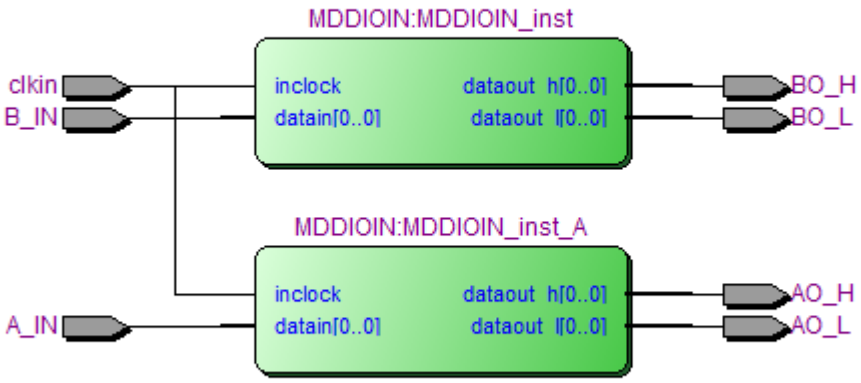
The two most common methods of implementing source synchronous input interfaces are:

- **Edge aligned interface where a PLL is used** in the destination device to shift the clock by 90Degees to centre align it with respect to the data.
- **Centre aligned interface** where the clock to data relationship is already correct to allow for capture in the destination device. A PLL may not be required for this case.

This example design is intended to show how to achieve timing closure for the case where only edge aligned data is possible and there is no possibility of introducing a pll into the destination device to centre align the clock with respect to the data.

The Design:

The design consists of a 2 bit source synchronous interface where each data bit is connected to an altddio_in input block:



The default constraints (for a 125MHz edge aligned interface) are as follows:

```
#Virtual Clock
create_clock -name clkin_virt        -period 8
#Clock Input
create_clock -name clkin  -period 8 [get_ports clkin]

#Edge aligned constraints (Data is aligned to clock edge +/- 0.8ns)
set_input_delay -max 0.8 -clock [get_clocks clkin_virt] [get_ports *_IN*]  -add_delay
set_input_delay -max 0.8 -clock [get_clocks clkin_virt] -clock_fall [get_ports *_IN*]  -add_delay
set_input_delay -min -0.8 -clock [get_clocks clkin_virt] [get_ports *_IN*] -add_delay
set_input_delay -min -0.8 -clock [get_clocks clkin_virt] -clock_fall [get_ports *_IN*] -add_delay

# Ensure rise-fall or fall-rise transitions for setup time analysis
set_false_path -setup -fall_from [get_clocks clkin_virt] -fall_to [get_clocks clkin]
set_false_path -setup -rise_from [get_clocks clkin_virt] -rise_to [get_clocks clkin]
# Ensure rise-rise or fall-fall transitions for hold time analysis
```

```
set_false_path -hold -fall_from [get_clocks clkin_virt] -rise_to [get_clocks clkin]
set_false_path -hold -rise_from [get_clocks clkin_virt] -fall_to [get_clocks clkin]
```

The Timing Problem:

The problem with such an interface is that if no PLL is available to either shift the clock or compensate for the clock insertion delay then there is a fixed clock insertion delay which can be in the magnitude of around 4ns (device dependent).

It is this clock insertion delay that causes timing closure problems.
The default setup and hold relationships (as defined in the constraints above) are as follows:

Hold requirement = 0; (edge aligned).
Setup requirement = 4ns; (0.5*clock period due to DDR implementation).

The ~4ns clock insertion delay means that your data valid window is effectively shifted by 4ns giving:

Hold requirement = ~4ns;
Setup requirement = ~8ns; ((0.5*clock period) + ~4ns)

Although the window (Setup requirement - Hold requirement) does not change, the fact that the window has shifted means that the data delay needs to be increased to achieve the hold requirement. This is usually done automatically by Quartus II using the IO cell delay chains.

The problem with forcing the data delay to be increased means that there is more variance in the timing model over PVT meaning that it may not be possible to close timing for both setup and hold over all timing corners.

Even for a relatively slow interface such as this one which is running at 125MHz, this clock insertion delay can result in hold violations such as the one shown below:

Report Timing

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-0.309	A_IN	MDDIOIN:MDDIOIN_inst_A|altddio_in:ALTDDIO_IN_c...t|ddio_in_j5i:auto_generated|ddio_ina[0]~DFFLO	clkin_virt	clkin	0.000	2.472	3.003

Path #1: Hold slack is -0.309 (VIOLATED)

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Data Arrival Path

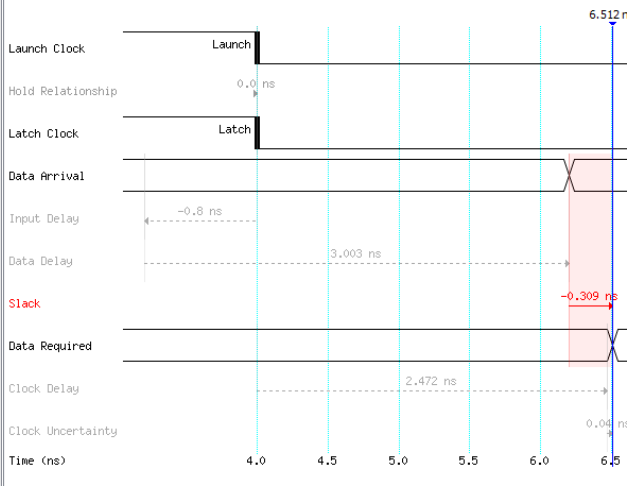
	Total	Incr	RF	Type	Fanout	Location	Element
1	4.000	4.000					launch edge time
2	◢ 4.000	0.000					clock path
1	4.000	0.000	F				clock network delay
3	3.200	-0.800	R	iExt	1	PIN_AF23	A_IN
4	◢ 6.203	3.003					data path
1	3.200	0.000	RR	IC	1	IOIBUF_X80_Y0_N52	A_IN~input|i
2	3.942	0.742	RR	CELL	1	IOIBUF_X80_Y0_N52	A_IN~input|o
3	3.942	0.000	RR	IC	2	DDIOINCELL_X80_Y0_N65	MDDIOIN_inst_A|ALTDDIO_IN_component|auto_generated|ddio_ina
4	6.203	2.261	RR	CELL	0	DDIOINCELL_X80_Y0_N65	MDDIOIN:MDDIOIN_inst_A|altddio_in:ALTDDIO_IN_c...t|ddio_in_j5i:a

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	4.000	4.000					latch edge time
2	◢ 6.472	2.472					clock path
1	4.000	0.000					source latency
2	4.000	0.000			1	PIN_Y15	clkin
3	4.000	0.000	FF	IC	1	IOIBUF_X64_Y0_N1	clkin~input|i
4	4.656	0.656	FF	CELL	2	IOIBUF_X64_Y0_N1	clkin~input|o
5	6.267	1.611	FF	IC	3	DDIOINCELL_X80_Y0_N65	MDDIOIN_inst_A|ALTDDIO_IN_component|auto_generated|ddio_ina
6	6.472	0.205	FR	CELL	0	DDIOINCELL_X80_Y0_N65	MDDIOIN:MDDIOIN_inst_A|altddio_in:ALTDDIO_IN_c...t|ddio_in_j5i:a
3	6.512	0.040					clock uncertainty
4	6.512	0.000		uTh	0	DDIOINCELL_X80_Y0_N65	MDDIOIN:MDDIOIN_inst_A|altddio_in:ALTDDIO_IN_c...t|ddio_in_j5i:a

Path #1: Hold slack is -0.309 (VIOLATED)

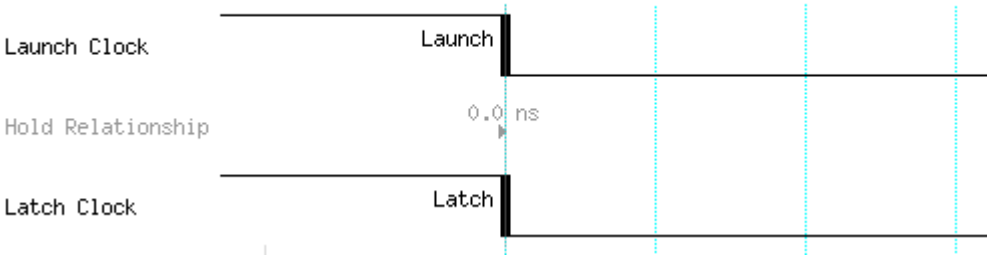
Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information



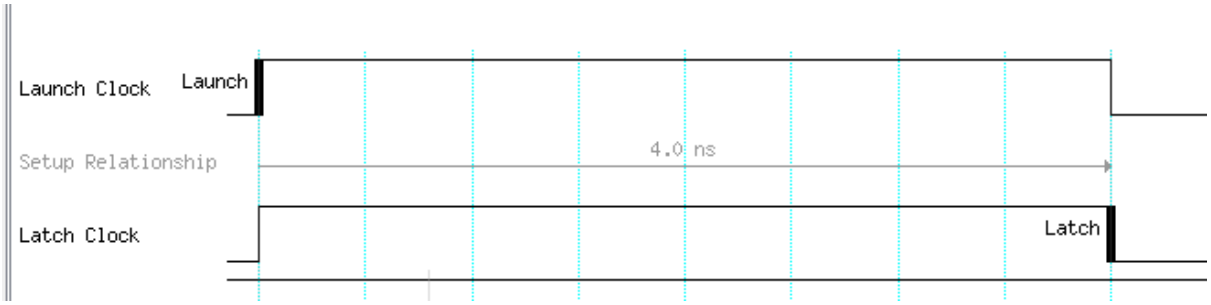
The Timing Solution:

The default constraints (as mentioned above) are:

Hold requirement = 0; (edge aligned).

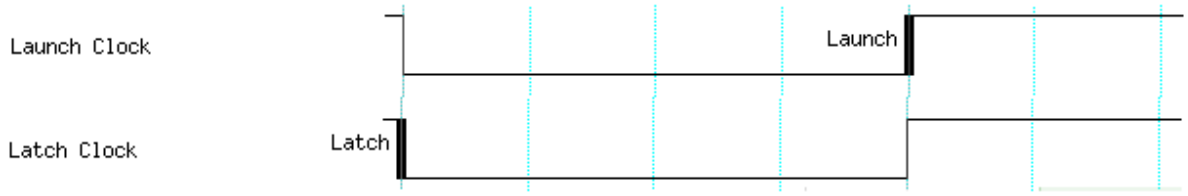


Setup requirement = 0.5*clock period; (DDR implementation).



In order to account for the large clock insertion delay and allow for a smaller data delay(to reduce variance over pvt) then the clock edges used for analysis need to be changed to give the following requirements:

Hold requirement = -0.5*clock period;



Setup requirement = 0;

The basic principal is to shift the edges used for setup and hold forward by one half clock cycle (So in hardware the capture takes place one half clock cycle earlier than it would normally have done so). So rather than the data being captured by the next clock edge which would usually be the case it is instead captured by the clock edge that it is aligned to ("Zero Cycle Of Latency" or "Same edge" transfer).

This relationship can be achieved via the following constraints:
(Note that the **Blue & Bold** font indicates assignments that have changed from the default ones above)

```
#Virtual Clock
create_clock -name clkin_virt        -period 8
#Clock Input
create_clock -name clkin  -period 8 [get_ports clkin]

#Edge aligned constraints (Data is aligned to clock edge +/- 0.8ns)
set_input_delay -max 0.8 -clock [get_clocks clkin_virt] [get_ports *_IN*]  -add_delay
set_input_delay -max 0.8 -clock [get_clocks clkin_virt] -clock_fall [get_ports *_IN*]  -add_delay
set_input_delay -min -0.8 -clock [get_clocks clkin_virt] [get_ports *_IN*] -add_delay
set_input_delay -min -0.8 -clock [get_clocks clkin_virt] -clock_fall [get_ports *_IN*] -add_delay


#Shift back the clock edge used for setup
set_multicycle_path -setup -from [get_clocks clkin_virt] -to [get_clocks clkin] 0
#Shift forward the clock edge used for hold
set_multicycle_path -hold -from [get_clocks clkin_virt] -to [get_clocks clkin] -1
```

```
# Ensure rise-rise or fall-fall transitions for setup time analysis
set_false_path -setup -fall_from [get_clocks clkin_virt] -rise_to [get_clocks clkin]
set_false_path -setup -rise_from [get_clocks clkin_virt] -fall_to [get_clocks clkin]
# Ensure rise-fall or fall-rise transitions for hold time analysis
set_false_path -hold -fall_from [get_clocks clkin_virt] -fall_to [get_clocks clkin]
set_false_path -hold -rise_from [get_clocks clkin_virt] -rise_to [get_clocks clkin]
```

After adding in the ~4ns clock insertion delay in to the analysis your requirements become:

Hold requirement = ~0ns; (-4ns hold requirement + ~4ns clock insertion delay).
Setup requirement = ~4ns; (0ns setup requirement + ~4ns clock insertion delay).

By shifting back the setup and hold edges by 0.5 clock periods we have basically accounted for the clock insertion delay.
As the overall hold requirement (hold requirement + clock insertion delay) is now in the region of 0ns then no additional delay needs to be added to the datapath meaning that the variance over pvt is now significantly reduced and timing can be met.

Limitations:

There are two main limitations to this solution:

1) As this is a DDR interface and you are changing the edge used to capture the data by 1/2clock cycles then the data will appear inverted on the output of the altddio_in module. This needs to be accounted for in your design logic.
2) This solution is heavily dependent on the device (and speedgrade) used along with the target frequency of the interface. It will need to be tried on a case by case basis to verify timing closure.