`: Use to select a source multicycle
- `-end`: Use to select a destination multicycle (default)
- `-setup` | `-hold`: Specifies if the multicycle value is applied to the setup or hold calculation
- `<value>`: Cycle multiplier - Number of edges by which to extend analysis
- All other options behave similar to `set_false_path` options

Set Multicycle Path (GUI)



The dialog box is titled "Set Multicycle Path" and features a blue title bar with a close button. It contains several input fields and radio button groups. The "From:" field is set to "[get_clocks {clk}]", the "Through:" field is empty, and the "To:" field is set to "[get_clocks {clkx2}]". Each of these fields has a browse button (three dots). Below these fields are two groups of radio buttons. The "Analysis type" group has "Setup" selected and "Hold" unselected. The "Reference clock" group has "Start (launch clock)" unselected and "End (latch clock)" selected. A "Value:" field contains the number "2". At the bottom, an "SDC command:" field displays the command "set_multicycle_path -from [get_clocks {clk}] -to [get_clocks {clkx2}] -s". Three buttons labeled "Insert", "Cancel", and "Help" are at the bottom right.

Set Multicycle Path

From: [get_clocks {clk}] ...

Through: ...

To: [get_clocks {clkx2}] ...

Analysis type

- ☒ Setup
- ☐ Hold

Reference clock

- ☐ Start (launch clock)
- ☒ End (latch clock)

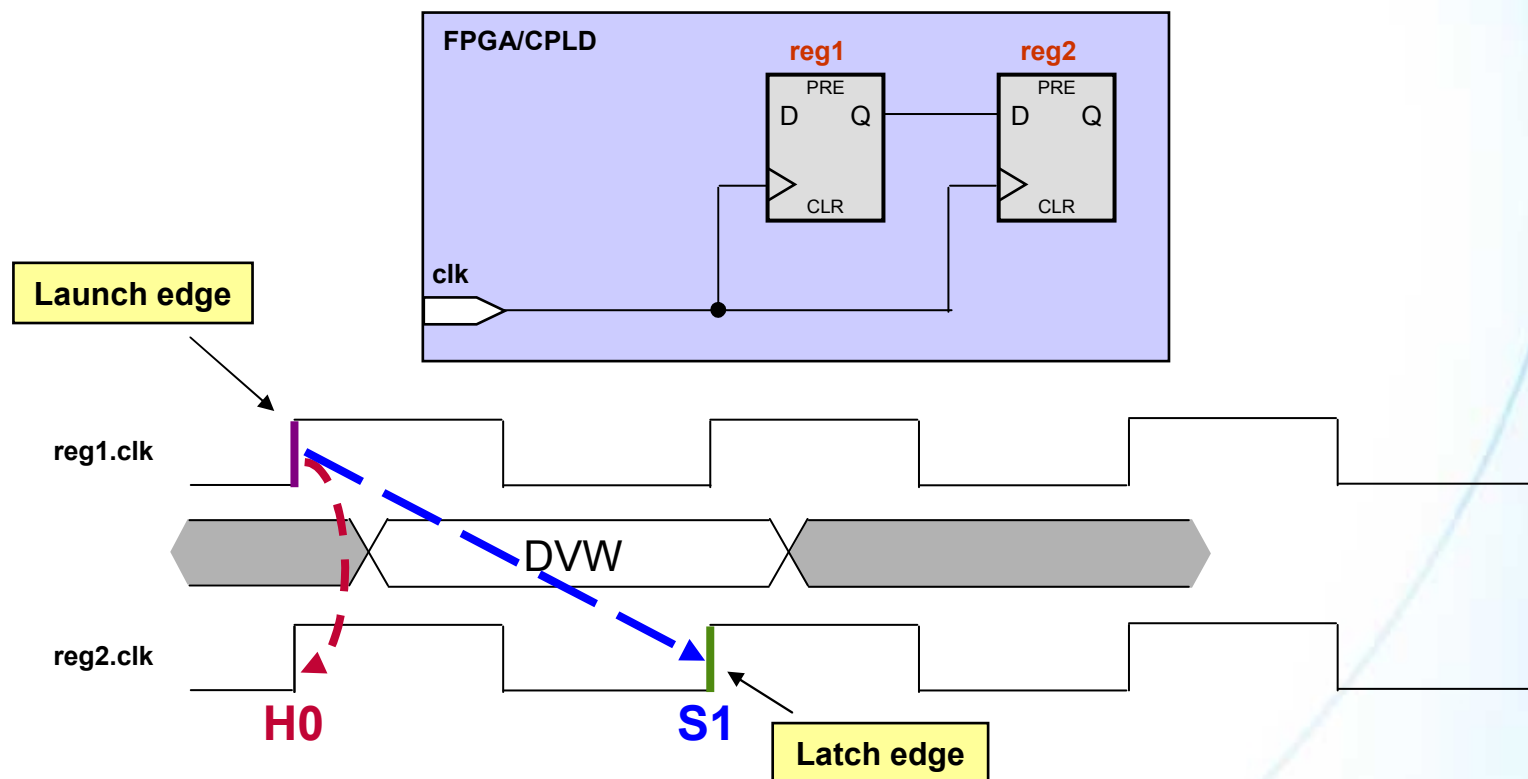
Value: 2

SDC command: set_multicycle_path -from [get_clocks {clk}] -to [get_clocks {clkx2}] -s

Insert Cancel Help

Understanding Multicycle (1)

Standard single-cycle register transfer



— — — Multicycle Setup = 1 (Default)

- - - Multicycle Hold = 0 (Default)*

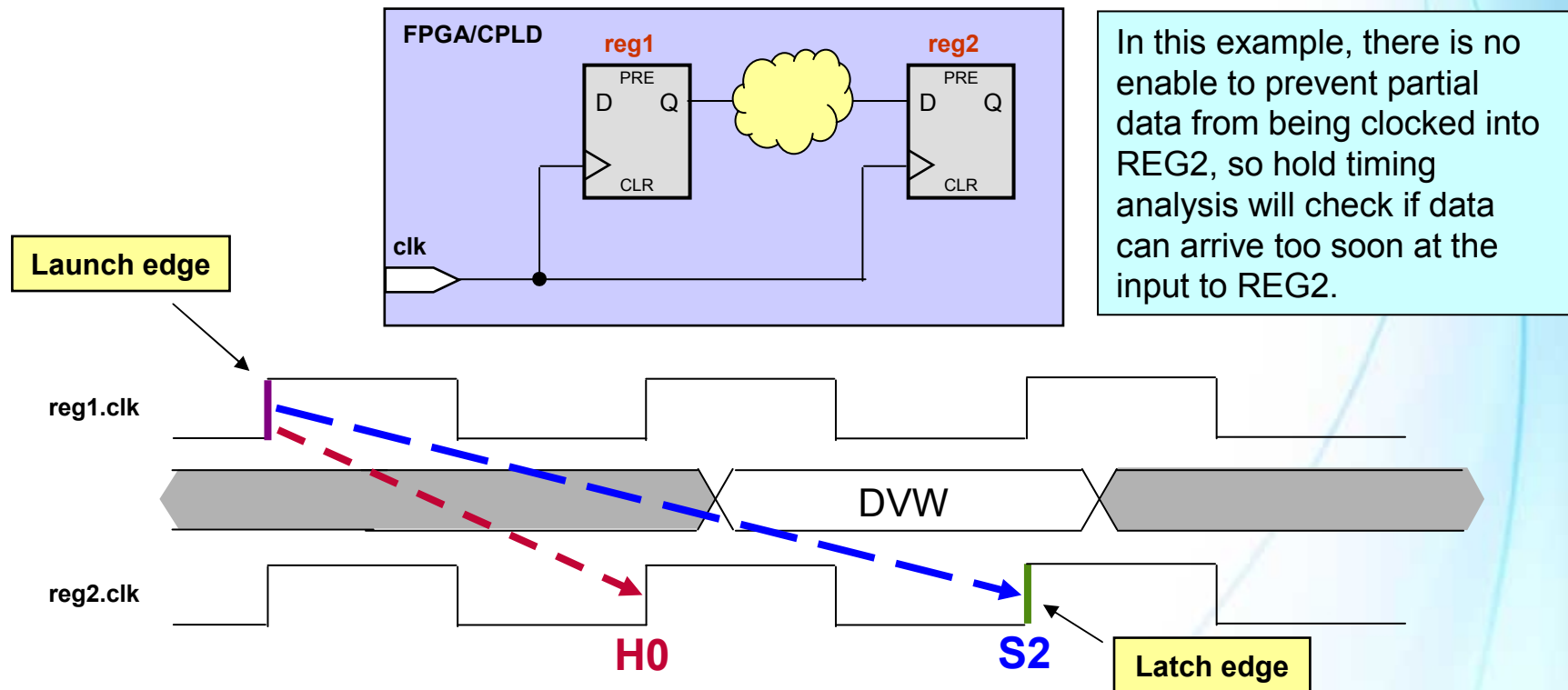
**Default hold edge is one edge before/after setup edge*

© 2009 Altera Corporation

Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation

Understanding Multicycle (2)

Change to a *two cycle setup; single cycle hold* transfer



- — . Multicycle Setup = 2
- - - Multicycle Hold = 0 (Default)

***Default hold edge is one edge before/after setup edge; hold edge moves with setup edge**

© 2009 Altera Corporation

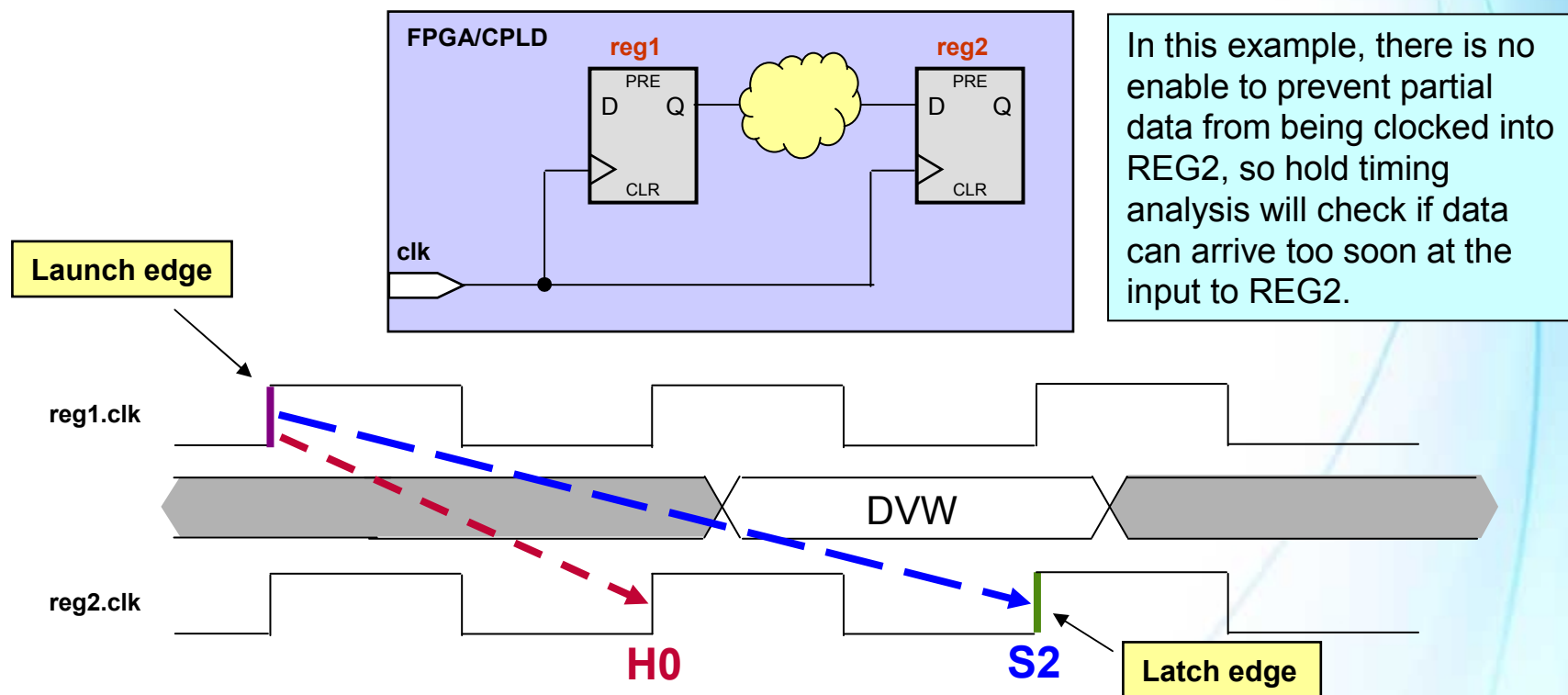
Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation

110

ALTERA

Understanding Multicycle (2) (cont.)

Change to a *two cycle setup; single cycle hold* transfer



```
set_multicycle_path -from [get_pins reg1|clk] -to [get_pins reg2|datain] \  
-end -setup 2
```

***Default hold edge is one edge before/after setup edge; hold edge moves with setup edge**

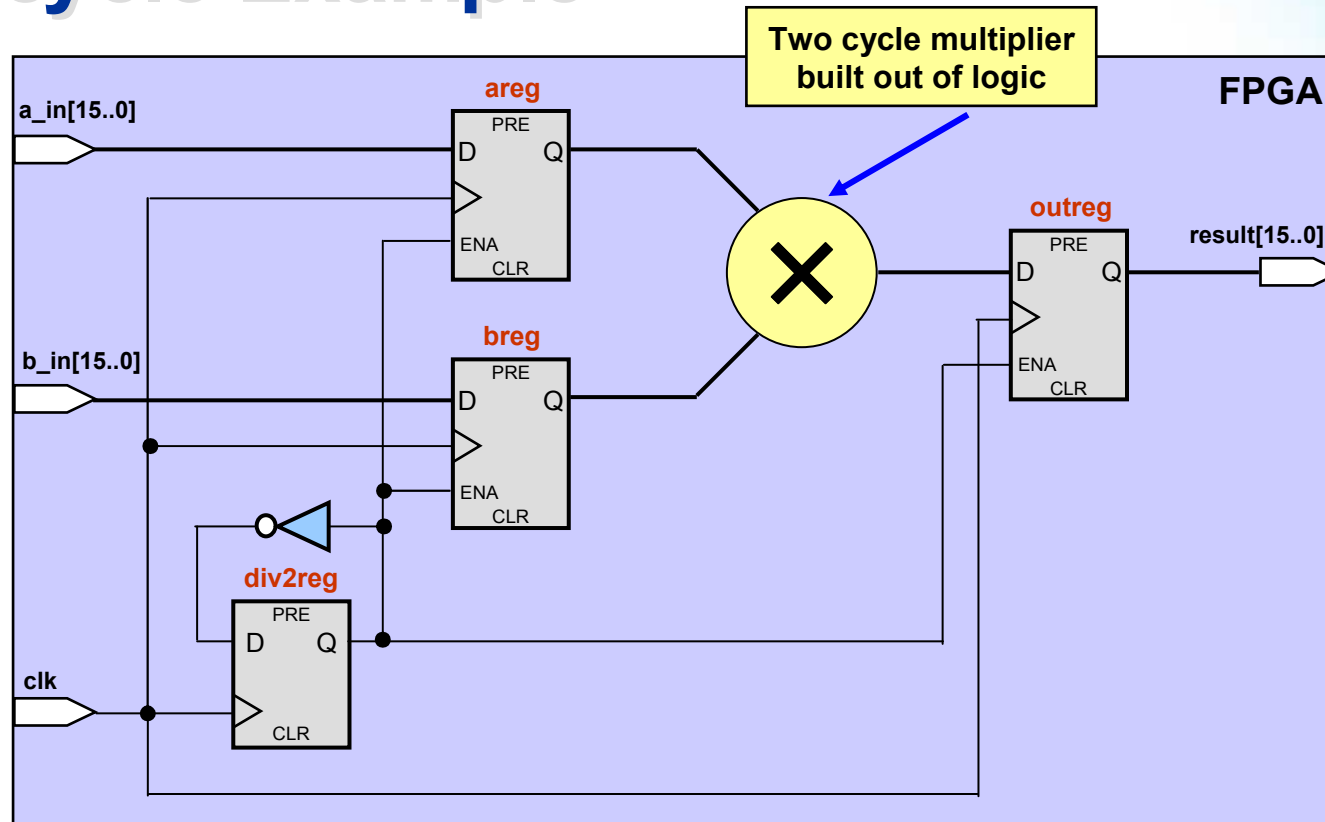
© 2009 Altera Corporation

Altera, Stratix, Arria, Cyclone, MAX, HardCopy, Nios, Quartus, and MegaCore are trademarks of Altera Corporation

111

ALTERA

Multicycle Example

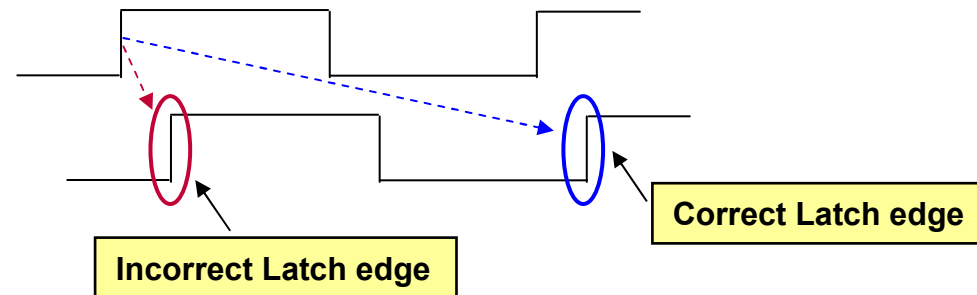


```
# Need to specify that the multiplier is allowed 2 cycles to compute a result
# Note this has already been determined by design (half-rate clock enable)
set_multicycle_path -from [get_pins {areg*|clk breg*|clk}] \
    -to [get_pins outreg*|datain] -end -setup 2

set_multicycle_path -from [get_pins {areg*|clk breg*|clk}] \
    -to [get_pins outreg*|datain] -end -hold 1
```

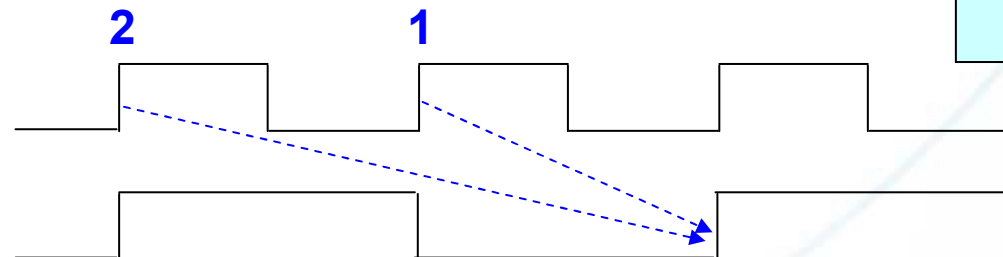
Other Multicycle Cases

■ Positive clock phase shift or offset



■ Source clock at higher frequency

- Use `-start` option



Note using the `-start` option moves the latch edge forward one edge (to relax constraint)

Reporting Multicycles

Report Timing: No multicycle

Command Info Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock
1	-2.357	y_regtwo[2]	...OBSERVABLEDATAB_REGOUT2	c100	c200

No Multicycle

Path #1: Setup slack is -2.357 (VIOLATED)

Path Summary Statistics Data Path Waveform

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	0.091	0.091	R				clock network delay
3	0.341	0.250		uTco	1	LCFF_X27_Y7_N7	y_regtwo[2]
4	0.341	0.000	RR	CELL	1	LCFF_X27_Y7_N7	y_regtwo[2]regout
5	0.341	0.000	RR	IC	1	LCCOMB_X27_Y7_N6	inst24[inst[2]]datac
6	0.664	0.323	RR	CELL	1	LCCOMB_X27_Y7_N6	inst24[inst[2]]combout
7	0.909	0.245	RR	IC	1	LCCOMB_X27_Y7_N0	inst24[inst11[2]]datad
8	1.058	0.149	RR	CELL	1	LCCOMB_X27_Y7_N0	...24[inst11[2]]combout

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	5.000	5.000					latch edge time
2	5.136	0.136	R				clock network delay
3	5.089	-0.047		uTsu	14	DSPMULT_X16_Y10_N0	...BLEDATAB_REGOUT2

Reporting Multicycles

Report Timing: set_multicycle_path

Command Info Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock
1	2.643	y_regtwo[2]	...OBSERVABLEDATAB_REGOUT2	c100	c200

Same path with Setup Multicycle = 2

Path #1: Setup slack is 2.643

Path Summary Statistics Data Path Waveform

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	0.091	0.091	R				clock network delay
3	0.341	0.250		uTco	1	LCFF_X27_Y7_N7	y_regtwo[2]
4	0.341	0.000	RR	CELL	1	LCFF_X27_Y7_N7	y_regtwo[2]regout
5	0.341	0.000	RR	IC	1	LCCOMB_X27_Y7_N6	inst24[inst2]datac
6	0.664	0.323	RR	CELL	1	LCCOMB_X27_Y7_N6	inst24[inst2]combout
7	0.909	0.245	RR	IC	1	LCCOMB_X27_Y7_N0	inst24[inst11[2]]datad
8	1.058	0.149	RR	CELL	1	LCCOMB_X27_Y7_N0	inst24[inst11[2]]combout
9	1.303	0.245	RR	IC	1	LCCOMB_X27_Y7_N26	inst24[inst12[2]]datad
10	1.452	0.149	RR	CELL	1	LCCOMB_X27_Y7_N26	inst24[inst12[2]]combout
11	1.700	0.248	RR	IC	1	LCCOMB_X27_Y7_N10	inst24[inst13[2]]datad

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	10.000	10.000					latch edge time
2	10.136	0.136	R				clock network delay
3	10.089	-0.047		uTsu			...LEDATAB_REGOUT2


Latch edge extended by one destination clock cycle

Path #1: Setup slack is 2.643

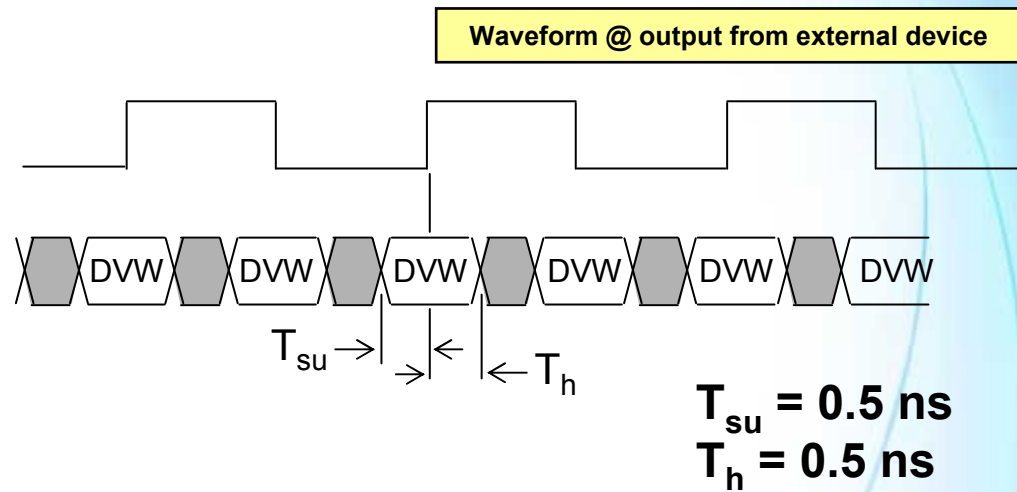
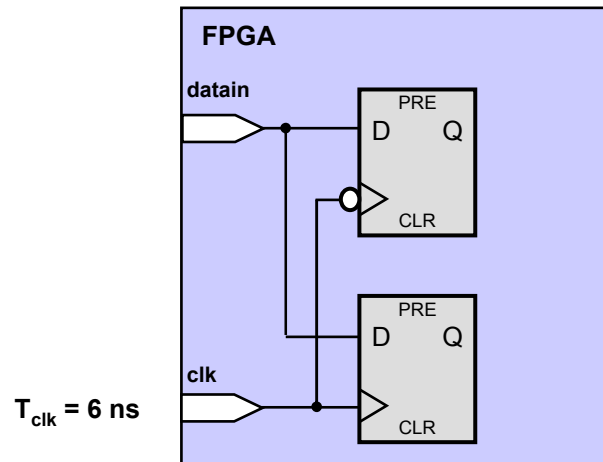
Path Summary Statistics Data Path Waveform

	Property	Value
1	From Node	y_regtwo[2]
2	To Node	...ABLEDATAB_REGOUT2
3	Launch Clock	c100
4	Latch Clock	c200
5	Multicycle - Setup End	2
6	Data Arrival Time	7.446
7	Data Required Time	10.089
8	Slack	2.643

Timing Analysis Agenda

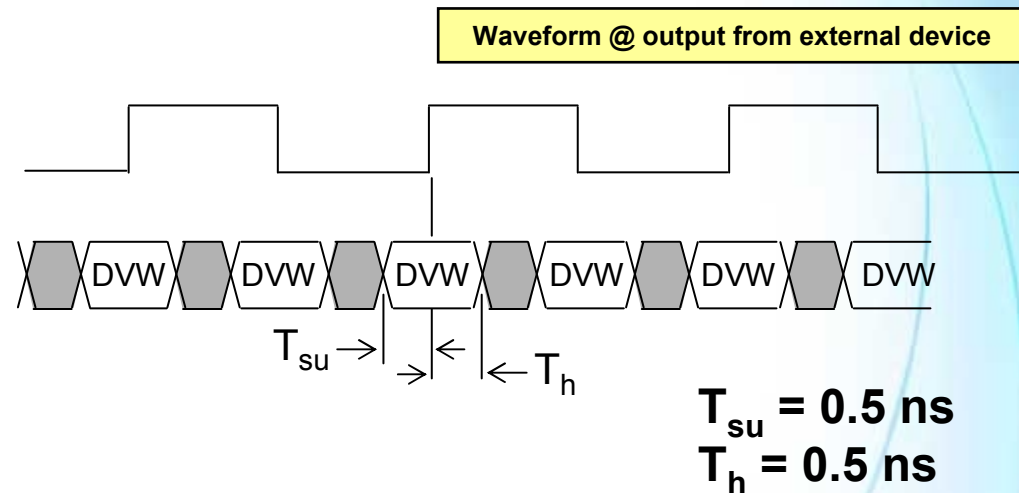
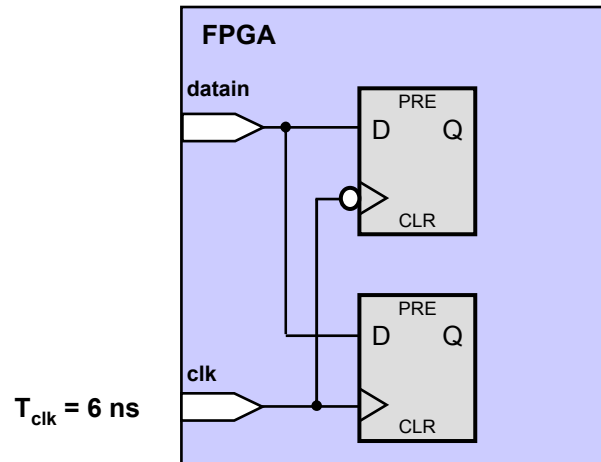
- TimeQuest basics
- Timing constraints
- Example 

DDR Input Example



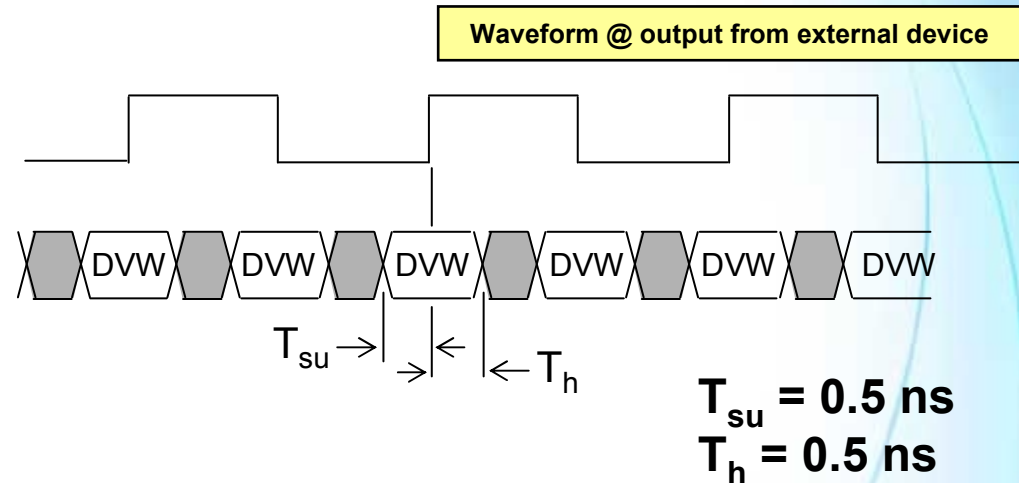
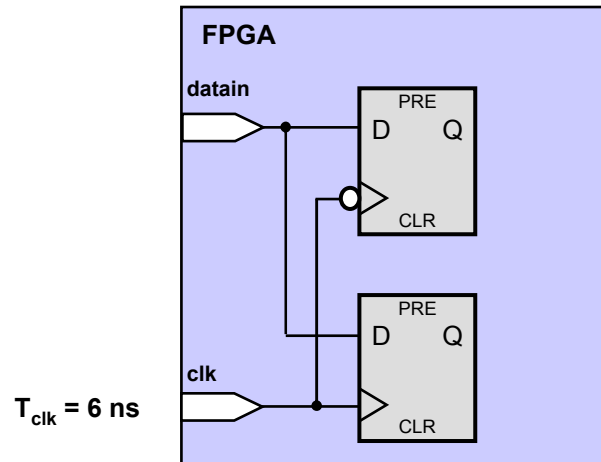
- What constraints do you need?
- Clock
- Input delay maximum & minimum
 - Use source-synchronous methodology

DDR Input Example



- What's different about this circuit than prior examples?
- Rising & falling edge input registers from same input port
- Registers have $\frac{1}{2}$ clock period for required time

DDR Input Example



```
create_clock -period 6 [get_ports clk]
```

```
# Rising edge clock constraint
```

```
set_input_delay -clock clk -max [expr 6 / 2 - 0.5] datain
```

```
set_input_delay -clock clk -min 0.5 datain
```

```
# Falling clock edge constraint
```

```
set_input_delay -clock clk -max [expr 6 / 2 - 0.5] datain \
```

```
    -clock_fall -add_delay
```

```
set_input_delay -clock clk -min 0.5 datain -clock_fall -add_delay
```

DDR Reporting

- Use `report_timing` Command
- Must check all rising & falling edge transitions
 - Two data valid windows to check
 - One from a rising edge source clock
 - One from a falling edge source clock
 - Use `rise_from`, `rise_to`, `fall_from`, `fall_to`

Timing Analysis Summary

- Timing constraints are very important in FPGA/CPLD design
- Use timing constraints to tell fitter & timing analyzer how logic is designed to function
- SDC provides an easy-to-use, standard interface for constraining design
- See the Quartus II Handbook: Volume 3, Section II, for more information about timing analysis

Reference Documents

- Quartus II Handbook, Volume 3, Chapter 7 The Quartus II TimeQuest Timing Analyzer

http://www.altera.com/literature/hb/qts/qts_qii53018.pdf

- Quick Start Tutorial

http://www.altera.com/literature/hb/qts/ug_tq_tutorial.pdf

- Cookbook

– http://www.altera.com/literature/manual/mnl_timequest_cookbook.pdf

Reference Documents

- SDC and TimeQuest API Reference Manual
 - http://www.altera.com/literature/manual/mnl_sdctmq.pdf
- AN 481: Applying Multicycle Exceptions in the TimeQuest Timing Analyzer
 - <http://www.altera.com/literature/an/an481.pdf>
- AN 433: Constraining and Analyzing Source-Synchronous Interfaces
 - <http://www.altera.com/literature/an/an433.pdf>

Learn More Through Technical Training

Instructor-Led Training



With Altera's instructor-led training courses, you can:

- Listen to a lecture from an Altera technical training engineer (instructor)
- Complete hands-on exercises with guidance from an Altera instructor
- Ask questions & receive real-time answers from an Altera instructor
- Each instructor-led class is one or two days in length (8 working hours per day).

Online Training



With Altera's online training courses, you can:

- Take a course at any time that is convenient for you
- Take a course from the comfort of your home or office (no need to travel as with instructor-led courses)

Each online course will take approximate one to three hours to complete.

<http://www.altera.com/training>

View training class schedule & register for a class

Other Quartus II Design Series courses

- Quartus II Software Design Series: Foundation
 - Project creation and management
 - Design entry methods and tools
 - Compilation and compilation results analysis
 - Creating and editing settings and assignments
 - I/O planning and management
 - Introduction to timing analysis with TimeQuest

- Quartus II Software Design Series: Verification
 - Basic design simulation with ModelSim-Altera
 - Power analysis
 - Debugging solutions

- Quartus II Software Design Series: Optimization
 - Incremental compilation
 - Quartus II optimization features & techniques

Altera Technical Support

- Reference Quartus II software on-line help
- [Quartus II Handbook](#)
- Consult Altera applications (factory applications engineers)
 - MySupport: <http://www.altera.com/mysupport>
 - Hotline: (800) 800-EPLD (7:00 a.m. - 5:00 p.m. PST)
- Field applications engineers: contact your local Altera sales office
- Receive literature by mail: (888) 3-ALTERA
- FTP: <ftp.altera.com>
- World-wide web: <http://www.altera.com>
 - Use solutions to search for answers to technical problems
 - View design examples