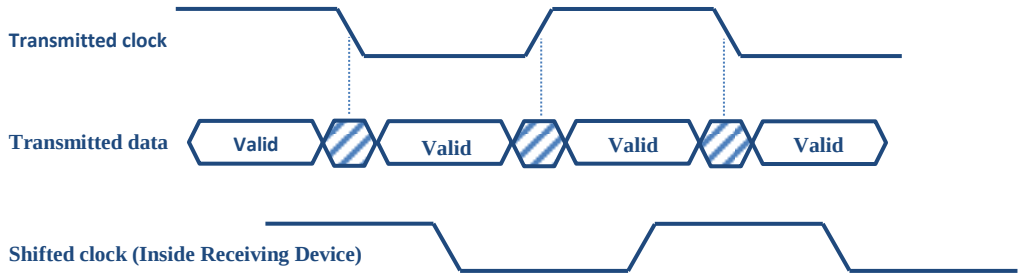
Constraining Edge Aligned Source Synch input interfaces in TimeQuest.

Overview:

An Edge Aligned interface is where the changing data is aligned with the active clock edge. Such interfaces usually require that the clock be shifted by 90Degrees in the receiving device to position the clock edge in the centre of the data valid window:



The performance of such an interface will normally be specified as skew such as +/- 0.25ns.

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

Implementation Types:

The user can implement a PLL (Most common method) to shift the clock by 90 degrees to align the clock in the centre of the data valid window or alternatively can rely on the clock insertion delay to achieve the same shift. Both methods result in slightly different constraints which are covered separately below.

PLL Capture mode:

In this mode a PLL is inserted into the receiving clock path to shift the clock by 90 degrees.

For the constraints below assume a 125MHz clock with a data to clock skew of -200ps to +400ps (data arrives from 200ps before the clock edge to 400ps after it).

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:

```
create_clock -name clkin -period 8 [get_ports clkin]
```

Create a 90 degree shifted clock for the PLL output:

```
create_generated_clock -name data_clock -source [get_pins pll|inclk[0]]    -phase 90 \
 [get_pins pll|clk[0]]
```

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

Input Constraints:

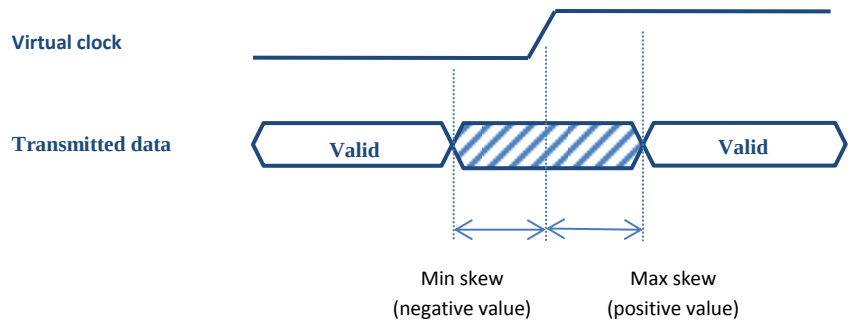
You can use a maximum skew specification to calculate input delay values. The maximum skew specification indicates the allowable time variation for individual bits of a data bus to arrive at the FPGA.

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:

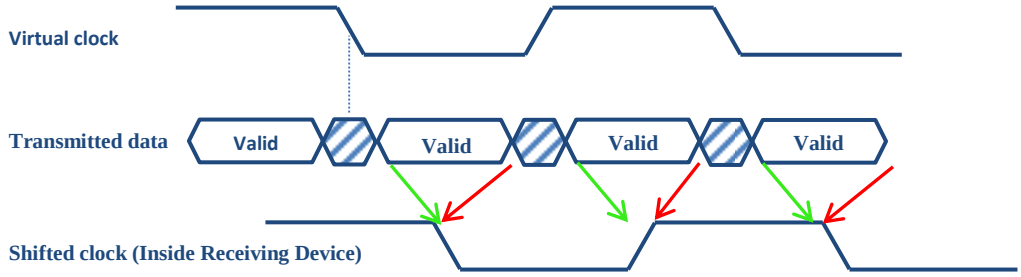
The data is transferred as rise-rise (launched on rising edge and captured on rising edge) or fall-fall (launched on rising edge and captured on rising edge).

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
```

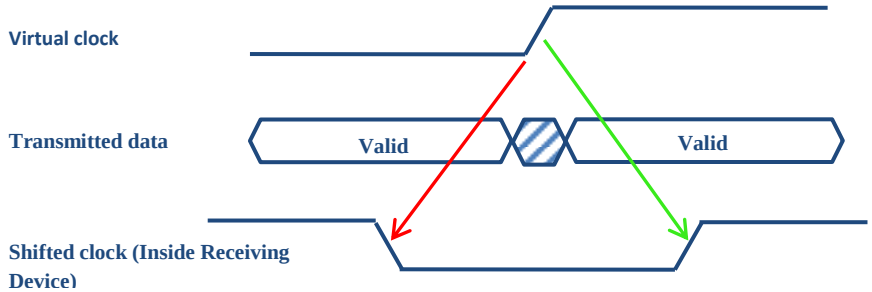
set_false_path -fall_from [get_clocks rx_clock_virt] -fall_to data_clock -hold
set_false_path -rise_from [get_clocks rx_clock_virt] -rise_to data_clock -hold
The Setup and hold analysis is shown below(Green is for setup, Red is for hold)



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). (Green is for setup, Red is for hold)



Clock Delay Capture mode:

In this mode the inherent clock insertion delay is used to delay the clock relative to the data to centre align the clock edge relative to the data valid window.

For the constraints below assume a 125MHz clock with a data to clock skew of -200ps to +400ps (data arrives from 200ps before the clock edge to 400ps after it).

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:

```
create_clock -name clkin -period 8 [get_ports clkin]
```

Input Constraints:

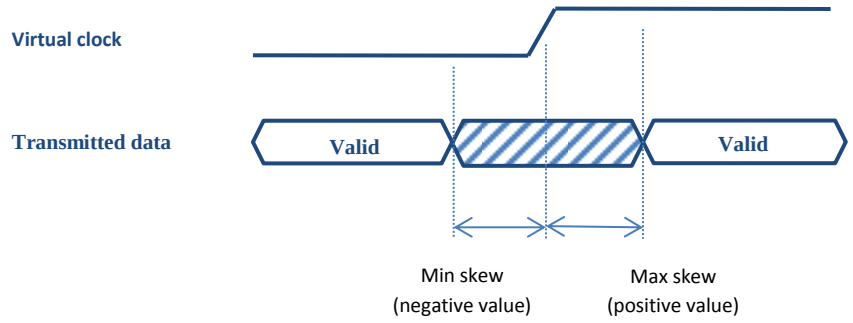
You can use a maximum skew specification to calculate input delay values. The maximum skew specification indicates the allowable time variation for individual bits of a data bus to arrive at the FPGA.

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

(A positive max figure states that the data arrives “after” the 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[*]}
```

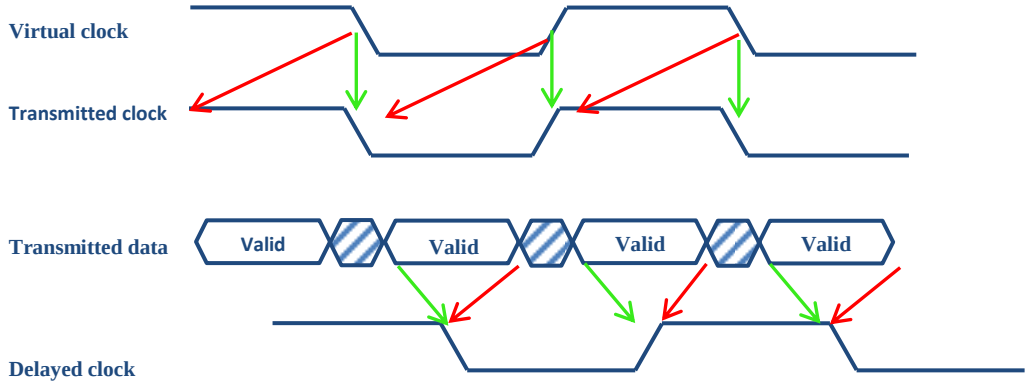
Timing Exceptions:

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

As the clock is being delayed internally, the clocks to which the constraints are applied are actually in phase:

(Green is for setup, Red is for hold)



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

There is no clock assignment for the delayed clock, the only clock assignments that exist are those for the virtual and transmitted clock.

As the delayed clock is only modelled as insertion delay, we need to create the setup and hold relationship for the “Virtual clock” to “Transmitted clock” as shown above (setup = 0, hold = -0.5* period).

We do this using the following multicycle constraints:

```
set_multicycle_path 0 -setup -end -from [get_clocks rx_clock_virt] -to [get_clocks clkin]
set_multicycle_path -1 -hold -end -from [get_clocks rx_clock_virt] -to [get_clocks clkin]
```

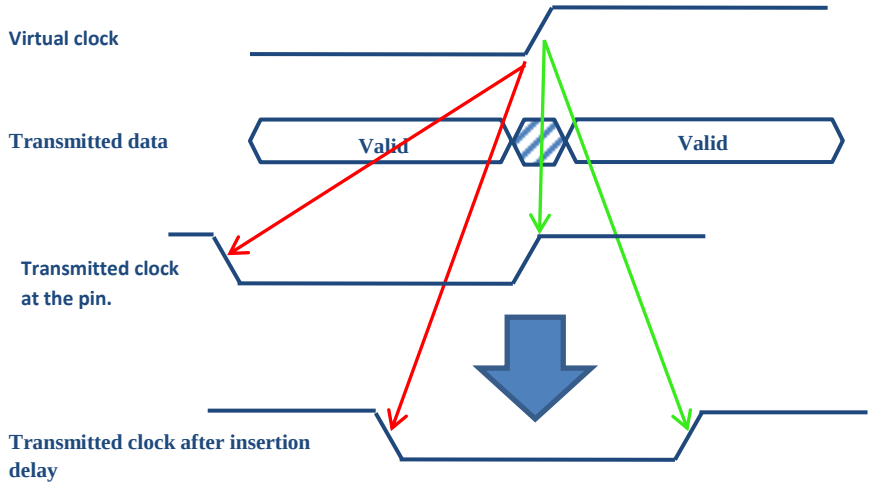
To prevent any unwanted relationships the following false path assignments are required:

```
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)

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

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



The setup and hold relationship from the virtual clock to transmitted clock (at the pin) is what you will see as the default relationships in TimeQuest(setup = 0, hold = -0.5* period). The relationships shown for the virtual clock to the delayed transmitted clock show how the default relationships change (accounted for as clock insertion delay in TimeQuest) as the clock is delayed.