Achieving Timing Closure in Basic (PMA Direct) Functional Mode

AN-580-4.0 Application Note

This application note describes the method to achieve timing closure for designs that use transceivers in Basic (PMA Direct) mode in Altera's Stratix® IV GX or Stratix IV GT FPGAs. It also describes best practices for the Quartus® II software version 9.1 SP1 and earlier and best practices for the Quartus II software version 9.1 SP2 and later.

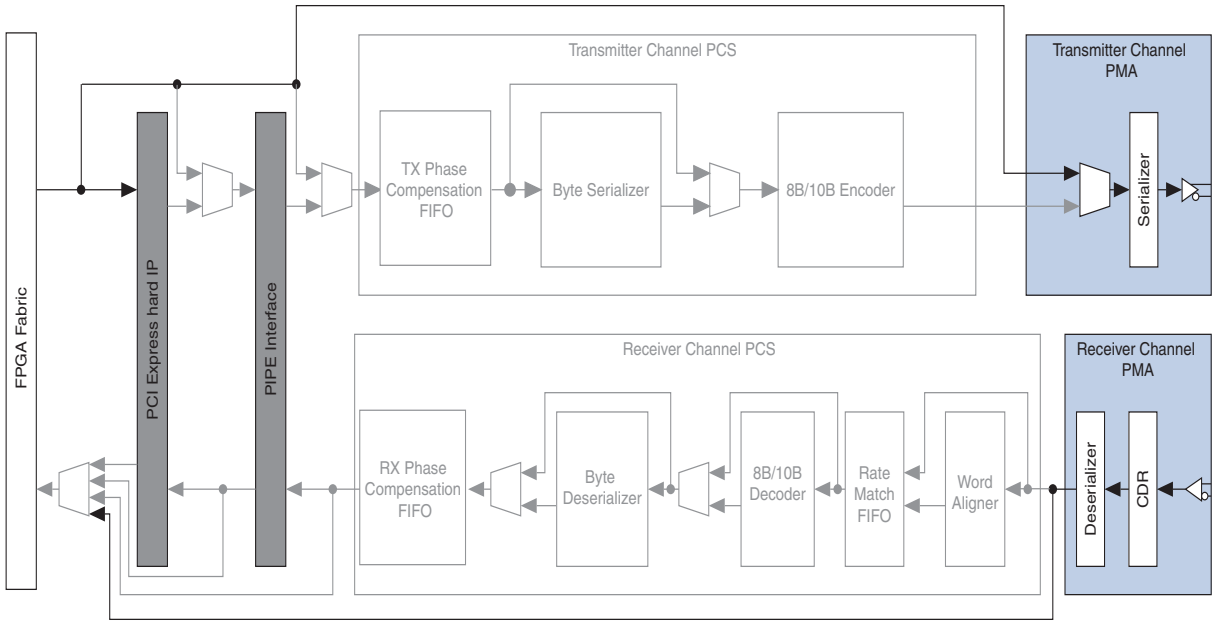
This application note describes techniques for resolving timing violations for paths between the transceiver and the FPGA core only. You must follow general best practices for timing closure in FPGA designs to resolve any timing violations that exist in the FPGA core.

If you are using the Quartus II software version 9.1 SP1 or earlier, Altera strongly recommends upgrading to the latest software version. Achieving timing closure is greatly improved in the software versions 9.1 SP2 and later.

Introduction

Transceiver channels configured in Basic (PMA Direct) functional mode only use the physical medium attachment (PMA) blocks of the transceiver channels. The physical coding sub-layer (PCS) blocks of all channels are bypassed, as shown in Figure 1.

Figure 1. Basic (PMA Direct) Functional Mode



101 Innovation Drive
San Jose, CA 95134
www.altera.com

The interface between the transceivers and the FPGA core introduces significant delay for the clocks that are forwarded from the transceivers to the user logic in the FPGA core. A phase-compensation FIFO buffer in the transceiver PCS block compensates for the phase difference (due to delays) between the transceiver clocks used internally and the transceiver clocks routed to and from the FPGA core. When transceivers are used in Basic (PMA Direct) functional mode, the FIFO is bypassed because the entire PCS block is bypassed. The result is the timing requirement is not easily met.

If you are having problems closing timing on the transmit side, start by making design changes described in "Manage Data Valid Windows" on page 3 and "Manage Clock Resource Allocation" on page 9. Based on the design changes you have made, you may need to make additional changes based on the information in "Manage Insertion Delay" on page 4. After these changes, recompile your design with the software settings described in "Software Settings" on page 11 and analyze your results with the information described in "Analyzing the Design" on page 12. Continue to use the techniques in "Manage Insertion Delay" on page 4 and "Manage Long Data Paths" on page 7, as necessary.

If is often much easier to close timing on the receive side. If you are having problems on the receive side, refer to "Meeting Timing on the Receive Side" on page 8.

For more information about Basic (PMA Direct) functional mode, refer to the *Transceiver Architecture in Stratix IV Devices* chapter in volume 2 of the *Stratix IV Device Handbook*.

Meeting Timing on the Transmit Side

Closing timing on the transmit side is challenging for the following reasons:

- different data valid windows at the transceiver inputs for high and low data rates
- large insertion delays on certain clock resources used for the transceiver transmit clocks
- long distances between the user logic and the transceivers when you use most or all of the transceivers

To address these timing challenges, do the following:

- accommodate different data valid windows with clock polarity changes
- manage insertion delays with additional register stages or FIFOs
- assign specific, low insertion delay clock resources to the channel clocks
- control placement with LogicLock regions to assist the Fitter in timing closure

These techniques are described in the following sections.

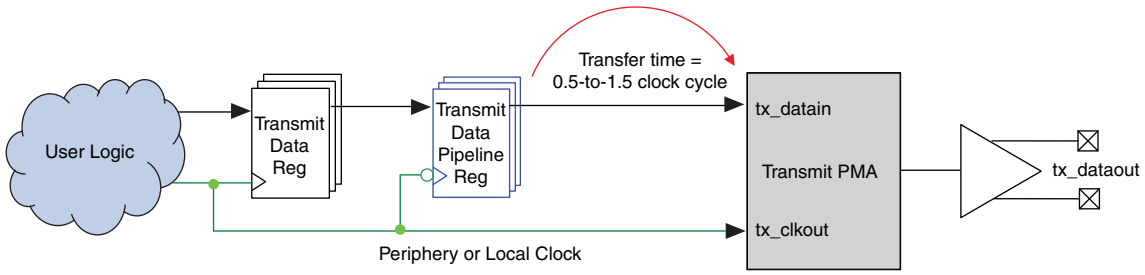
Manage Data Valid Windows

Data must arrive at the inputs of the transmit transceiver within a certain time window. The data valid window varies according to the data rate of the interface. The data valid window for low data rates does not overlap with the data valid window for high data rates. Therefore, different techniques are required to transfer data into the transmit transceiver for high and low data rates.

High Data Rates

At data rates of 5G and above, you must shift the data arrival time at the transmit transceiver by half a clock period to meet its timing requirement. Use a negative edge-clocked register to drive data into the transmitter, as shown in Figure 2, and add a multicycle SDC constraint.

Figure 2. Negative Edge-Triggered Pipeline Registers to Meet the Timing Requirement



The following Synopsys Design Constraint (SDC) is required for proper timing analysis with the extra half clock cycle:

```
set_multicycle_path -setup -end -from [get_registers <negative edge triggered register name>] 2
```

The negative edge-clocked registers for each channel must be driven by the individual channel clocks, regardless of whether you use bonded or non-bonded mode. Do not use one common clock signal to drive all the channels' negative edge-triggered registers.

Even with the extra half clock cycle, you must also reduce insertion delay and assign clock resources to close timing.

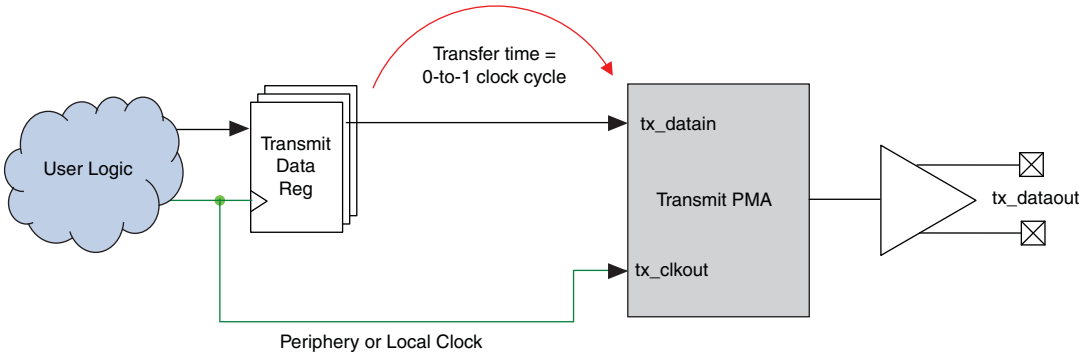
For more information about reducing insertion delay, refer to "Manage Insertion Delay" on page 4.

For more information about assigning specific clock resources, refer to "Manage Clock Resource Allocation" on page 9.

Low Data Rates

At low data rates, the data valid window is properly aligned with positive edge-clocking. Use a positive edge-clocked register to drive data into the transmitter, as shown in Figure 3.

Figure 3. Single Channel Configured in Basic (PMA Direct) Functional Mode with a Positive Edge-Clocked Register



At low data rates, regional clocks often have low enough insertion delay to allow timing closure. If you cannot close timing with regional clocks, you may also need to reduce insertion delay and assign clock resources to close timing.

For more information about reducing insertion delay, refer to "Manage Insertion Delay".

For more information about assigning specific clock resources, refer to "Manage Clock Resource Allocation" on page 9.

Manage Insertion Delay

There are various techniques to mitigate clock insertion delay on the clock signals for the transmit side of the transceiver blocks. For example, depending on the data rate of your interface, you can use:

- data transfers from clock resources with high insertion delay to clock resources with low insertion delay with multiple register stages
- data transfers between clock resources with a FIFO

Multiple Register Stages

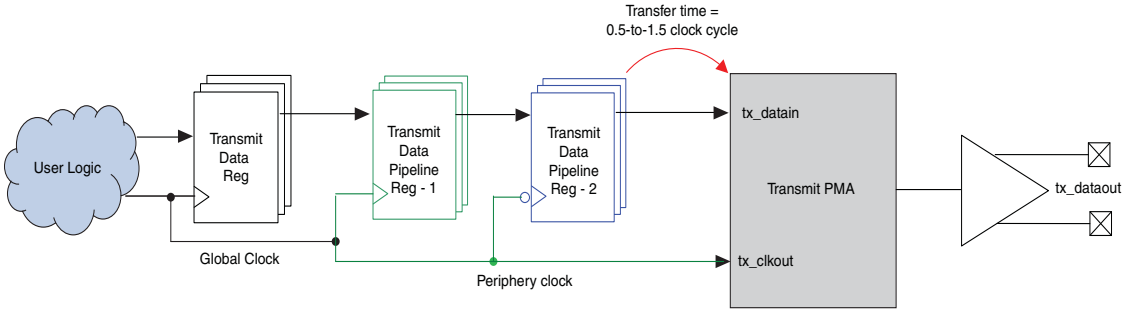
Different data rates require a different number of additional register stages.

If you drive data into the transmit transceiver with a register that uses a clock resource with low insertion delay, you may need to add register stages to manage the clock insertion delay differences. If the register driving data into the transmit transceiver uses different clock resources, or different clock polarities compared with your transmit user logic, you must add additional register stages to accommodate the clock insertion delay differences.

Add one level of registers for each clock resource change and clock polarity change, between the register driving the transmit transceiver and your transmit user logic.

For example, if you added a negative edge-clocked register and used a periphery clock to drive it, and your transmit user logic is positive edge-clocked on a global clock, you must add two levels of registers to switch clock resources and clock polarities, as shown in Figure 4.

Figure 4. Negative Edge-Triggered Pipeline Registers to Meet the Timing Requirement at Higher Data Rates



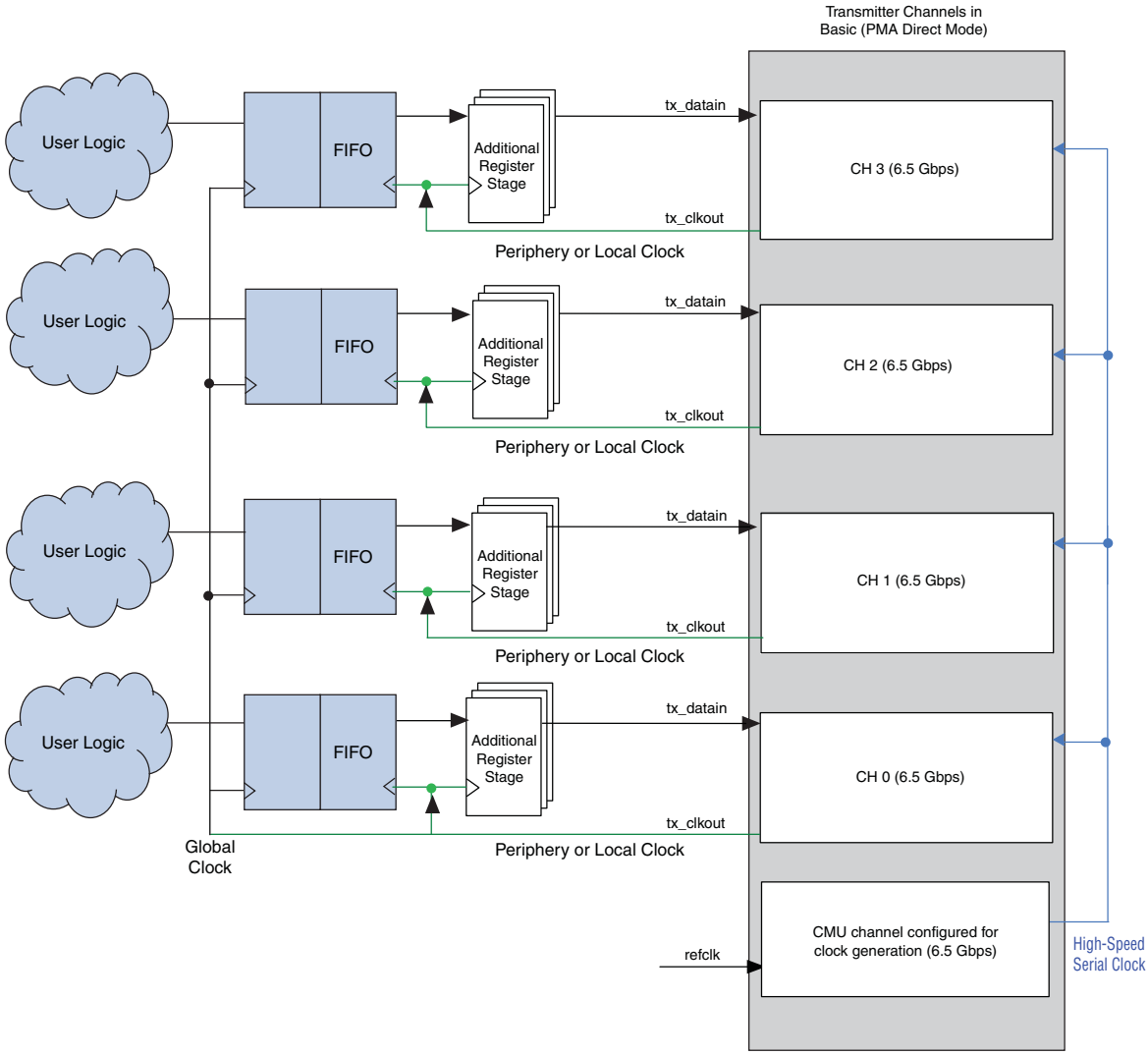
One register transfer accommodates the clock insertion delay difference between the global and the periphery clock and one register transfer accommodates the switch from positive to negative clock polarity.

FIFOs

You can also use a FIFO to manage the clock insertion delay differences. A FIFO can replace multiple registers used for clock resource crossing. Even when you use a FIFO, you must typically add registers between the user logic and the FIFO, and the FIFO and the transceiver. The reason for this register usage is described in "Manage Long Data Paths" on page 7.

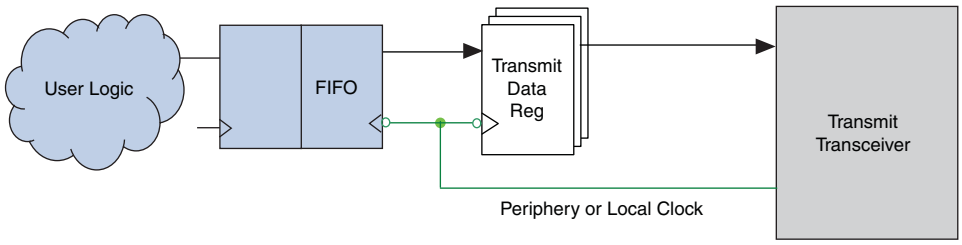
When you use a FIFO, it is typical to use one clock to drive all the write ports for the transmit side, then use individual channel clocks to drive the read ports interfacing with the transceivers, as shown in Figure 5.

Figure 5. Multiple Channels Using FIFOs



If you use a negative edge-clocked register to feed the transmit transceiver, and you use a FIFO, drive the FIFO's read clock port with a negative edge-clock, as shown in Figure 6.

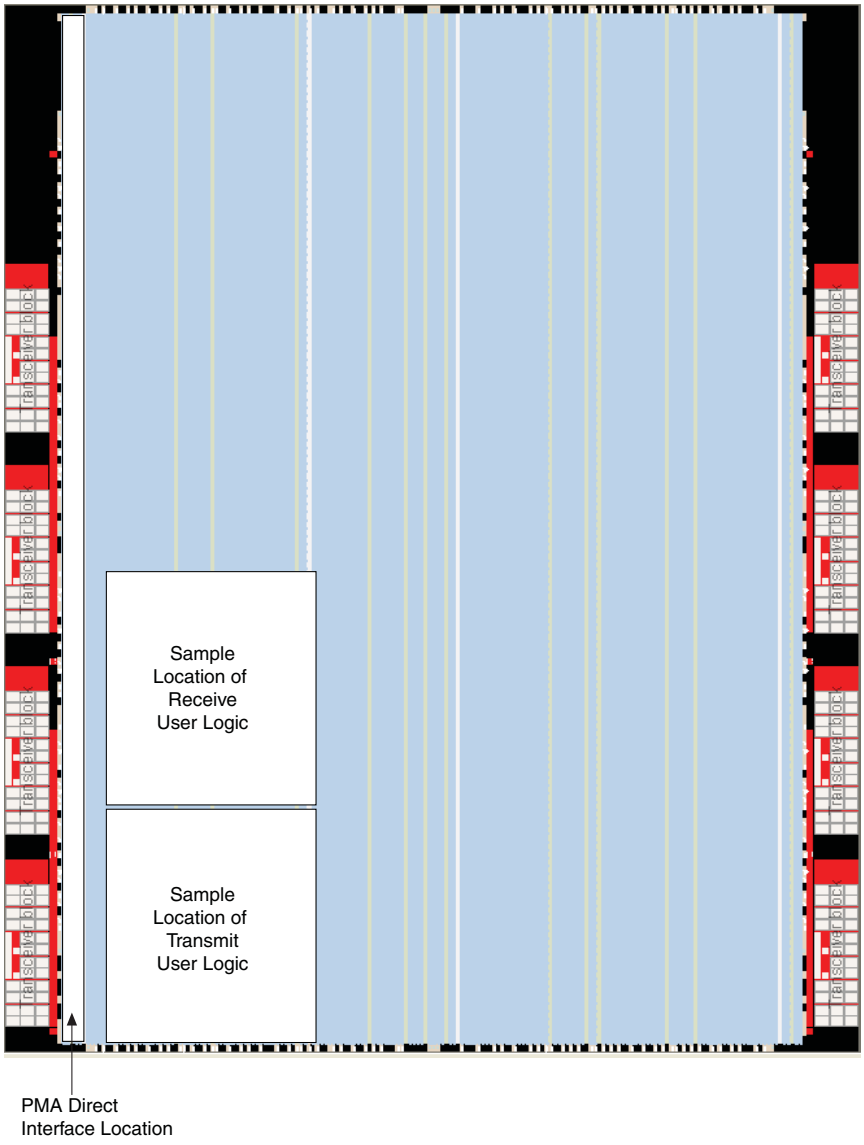
Figure 6. Transmit Transceiver with a Negative Edge-Clocked Register and a FIFO



Manage Long Data Paths

At high data rates, short clock periods make timing closure difficult. At high data rates and channel counts, many transceivers are too far away from the user logic to transfer data in a single clock cycle. In addition, with high channel counts, the channels are spread along one side of the device, while the transmit and receive user logic is typically clustered in a fraction of the chip, as shown in Figure 7.

Figure 7. Floorplan for a Stratix IV GX Device



At high data rates and channel counts, having additional registers between the user logic and the transceiver reduces the physical distance the data signals must travel during each clock cycle. The number of registers you need varies depending on the data rate, channel count, and congestion near the transceivers. At high data rates, with a small amount of logic placed close to the transceivers, start with one additional register stage. In full designs, start with two additional register stages. Using more registers eases the placement and routing pressure near the transceivers when the device is full, or when the area near the transceivers is full, and allows the Fitter more flexibility.

Meeting Timing on the Receive Side

Closing timing on the receive side is typically easier than on the transmit side because the additional clock insertion delay relaxes the setup requirement on the transfers from the receive transceiver block to user logic.

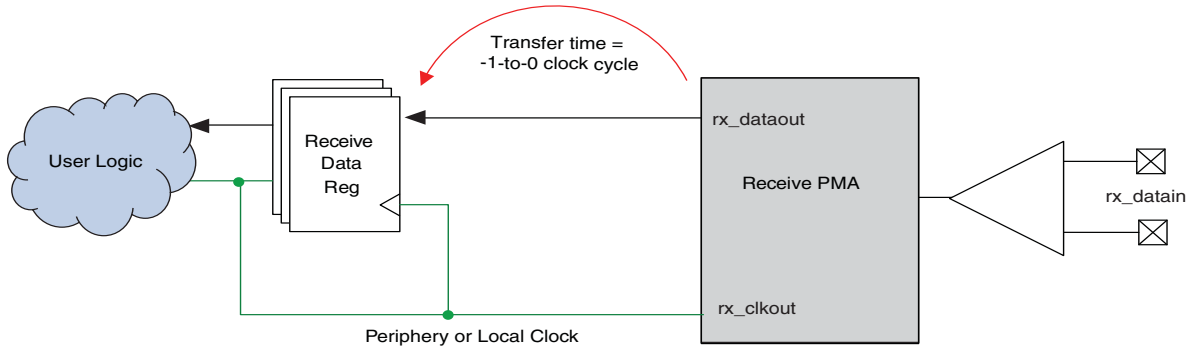
Manage Insertion Delay

Every channel on the receive side uses its own clock. The receive-side clock path is slower than the receive-side data path. If you observe significant hold violations on the paths from the receive transceiver block, use the following multicycle SDC constraint:

```
set_multicycle_path -setup -end -from [get_registers *recoverdataout*] 0
```

Figure 8 shows how the multicycle SDC constraint changes the transfer time to -1-to-0 clock cycles to account for the delay on the data path on the receive side.

Figure 8. Timing Constraint Changes due to Delay on the Data Path (Receive Side)



Typically, you do not need to add register stages to manage clock insertion delay differences as you do for the transmit side. However, additional registers are sometimes required to manage long data paths, just as with the transmit side. For more recommendations when using long data paths, refer to "Manage Long Data Paths" on page 7.

Manage Clock Resource Allocation

In addition to using additional register stages, you must also make global signal assignments to control the resources used by the channel clocks. On the transmit side, these clocks must be implemented with clock resources that have low insertion delay. Periphery clocks have the lowest insertion delay of all global signals. However, if you need even lower clock insertion delay to close timing, switch from periphery clocks to use local routing. For more information, refer to "Using Local Routing" on page 10. The exact clock resource type is less critical on the receive side, but at high data rates, periphery clocks or local routing may be required.

Make a separate clock resource assignment for each channel clock. For example, a 20-channel design would have 20 separate transmit clock resource assignments, one for each channel. If you hand-edit the project settings file (**.qsf**), make the assignments according to the following pattern:

```
set_instance_assignment -name GLOBAL_SIGNAL "Periphery clock" -to <clock name>
```

To make the assignment to *<clock name>* using the Assignment Editor, follow these steps:

1. Select **Global Signal** for the **Assignment Name** option.
2. Select **Periphery clock** for the **Value** option.

Clock Naming

When you create resource assignments for the transmit and receive clocks, you must use post-synthesis names that may not be the same as the names for the clocks in your source code. For the transmit clocks, the clock names have the following form:

```
<hierarchical path of the ALTGX megafunction>|wire_transmit_pma<n>_clockout.
```

For the receive clocks, the clock names have the following form:

```
<hierarchical path of the ALTGX megafunction>|wire_receive_pma<n>_clockout.
```

An example of a clock name is:

```
serdes_if:u0|hssi:u1|gxb:gxb_alt4gxb_71h8_component|wire_transmit_pma0_clockout.
```

The numeric value *<n>* in the clock name has a range that corresponds to the number of channels in your ALTGX megafunction variation. The value varies from 0 to one less than the number of channels in your megafunction variation.

For example, in a 20-channel ALTGX megafunction variation, there are 20 transmit clock signals, ending in the pattern from `wire_transmit_pma0_clockout` to `wire_transmit_pma19_clockout`. A design with a single-channel ALTGX megafunction variation instantiated multiple times has multiple transmit clock signals names ending in `wire_transmit_pma0_clockout`, differentiated by the instance names above it in the hierarchy.

Using Local Routing

To make a channel clock use local routing, set a value of **OFF** in its global signal assignment.

When you use local routing for a transceiver channel clock, do not use the local routing to drive more logic than any register stages you added. Use a global signal to drive any user logic.

When you use a FIFO, use the locally routed clock to drive the side of the FIFO connected to the transceiver and any register stages between the FIFO and the transceiver block. Do not use the locally routed clock to drive more logic than this. Use a single global clock to drive the user logic side of all the channels' FIFOs. If you use a FIFO for managing clock insertion delay on any channel, it is easiest to use FIFOs for managing clock insertion delay on all channels.

Clock Resources for User Logic

In a design where you use the clock from one transmit channel to also drive user logic, the clock driving the user logic must use a global signal. Typically, it would be on a global clock or regional clock. If the Quartus II software does not automatically promote the clock for user logic to an appropriate resource, add an additional global signal assignment to perform the promotion.

For this example, assume that you have made periphery clock assignments to all the transmit channel clocks, and the clock <*altgx hierarchy*>|wire_transmit_pma0_clockout is also used to drive user logic, and it needs to use a global clock. Add the following assignment to your **.qsf** to promote <*altgx hierarchy*>|wire_transmit_pma0_clockout to a global clock for the user logic.

```
set_instance_assignment -name GLOBAL_SIGNAL "Global clock" -to
<altgx hierarchy>|wire_transmit_pma0_clockout
```

If you add a global signal assignment for the clock from one channel that drives user logic, and you made global signal assignments as described in the example, you must verify that the Quartus II software uses the appropriate clock resource to drive any registers you added to close timing. To understand how to verify resource use, refer to "Clock Resource Analysis" on page 13.

When one clock has multiple types of global signal assignments made to it, the Quartus II software may not choose the clock resource you want to drive the particular registers. You may need to add point-to-point global signal assignments to disambiguate which global signal type must be used to clock particular registers. In the example just described, with a global and a periphery clock assignment made to <*altgx_hierarchy*>|wire_transmit_pma0_clockout, the following assignment forces the named registers to be driven with the periphery clock:

```
set_instance_assignment -name GLOBAL_SIGNAL "Periphery clock" -from <altgx
hierarchy>|wire_transmit_pma0_clockout -to <additional register stages>
```

Clock Resources for Individual Registers

If only a few registers in a few channels fail timing due to insertion delay, assign those individual registers to clock resources with lower insertion delay, if they are not already driven with local routing, which has the lowest insertion delay.

Use a point-to-point global signal assignment from the channel clock name to a single register. If the channel clock uses a regional clock, use a value of **Periphery Clock** for the point-to-point assignment. If the channel clock uses a periphery clock, use a value of **OFF** to switch to local routing. The following assignment is appropriate to switch a single register to local routing in channel zero which uses a periphery clock:

```
set_instance_assignment -name GLOBAL_SIGNAL "Off" -from <altgx
hierarchy>|wire_transmit_pma0_clockout -to <register name>
```

If the few registers failing timing are already driven with local routing, add another register stage or add logic array block (LAB) location assignments to the failing registers.

Control Placement

If you add register stages or a FIFO to manage clock insertion delay or long data paths, you may also need to create one or two LogicLock regions to contain the additional logic. Create these LogicLock regions only after you assign clock resources for each channel and verify that they are implemented correctly. LogicLock regions can help if registers that fail timing are placed more than five or six columns away from the edge of the device.

- If you add register stages, create a LogicLock region for them that is the full height of the chip and five columns wide adjacent to the edge of the chip with the transceivers.
- If you use FIFOs between the transceiver and the core logic, create a LogicLock region for them that is the full height of the chip and three columns wide, and overlap it with the first column of the embedded memory next to the transceivers.

Even with LogicLock regions, you may have to make more specific location assignments to certain registers if a small number of registers fail timing and the registers are placed poorly.

Software Settings

To achieve the best timing closure results, use the most recent version of the Quartus II software, or at least version 9.1 SP2.

To improve timing closure, in the **Fitter Settings** page of the **Settings** dialog box, set the **Optimize Hold Timing** option to **All Paths** and select the **Optimize Multicorner Timing** option in the **Timing-driven compilation** section. Also select the **Standard Fit (highest effort)** option in the **Fitter effort** section.

For more information, refer to the *Fitter Settings Page (Settings Dialog Box)* section in the Quartus II Help.

If you are very close to meeting timing (within 50 ps), and you have used all the techniques already described, increase the effort for the placer and router. In the **Fitter Settings** page of the **Settings** dialog box, click **More Settings**, then set the parameters on the **More Fitter Settings** dialog box. Set **Placement Effort Multiplier** to **4.0**. Set **Router Effort Multiplier** to **4.0**.

For more information, refer to the *More Fitter Settings Dialog Box* section in the Quartus II Help.

Analyzing the Design

The following sections describe analyzing your design to improve timing closure.

Deciding What to Change

When you perform timing analysis and review the failing paths in your PMA Direct interface, use the following information to decide what approach to take to close timing. If the variation between the setup and hold slacks for a failing path is too large, it will be impossible to close timing without changing the clock resources.

Perform setup and hold timing analysis for one failing path at all operating conditions and calculate slack variation using the formula shown in Equation 1.

Equation 1.

$$\frac{\min(\text{setup slacks at all operating conditions}) + \min(\text{hold slacks at all operating conditions})}{2}$$

If the result is negative, it will be impossible to close timing on the failing path without changing the clocking resources used by the registers in the failing path. If the result is positive, but less than approximately 300 ps, change the clocks to use resources with lower insertion delay, if possible. If it is not possible to use clock signals with lower insertion delay, the placement of the registers in the failing paths is critical. LogicLock regions may not provide enough control and LAB-level assignments to each failing register may be required.

If the result is greater than approximately 300 ps, the clock resources are appropriate for the path, but the placement is not optimal. Attempt to shift the registers closer together or further apart with LogicLock regions or add an SDC constraint to over-constrain the failing paths by up to 10%.

If paths fail setup time, use a `set_max_delay` constraint between the failing register stages with a value of 90% of the setup relationship. If paths fail hold time, use a `set_min_delay` constraint between the failing register stages with a value of the current hold relationship plus 10% of the clock period. The setup and hold relationship values are shown on the **Waveform** tab of the Report Timing results when you perform setup and hold analysis.

If you over-constrain paths to improve placement, apply the extra constraints only during fitting. Remember to remove the extra constraints before performing timing analysis. Using code similar to the following example applies the extra constraints only during fitting:

```
if { ![string equal "quartus_sta" $::TimeQuestInfo(nameofexecutable)] } {
   # add extra constraints here
}
```

Analyzing Timing

To verify that the PMA Direct interface meets the timing requirements, review the timing reports from the TimeQuest Timing Analyzer. You must review the timing analysis results for all operating conditions to ensure that the interface meets the timing requirements for each operating condition. **Report All Summaries** provides a high-level summary of all the timing violations and slack.

Use the `report_timing` command to review the timing analysis of the PMA Direct transceiver interface. Specify a pattern to restrict timing analysis to only matching register names in the transceiver block. The destination register in the transmitter transceiver ends in the name `~OBSERVABLEOUT`. The source register in the receiver transceiver ends in the name `recoverdataout`.

Use the following commands to report the worst 100 setup and hold timing paths to the transmit transceiver interface:

```
report_timing -setup -to [get_registers *~OBSERVABLEOUT*] -npaths 100 -panel_name {PMA
Direct TX setup}

report_timing -hold -to [get_registers *~OBSERVABLEOUT*] -npaths 100 -panel_name {PMA
Direct TX hold}
```

Use the following commands to report the worst 100 setup and hold timing paths from the receive transceiver interface:

```
report_timing -setup -from [get_registers *recoverdataout*] -npaths 100 -panel_name {PMA
Direct RX setup}

report_timing -hold -from [get_registers *recoverdataout*] -npaths 100 -panel_name {PMA
Direct RX hold}
```

Use similar commands, with the names of the additional register stages you added, to verify timing for the additional register stages.

Clock Resource Analysis

Use the compilation report to verify that your design was implemented using the clock resources you assigned. Review the information in the **Global & Other Fast Signals** panel in the **Resource** section of the Fitter report (Figure 9). The report shows the global signals used for each clock in the design. Verify that the clocks for the PMA Direct channels use the global signals you assigned them to use. Remember that switching a clock to use local routing causes it to not appear in the report.

Figure 9. Fitter Report

Global & Other Fast Signals

	Name	Location	Fan-Out	Fan-Out L	Global Resource Used	Global Line Name
1	altera_interlaken_vrqmhm3t:interlaken_...s4_alt4gxb_etb9_component|tx_clkout[0]	TXPCS_X119_Y50_N140	23086	18502	Global Clock	GCLK11
2	altera_interlaken_vrqmhm3t:interlaken...c:auto_generated|l2_w0_n0_mux_dataout	LABCELL_X118_Y46_N16	2	0	Global Clock	GCLK9
3	altera_interlaken_vrqmhm3t:interlaken...c:auto_generated|l2_w1_n0_mux_dataout	MLABCELL_X111_Y36_N16	2	0	Global Clock	GCLK1
4	altera_interlaken_vrqmhm3t:interlaken...c:auto_generated|l2_w2_n0_mux_dataout	LABCELL_X118_Y46_N8	2	0	Global Clock	GCLK13
5	altera_interlaken_vrqmhm3t:interlaken...c:auto_generated|l2_w3_n0_mux_dataout	LABCELL_X118_Y46_N30	2	0	Global Clock	GCLK6
6	altera_interlaken_vrqmhm3t:interlaken_...tb9_component|wire_receive_pcs0_clkout	RXPCS_X119_Y63_N139	3852	124	Global Clock	GCLK4
7	altera_interlaken_vrqmhm3t:interlaken...c:auto_generated|l2_w0_n0_mux_dataout	MLABCELL_X99_Y73_N24	2	0	Global Clock	GCLK14
8	altera_interlaken_vrqmhm3t:interlaken...c:auto_generated|l2_w1_n0_mux_dataout	MLABCELL_X99_Y73_N6	2	0	Global Clock	GCLK12
9	altera_interlaken_vrqmhm3t:interlaken...c:auto_generated|l2_w2_n0_mux_dataout	MLABCELL_X99_Y73_N22	2	0	Global Clock	GCLK7
10	altera_interlaken_vrqmhm3t:interlaken...c:auto_generated|l2_w3_n0_mux_dataout	MLABCELL_X99_Y73_N0	2	0	Global Clock	GCLK8
11	altera_internal_jtag~TCKUTAP	JTAG_X0_Y95_N125	1177	17	Global Clock	GCLK15
12	clk_156_25	PIN_AF34	1	0	Global Clock	GCLK2
13	clkin_50	PIN_AC34	892	9	Global Clock	GCLK0
14	rx_mac_r_reset	MLABCELL_X58_Y49_N30	7297	0	Global Clock	GCLK5
15	tx_lane_r_reset~0	MLABCELL_X58_Y48_N38	5374	0	Global Clock	GCLK10
16	tx_mac_r_reset	MLABCELL_X58_Y48_N18	4783	0	Global Clock	GCLK3

When you use `report_timing` to perform timing analysis on the PMA Direct interface, use the `-show_routing` option to show the type of clock resource used for any register in the design. Clock resource information is included under "data arrival path" in the Data Path section of the report, as shown in Figure 10.

Figure 10. Data Arrival Path

Path #1: Setup slack is 0.096

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

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	1.535	1.535					launch edge time
2	3.978	2.443					clock path
3	1.535	0.000					source latency
4	1.535	0.000			2	TXPMA_X185_Y42_N138	serdes_inst1|serdes_6500
5	1.833	0.298	FF	CELL	1	TXPMA_X185_Y42_N138	serdes_inst1|serdes_6500
6	2.415	0.582		RE	302	TXPMA_X185_Y42_N138	TGX_HSSI_TX_PMA
7	2.577	0.162		RE	1	INTERQUAD_TXRX_PMATX_OUT_X185_Y33_N135_I3	IQPMATXOUT
8	2.577	0.000		RE	1	INTERQUAD_TXRX_PCLK_CTRL_X185_Y33_N0_I14	IQPCLKCTRL
9	2.577	0.000		RE	1	CLKBUF_IN_X185_Y59_N127_I0	CLKBUF_IN
10	2.577	0.000	FF	IC	2	CLKCTRL_X185_Y59_N127	serdes_inst1|serdes_6500
11	2.577	0.000	FF	CELL	75	CLKCTRL_X185_Y59_N127	serdes_inst1|serdes_6500
12	2.577	0.000		RE	1	CLKCTRL_X185_Y59_N127	TITAN_CLKBUF
13	2.629	0.052		RE	1	CLKBUF_OUT_X185_Y59_N127_I0	CLKBUF_OUT
14	2.795	0.166		RE	1	PERIPHERY_CLOCK_X162_Y48_N0_I0	PERIPHERY_CLOCK
15	3.163	0.368		RE	3	SPINE_CLOCK_X162_Y33_N0_I0	SPINE_CLOCK
16	3.275	0.112		RE	1	SCLK_TO_ROWCLK_BUF_X162_Y59_N0_I0	SCLK_TO_ROWCLK_BUF
17	3.300	0.025		RE	1	SCLK_TO_ROWCLK_BUF_X162_Y59_N0_I2	SCLK_TO_ROWCLK_BUF
18	3.475	0.175		RE	4	LAB_CLK_X163_Y59_N0_I0	LAB_CLK

In Figure 10 the entry in the Element column on line 14 is `PERIPHERY_CLOCK`, which indicates that the clock signal uses a periphery clock to drive the register.

Global signals are indicated by elements named `GLOBAL_CLOCK`, `QUADRANT_CLOCK`, and `PERIPHERY_CLOCK`. The element named `GLOBAL_CLOCK` indicates that the clock driving the register used a global clock. The element named `QUADRANT_CLOCK` indicates that the clock driving the register used a regional clock. The element named `PERIPHERY_CLOCK` indicates that the clock driving the register used a periphery clock. If the register is clocked through local routing, without using a global signal, no type of global clock element is named in the Data Path section.

For more information about Timing Report Generation, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Combining High-Speed Serial Interfaces with Other Interfaces

When you use many or all of the transceivers on a device, and use independent global signals for each channel clock as Altera recommends, it uses a significant amount of the total clock resources available on the device. If you combine HSSI interfaces with other clock-intensive interfaces, such as external memory or LVDS, you may run out of clock resources and be unable to fit the design.

In this case, you can usually fit the design by switching some transceiver channel transmit clocks from global signals to local routing. For the assignment that switches a clock to use local routing, refer to "Manage Clock Resource Allocation" on page 9.

Timing Closure for the Quartus II Software Version 9.1 SP1 and Earlier

This section describes the best practices for achieving timing closure if you are using the Quartus II software version 9.1 SP1 or earlier.

☞ If you are using the Quartus II software version 9.1 SP1 and earlier, Altera strongly recommends upgrading to the latest software version. Achieving timing closure is greatly improved in the software versions after 9.1 SP1.

There is a timing closure script that adds assignments to the design to meet timing. Download the script package at AN580_scripts.zip. It is also included in the Quartus II installation directory <*installation*>**/quartus/common/tcl/apps/pmaff**.

The script adds the following assignments to the project:

- Register placement—The script looks for the registers in a pre-compiled design that are used to interface with the transmit and receive PMA. Location assignments are added to these registers to optimize timing.
- Clock selection—The script also adds clock assignments to the registers used to interface with the transmit and receive channels configured in Basic (PMA Direct) mode. Typically, you would use a clock with the least skew with respect to the data path to the meet timing requirement.

Each of the assignments has an "hssi_place" tag. To see the tags in the Assignment Editor, right-click on the column header and enable the Tag column.

☞ The timing closure script only runs on designs that have already achieved a fit. The script requires placement of all transceiver logic and all registers interfacing with the transceivers.

The script, with the filename **pmadirect_ff_placer.tcl**, requires one of a set of device-specific map files **<*device name*>_map.tcl**; for example, **EP4GX230_map.tcl**, for a design targeting an EP4GX230 device. To run the script you must have both the script and the specific device map.

To run the script, copy the script and the device map to the project directory and run them with the **quartus_cdb** executable using the following format:

```
quartus_cdb -t pmadirect_ff_placer.tcl <project> <options>
```

☞ You must run the script at a system command prompt and not at the Tcl console of the Quartus II software.

Table 1 describes the script options for pmadirect_ff_placer.tcl.

Table 1. Arguments for Timing Closure Script

Variable	Description
<*project*>	Project name
<*options*>	Options for the **pmadirect_ff_placer.tcl** are -undo for removing all the assignments made previously added by the script, and -test to display the assignments the script will add without making any assignments.

You may get errors running the script if you copy the text from this application note directly to the system command prompt. To avoid these errors, type the command in full at the system command prompt.

To remove the assignments added by the script, use the -undo option as shown in the following example:

```
quartus_cdb -t pmadirect_ff_placer.tcl <project> -undo
```

Document Revision History

Table 2 lists the revision history for this application note.

Table 2. Template Revision History

Date	Revision	Changes
July 2015	4.0	■ Changed the direction of the tx_clkout signal in the "Multiple Channels Using FIFOs" figure.
January 2011	3.0	■ Updated Figure 2, Figure 3, Figure 4, Figure 5, Figure 6, and Figure 8. ■ Updated the "Meeting Timing on the Transmit Side", "Meeting Timing on the Transmit Side", "Meeting Timing on the Receive Side", and "Timing Closure for the Quartus II Software Version 9.1 SP1 and Earlier" sections. ■ Added the "Manage Clock Resource Allocation". "Control Placement", "Software Settings", "Analyzing the Design", and "Combining High-Speed Serial Interfaces with Other Interfaces" sections. ■ Added Figure 7, Figure 9, and Figure 10. ■ Added Equation 1. ■ Removed Figure 2. ■ Reorganized the information. ■ Converted to the new template. ■ Minor text edits.

Table 2. Template Revision History

Date	Revision	Changes
February 2010	2.0	■ Updated Page 1. ■ Updated the “Achieving Timing Closure”, “Steps to Achieve Timing Closure”, and “Summary” sections. ■ Updated Figure 1, Figure 4, Figure 5, Figure 6, Figure 7, and Figure 8. ■ Added link to the associated scripts. ■ Removed the “Design Considerations” and “PLL Method” sections. ■ Minor text edits.
June 2009	1.0	Initial release.