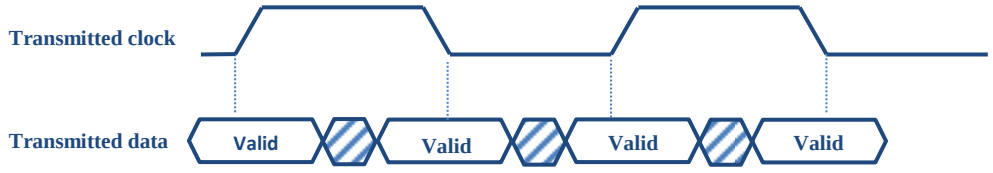
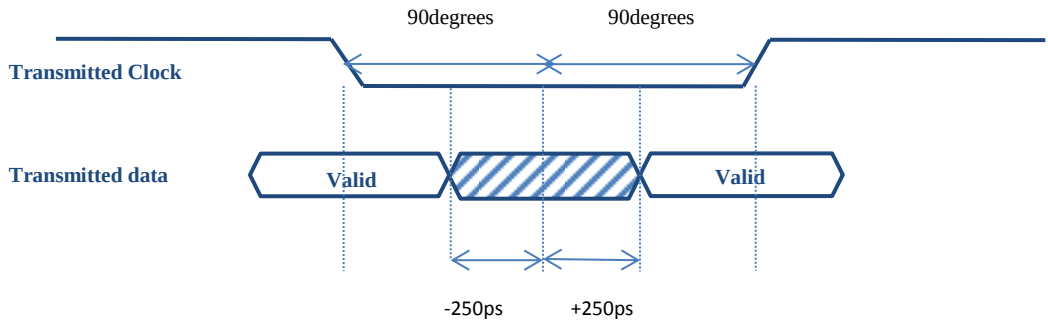
Constraining Center Aligned Source Synchronous input interfaces in TimeQuest.

Overview:

A Center aligned interface is where the transmitted clock edges are positioned in the middle of the data valid window. Such interfaces require no internal clock phase shift in the receiving device as the clock is already in the ideal position to capture the data.



The performance of such an interface could be specified as skew such as +/- 0.25ns with a defined clock offset(ideally 90 degrees).



(If the clock to data relationship is specified using setup & hold parameters then the skew can be calculated from these figures.)

Implementation Types:

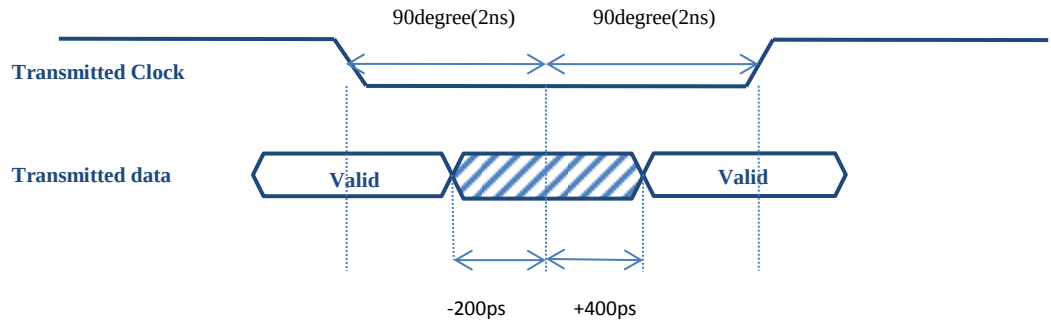
The following common implementation modes exist:

1) Direct clock connection mode.
    a. The clock is routed directly from the clock pin to the DDR input registers.
2) PLL in source synch mode.
    a. The clock is routed through a pll in source synch mode. In this mode the pll compensates for the clock insertion delay to maintain the clock to data relationship at the capture registers.

Direct clock connection mode:

In this mode the clock is routed directly from the clock input pin to the DDR input capture registers.

For the constraints below assume a 125MHz clock with a data to clock skew of -200ps to +400ps and a 90 degree shift on the clock.



(If the clock to data relationship is specified using setup & hold parameters then the skew can be calculated from these figures.)

Constraints:

Clocks:

Create a virtual clock to model the external clock source used to launch the data:

```
create_clock -name rx_clock_virt  -period 8
```

Theoretically, this is the clock to which the skew applies to.

Create a base clock to model the clock entering the fpga:

As this is a center aligned interface then we shift the input clock by 90 degrees (As this clock is effectively shifted by 90 degrees relative to the clock launching the data from the source device)

```
create_clock -name clkin -period 8 -waveform { 2 6 } [get_ports clkin]
```

Input Constraints:

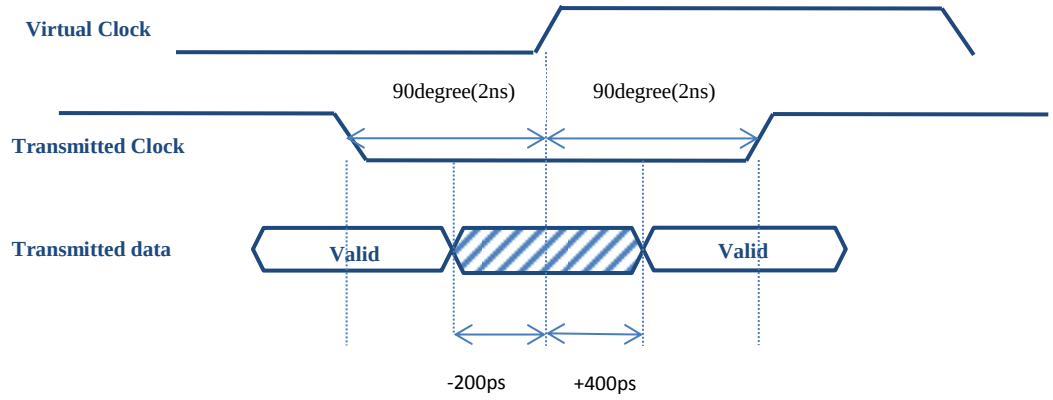
You can use a maximum skew specification to calculate input delay values. The maximum skew specification indicates the allowable time variation for individual bits of a data bus to arrive at the FPGA.

The value of the input maximum delay is maximum skew value(400ps).

(A positive max figure states that the data arrives “after” the virtual clock edge)

The value of the input minimum delay is -maximum skew value(-200ps).

(A negitive min figure states that the data arrives “before” the virtual clock edge)




```
set_input_delay -max  0.4 -clock [get_clocks rx_clock_virt] -add_delay {rx[*]}
set_input_delay -min -0.2 -clock [get_clocks rx_clock_virt] -add_delay {rx[*]}
#Use the –clock_fall argument to perform analysis for both rising and falling clock edges.
set_input_delay -max  0.4 -clock_fall -clock [get_clocks rx_clock_virt] -add_delay {rx[*]}
set_input_delay -min -0.2 -clock_fall -clock [get_clocks rx_clock_virt] -add_delay {rx[*]}
```

Timing Exceptions:

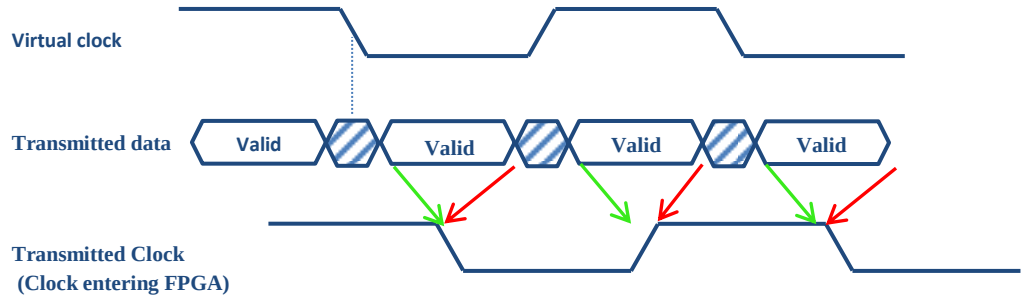
For Setup, the data is transferred as rise-rise (launched on rising edge and captured on rising edge) or fall-fall (launched on falling edge and captured on falling edge).

Hold analysis is performed as rise-fall (launched on rising edge and captured on falling edge) or fall-rise (launched on falling edge and captured on rising edge).

To prevent any unwanted relationships the following false path assignments are required:

```
set_false_path -fall_from [get_clocks rx_clock_virt] -rise_to clkin  -setup
set_false_path -rise_from [get_clocks rx_clock_virt] -fall_to clkin -setup
set_false_path -fall_from [get_clocks rx_clock_virt] -fall_to clkin -hold
set_false_path -rise_from [get_clocks rx_clock_virt] -rise_to clkin –hold
```

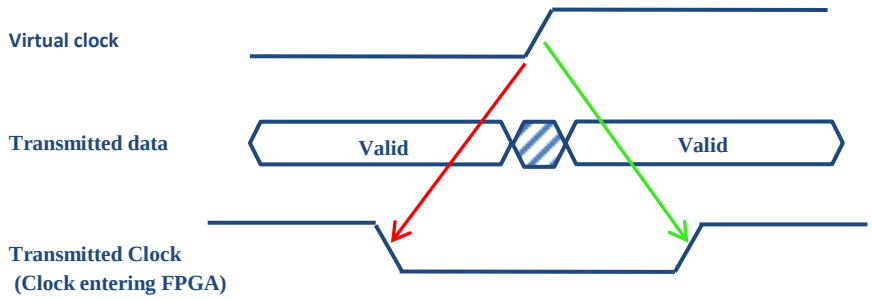
The Setup and hold analysis is shown below(Green is for setup, Red is for hold)



Note, the arrows above show which data transition point is used for setup and hold and the corresponding latch clock edges.

The following diagram is representative of the analysis of a single setup and hold path as would be analyzed in TimeQuest.

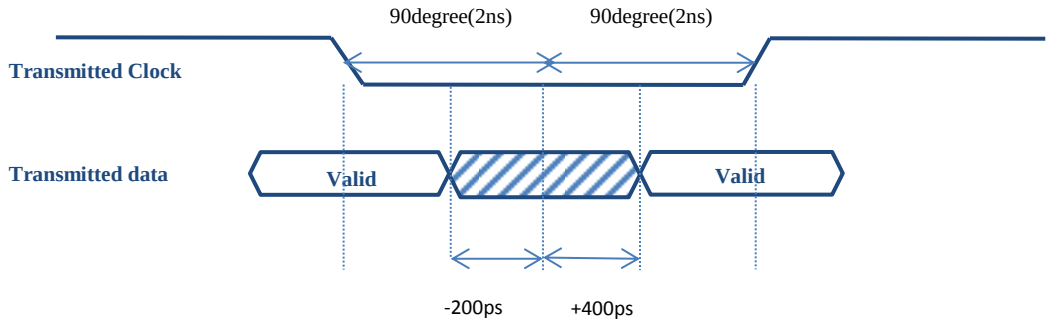
The transfers shown below are rise-rise(for setup) and rise-fall(for hold). (Green is for setup, Red is for hold)



PLL Capture mode:

In this mode the clock is routed through a pll in source synchronous compensation mode. The pll compensates for any clock insertion delay to maintain the clock to data relationship at the capture registers.

For the constraints below assume a 125MHz clock with a data to clock skew of -200ps to +400ps and a 90 degree shift on the clock.



(If the clock to data relationship is specified using setup & hold parameters then the skew can be calculated from these figures.)

Constraints:

Clocks:

Create a virtual clock to model the external clock source used to launch the data:

```
create_clock -name rx_clock_virt  -period 8
```

Create a base clock to model the clock entering the fpga:

As this is a center aligned interface then we shift the input clock by 90 degrees (As this clock is effectively shifted by 90 degrees relative to the clock launching the data from the source device)

```
create_clock -name clkin -period 8 -waveform { 2 6 } [get_ports clkin]
```

Create a generated clock for the PLL clock output:

```
create_generated_clock -name internal_clock -source [get_ports clock] \
[get_pins {UPLL|mpll_inst|altera_pll_i|general[0].gpll~PLL_OUTPUT_COUNTER|divclk}]
```

(Note that this constraint is usually automatically generated by derive_pll_clocks)

Input Constraints:

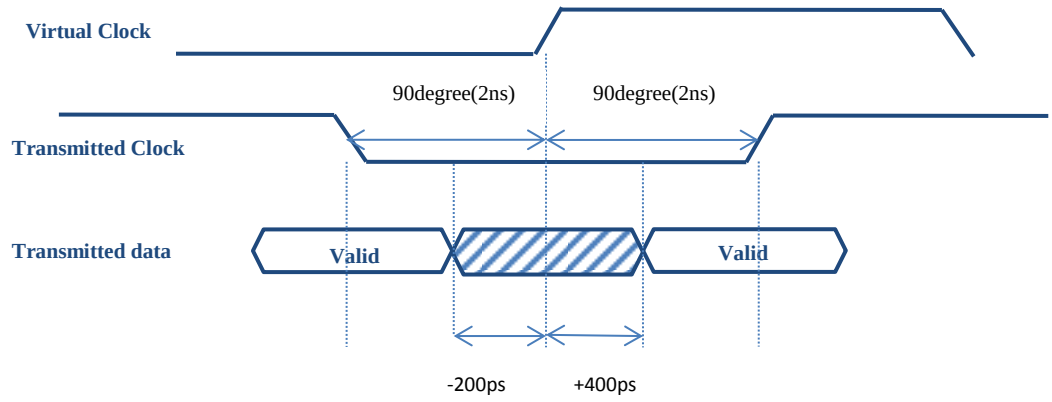
You can use a maximum skew specification to calculate input delay values. The maximum skew specification indicates the allowable time variation for individual bits of a data bus to arrive at the FPGA.

The value of the input maximum delay is maximum skew value(400ps).

(A positive max figure states that the data arrives “after” the clock edge)

The value of the input minimum delay is -maximum skew value(-200ps).

(A negitive min figure states that the data arrives “before” the clock edge)




```
set_input_delay -max  0.4 -clock [get_clocks rx_clock_virt] -add_delay {rx[*]}
set_input_delay -min -0.2 -clock [get_clocks rx_clock_virt] -add_delay {rx[*]}
#Use the –clock_fall argument to perform analysis for both rising and falling clock edges.
set_input_delay -max  0.4 -clock_fall -clock [get_clocks rx_clock_virt] -add_delay {rx[*]}
set_input_delay -min -0.2 -clock_fall -clock [get_clocks rx_clock_virt] -add_delay {rx[*]}
```

Timing Exceptions:

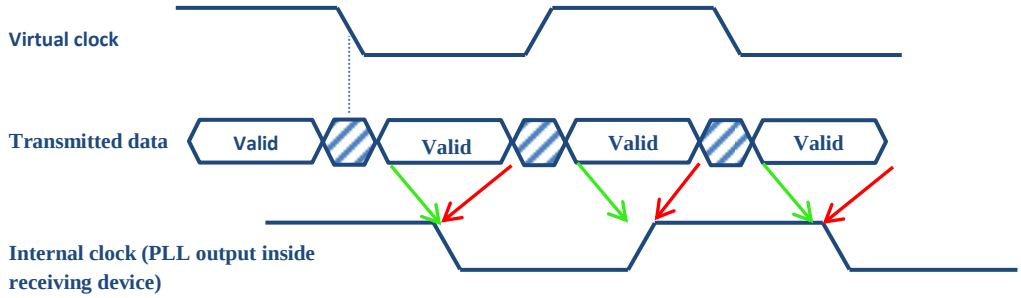
The data is transferred as rise-rise (launched on rising edge and captured on rising edge) or fall-fall (launched on rising edge and captured on rising edge).

Hold analysis is performed on opposite edges as rise-fall (launched on rising edge and captured on falling edge) or fall-rise (launched on falling edge and captured on rising edge).

To prevent any unwanted relationships the following false path assignments are required:

```
set_false_path -fall_from [get_clocks rx_clock_virt] -rise_to data_clock  -setup
set_false_path -rise_from [get_clocks rx_clock_virt] -fall_to data_clock -setup
set_false_path -fall_from [get_clocks rx_clock_virt] -fall_to data_clock -hold
set_false_path -rise_from [get_clocks rx_clock_virt] -rise_to data_clock -hold
```

The Setup and hold analysis is shown below(Green is for setup, Red is for hold)



Note, the arrows above show which data transition point is used for setup and hold and the corresponding latch clock edges.

The following diagram is representative of the analysis of a single setup and hold path as would be analyzed in TimeQuest.

The transfers shown below are rise-rise(for setup) and rise-fall(for hold).