ALTDQ_DQS2 IP Core User Guide

2017.05.08

UG-01089

Subscribe Send Feedback

The Altera ALTDQ_DQS2 megafunction IP core controls the double data rate (DDR) I/O elements (IOEs) for the data (DQ) and data strobe (DQS) signals in Arria® V, Cyclone® V, and Stratix® V devices. A DQ group is composed of one DQS, one optional complementary DQS, and up to 36 configurable DQ I/Os.

Related Information

- **ALTDQ_DQS2 IP Core User Guide Archives** on page 95
  Provides a list of user guides for previous versions of the ALTDQ_DQS2 IP core.
- **Introduction to Altera IP Cores**

ALTDQ_DQS2 Features

The ALTDQ_DQS2 IP core has the following features:

- Access to dynamic on-chip termination (OCT) controls to switch between parallel termination during reads and series termination during writes.
- High-performance support for DDR interface standards.
- 4- to 36-bit programmable DQ group widths.
- Half-rate registers to enable successful data transfers between the I/O registers and the core logic.
- Access to I/O delay chains to fine-tune delays on the data or strobe signals.
- Access to hard read FIFO.
- Access to latency shifter FIFO and data valid FIFO for efficient control of DQS gating and read operations (Arria V and Cyclone V devices only).

ALTDQ_DQS2 Device Support

The ALTDQ_DQS2 IP core supports the following devices:

- Arria V devices
- Cyclone V devices
- Stratix V devices

Resource Utilization and Performance

To view the compilation reports in the Quartus® Prime software, follow these steps:

1. On the Processing menu, click **Start Compilation** to run a full compilation.
2. After compiling the design, on the Processing menu, click **Compilation Report**.
3. In the Table of Contents browser, expand the **Fitter** folder by clicking the "+" icon.
4. Expand **Resource section**, and select **Resource Usage Summary** to view the resource usage information.
5. Expand **Resource section**, and select **Resource Utilization by Entity** to view the resource utilization information.

ALTDQ_DQS2 Parameter Settings

You can instantiate and parameterize using the IP Catalog and parameter editor GUI, or the `ip-generate` command through the command-line interface (CLI).

The following table lists the ALTDQ_DQS2 parameter settings.

Table 1: ALTDQ_DQS2 Parameter Settings

Parameter Editor GUI Setting		CLI Parameter		Description
Name	Legal Values	Name	Legal Values[1]	
General Settings				
Pin width	**1** to **36**	`PIN_WIDTH`	`1 to 36`	This setting specifies the number of data (DQ) pins to make as part of this DQS group. The default value is **9**. Cyclone V devices support only x8/x9 DQ/DQS groups. The maximum number of pins including strobes is 12.

Pin type	**input**
	output
	bidir
		`PIN_TYPE`	`input`
			`output`
			`bidir`
				This setting specifies the direction of the data pins (input, output, or bidirectional).
				The default value is **bidir** (`bidir`).

(1) All CLI parameter values are type string, therefore you must enclose the values in double quotes.

Parameter Editor GUI Setting		CLI Parameter		Description
Name	Legal Values	Name	Legal Values[1]	
Extra output-only pins	**0 – 36**	EXTRA_OUTPUT_WIDTH	0 to 36	This setting specifies the extra output pins that you need as part of a DQS group. A common use for this setting is to add datamask pins. The default value is **0** (0).
Memory frequency	**1–1068**	INPUT_FREQ	120–1068	This setting specifies the full-rate clock frequency of the incoming DQS group signal from the external device in MHz. The default value is **300** MHz (300).
Use DLL Offset Control	—	USE_OFFSET_CTRL	true false	This setting enables dynamic control of the DLL offset. Altera recommends using this setting for test purposes only. For DQS data capture calibration, use the D1, D2, D3, and D4 delay chains.
Enable hard FIFOs	—	USE_HARD_FIFOS	true false	This setting enables the hard FIFOs (read FIFO for Stratix V devices and read FIFO, latency shifter FIFO and data valid FIFO for Arria V and Cyclone V devices) as part of the ALTDQ_DQS2 IP core.

(1) All CLI parameter values are type string, therefore you must enclose the values in double quotes.

Parameter Editor GUI Setting		CLI Parameter		Description
Name	Legal Values	Name	Legal Values[1]	
Use Capture Clock to clock the read Side of the Hard VFIFO	—	USE_DQSIN_FOR_VFIFO_READ	true false	Turn on this setting when you use the hard data valid FIFO and when the capture clock is not gated. This setting is available only for Arria V and Cyclone V devices.
Enable dual write clocks	—	DUAL_WRITE_CLOCK	true false	This setting enables the use of separate output clocks for data and strobe. This setting is disabled by default for Arria V and Cyclone V devices.
Use dynamic configuration scan chains	—	USE_DYNAMIC_CONFIG	true false	This setting enables run-time configuration of multiple delay chains, phase shifts, and transfer registers. Requires a correctly formatted bitstream. For more information, refer to **DQS Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices** on page 36 and **DQS Configuration Block Bit Sequence for Arria V and Cyclone V Devices** on page 50.
Output Path				

(1) All CLI parameter values are type string, therefore you must enclose the values in double quotes.

Parameter Editor GUI Setting		CLI Parameter		Description
Name	Legal Values	Name	Legal Values[1]	
Use half-rate output path	—	HALF_RATE_OUTPUT	true false	This setting doubles the width of the data bus on the FPGA side and clocks the FPGA side interface using the half-rate clock input. If this setting is enabled, drive the hr_clock_in port with the half-rate clock signal. When enabling hard read FIFO in a Stratix V device, you must set this parameter to true. This setting is enabled by default.
Use output phase alignment blocks	—	USE_OUTPUT_PHASE_ALIGNMENT	true false	This setting enables phase shift on the output path based on the delay settings from the DLL. This setting is disabled by default.
Capture Strobe				
Capture strobe type	**Single** **Differential** **Complementary**	CAPTURE_STROBE_TYPE	single differential complementary	This setting specifies the type of capture strobe (DQS signal from the external device). The default value is **Single** (single).

(1) All CLI parameter values are type string, therefore you must enclose the values in double quotes.

Parameter Editor GUI Setting		CLI Parameter		Description
Name	Legal Values	Name	Legal Values[1]	
Use inverted capture strobe	—	INVERT_CAPTURE_STROBE	true false	When enabled, this parameter captures data with an inverted capture strobe. Strobe inversion occurs from dqsbusout (an output port from the DQS delay chain block) to the clock input of the DDIO_IN block. This setting is enabled by default.
DQS phase shift	**0** degrees **45** degrees **90** degrees **135** degrees	DQS_PHASE_SETTING	0 1 2 3	This setting specifies the phase shift value for the DQS delay chain to shift the incoming strobe in the data valid window during read and write operations. The default value is **90** degrees (2). Arria V and Cyclone V devices support only 0 and 90 degrees.
Use capture strobe enable block [2]	—	USE_DQS_ENABLE	true false	This setting enables the capture strobe enable block, which allows control over the preamble state of the capture strobe. This setting is disabled by default. This setting is available for Arria V GZ and Stratix V devices only.

[1] All CLI parameter values are type string, therefore you must enclose the values in double quotes.
[2] Refer to **KDB Link** if **Use capture strobe enable block** and **Make capture strobe bidirectional** parameters are enabled.

Parameter Editor GUI Setting		CLI Parameter		Description
Name	Legal Values	Name	Legal Values[1]	
Treat the capture strobe enable as a half-rate signal	—	USE_HALF_RATE_DQS_ENABLE	true false	This setting doubles the width of the capture strobe enable bus on the FPGA side and clocks the FPGA side interface using the half-rate clock input.
DQS enable phase setting	**0** degrees **45** degrees **90** degrees **135** degrees	DQS_ENABLE_PHASE_SETTING	0 1 2 3	This setting specifies the value of phase shift to shift the full-rate clock signal that drives the capture strobe enable block. The default value is **0** degrees (0).
Output Strobe				
Generate output strobe	—	USE_OUTPUT_STROBE	true false	This setting generates an output strobe signal based on the OE signal and the full-rate clock. This setting is enabled by default.
Make capture strobe bidirectional [3][4]	—	USE_BIDIR_STROBE	true false	This setting enables the bidirectional capture strobe (capture strobe and output strobe is on the same port). This setting is disabled by default.
Differential/ complementary output strobe	—	DIFFERENTIAL_OUTPUT_STROBE	true false	This setting enables either the differential or complementary output strobe. This setting is disabled by default.

(1) All CLI parameter values are type string, therefore you must enclose the values in double quotes.
(3) For Arria V GZ and Stratix V devices, please refer to **KDB Link** if **Use capture strobe enable block** and **Make capture strobe bidirectional** parameters are enabled.
(4) For Arria V (except GZ) and Cyclone V devices, please refer to **KDB Link** if the parameter is enabled.

Parameter Editor GUI Setting		CLI Parameter		Description
Name	Legal Values	Name	Legal Values[1]	
Use reset signal to stop output strobe	—	USE_OUTPUT_STROBE_RESET	true false	This setting stops the unidirectional output strobe using a user-provided reset signal. The core_clock_in and the reset_n_core_clock_in signals are required.
OCT Source	**Output Strobe Enable** **Data Write Enable** **Dedicated OCT Enable**	OCT_SOURCE	0 1 2	This setting specifies the type of input signal to toggle the OCT control: • **Output Strobe Enable**—Uses the output_strobe_ena input as the OCT control signal. • **Data Write Enable**—Uses the write_oe_in input as the OCT control signal. • **Dedicated OCT Enable**—Adds a oct_ena_in input to the interface, which is used as the OCT control signal. The availability of the **Output Strobe Enable**, **Data Write Enable**,and **Dedicated OCT Enable** are dependent on PIN_TYPE and USE_BIDIR_STROBE parameters. Default value is **Data Write Enable**.

(1) All CLI parameter values are type string, therefore you must enclose the values in double quotes.

Parameter Editor GUI Setting		CLI Parameter		Description
Name	Legal Values	Name	Legal Values[1]	
Preamble type	**high** **low** **none**	PREAMBLE_TYPE	high low none	This setting sets the DQS preamble to **high** (DDR3), **low** (DDR2), or **none**: • When you select **low** and the strobe is bidirectional, the output strobe is held low for the first full rate cycle. • When you select **high** or **none**, the strobe is driven high for the first full rate cycle. • Default value is **low**. **Note:** The ALTDQ_DQS2 IP core does not support DQS tracking.

Related Information

- **Introduction to Altera IP Cores**
  Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- **Creating Version-Independent IP and Qsys Simulation Scripts**
  Create simulation scripts that do not require manual updates for software or IP version upgrades.
- **Project Management Best Practices**
  Guidelines for efficient management and portability of your project and IP files.
- **DQS Configuration Block Bit Sequence for Arria V and Cyclone V Devices** on page 50
- **DQS Configuration Block Bit Sequence for Stratix V Devices** on page 36
- **IP-Generate Command** on page 94

ALTDQ_DQS2 Data Paths

Describes the read and write data paths and using other IP cores with the ALTDQ_DQS2 IP core.

DQ and DQS Input Path

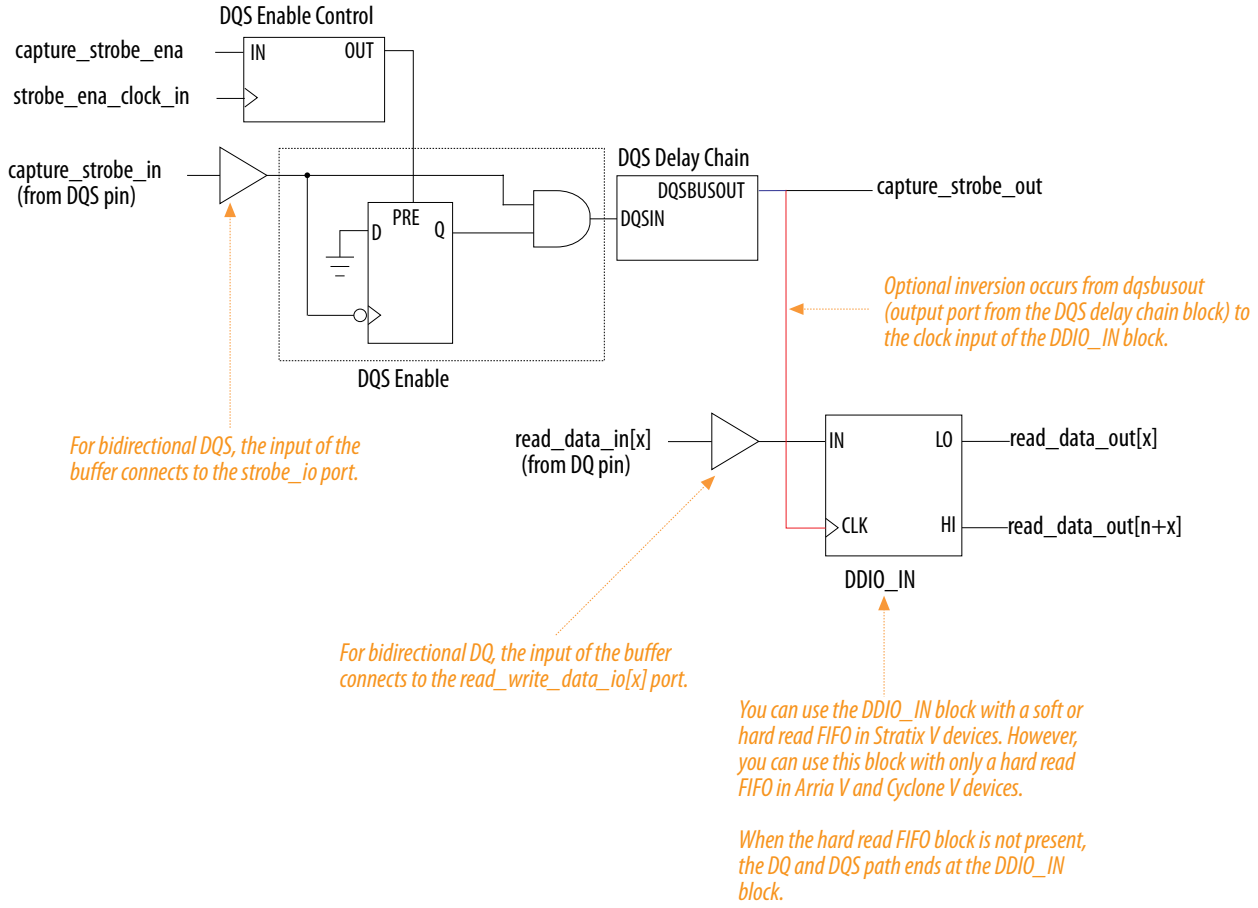
The DQ and DQS input paths receive the DQ and DQS signals from the external device during read operations.

[1] All CLI parameter values are type string, therefore you must enclose the values in double quotes.

DQ and DQS Input Paths for Stratix V Devices

The following figure shows the input paths where x = 0 to (n-1) and n = the number of DQ pins.

Figure 1: DQ and DQS Input Paths for Stratix V Devices



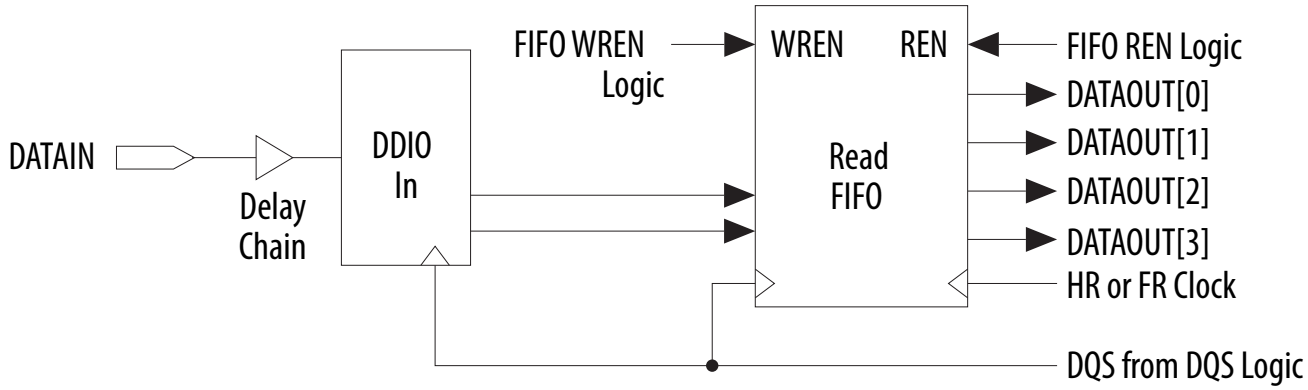
Note: For more information about the DQ and DQS input path with a hard read FIFO block, refer to **Figure 2** and **Figure 3**.

Data Input Path for Arria V, Cyclone V, and Stratix V Devices

The DQ and DQS input paths in Arria V and Cyclone V devices are the same, except for an additional read FIFO block to implement the second-stage rate conversion DDIO. The high-speed 4 x 8 read FIFO, clocked by the DQS clock, implements the half-rate to full-rate conversion, if necessary.

The following figure shows the data input path (when you enable the hard read FIFO) for Arria V, Cyclone V, and Stratix V devices.

Figure 2: Data Input Path for Arria V, Cyclone V, and Stratix V Devices



This figure is applicable to Stratix V devices because Stratix V devices support optional hard read FIFO.

Blocks in DQ and DQS Data Input Path

The following table lists the blocks in the DQ and DQS input paths.

Table 2: Blocks in DQ and DQS Input Path

Block Name	Description
DQS enable	• Represents the AND-gate control on the DQS input that grounds the DQS input strobe when the strobe goes to Hi-Z after a DDR read postamble. The DQS enable block enables the registers to allow enough time for the DQS delay settings to travel from the DQS phase-shift circuitry or core logic to all the DQS logic blocks before the next change. • The clock to the hard data valid FIFO must be synchronous with the `capture_strobe_in` to ensure the DQS enable signal is in-sync with `capture_strobe_in`. • For more information about the DQS enable block, refer to the "Update Enable Circuitry" in the *External Memory Interfaces* chapter in the respective device handbook.
DQS enable control	• Represents the circuitry that controls the DQS enable block. A DQS enable control block controls each DQS enable block. • For more information about the DQS enable control block, refer to the "DQS Postamble Circuitry" in the *External Memory Interfaces* chapter in the respective device handbook.

Block Name	Description
DQS delay chain	• Represents the delay chains that delay signals. • For more information about the DQS delay chain block, refer to "DQS Delay Chain" in the *External Memory Interfaces* chapter in the respective device handbook.

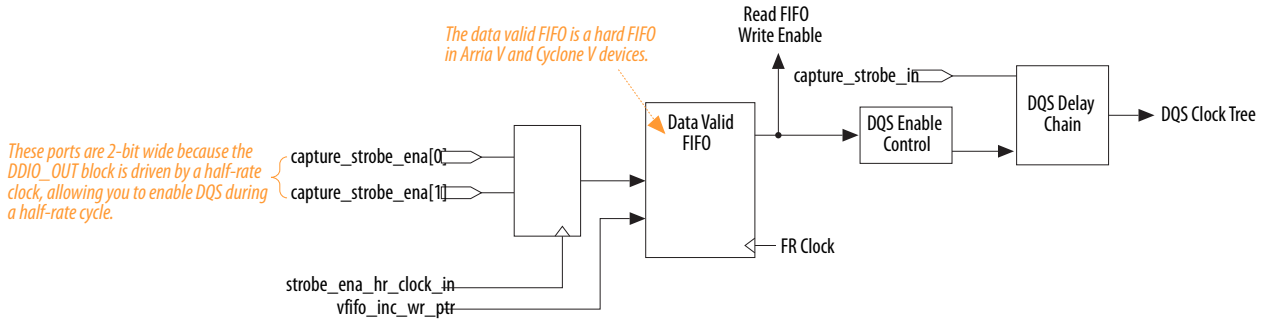
DQS Logic

The DQS input path in Arria V and Cyclone V devices has the following differences from Stratix V and earlier versions of the device families:

- A data valid FIFO delays the DQS enable path by up to 16 full-rate cycles. During a required calibration process, you can increase the unknown delay, which the data valid FIFO implements, by 1, by pulsing the `INC_WR_PTR` port. The delay wraps around after 16 increments.
- The DQS delay chain implements a static non-programmable phase shift of 90°.

The following figure shows the DQS input path in Arria V and Cyclone V devices.

Figure 3: DQS Input Path in Arria V and Cyclone V Devices



Capture DDIO to Read FIFO Path

The capture DDIO block captures input data (DQ) on the rising and falling edges of the capture clock or strobe (DQS).

For Stratix V devices, the capture DDIO block feeds the hard read FIFO or bypasses the hard read FIFO and goes directly to the core. If the capture DDIO block connects directly to the core, the data is transmitted at full rate. For protocols with bidirectional DQS, only an exact number of DQS edge is available for both capturing data in the capture DDIO block and then either in the read FIFO or in the core. The data transfer from the capture DDIO block and the next stage is referred to as zero-cycle transfer. This means that the transfer must happen on the same clock edge.

The hard read FIFO always changes the data rate from full-rate to half-rate, so if you choose to use full-rate, then you cannot use the Read FIFO.

For Arria V and Cyclone V devices, the capture DDIO block to Read FIFO path is similar, with the following exceptions:

- The read FIFO must always be used and cannot be bypassed.
- The read FIFO supports both half-rate and full-rate.

For Arria V, Cyclone V, and Stratix V devices, the hard read FIFO implements the functionality of a generic asynchronous FIFO. You can locate the hard read FIFO in a true dual-ported RAM. Data is written to the write side of the DQS clock domain and read from the read side of the core clock domain. For Arria V and Cyclone V devices, the core clock domain can run at half the frequency and implements a full-rate to half-rate transformation. You can use the write enable and read enable signals to control when to write and read data from the read FIFO. The same signal controls the increment of the write and read pointers. For protocols using a bidirectional strobe, the write enable signal is tied to VCC and DQS gating/ungating implements the write enable functionality.

For Arria V and Cyclone V devices, the hard data valid FIFO internally generates the write enable (gating/ungating) signal and DQS enable signal, while the hard latency FIFO internally generates the read enable signal.

FIFO Control

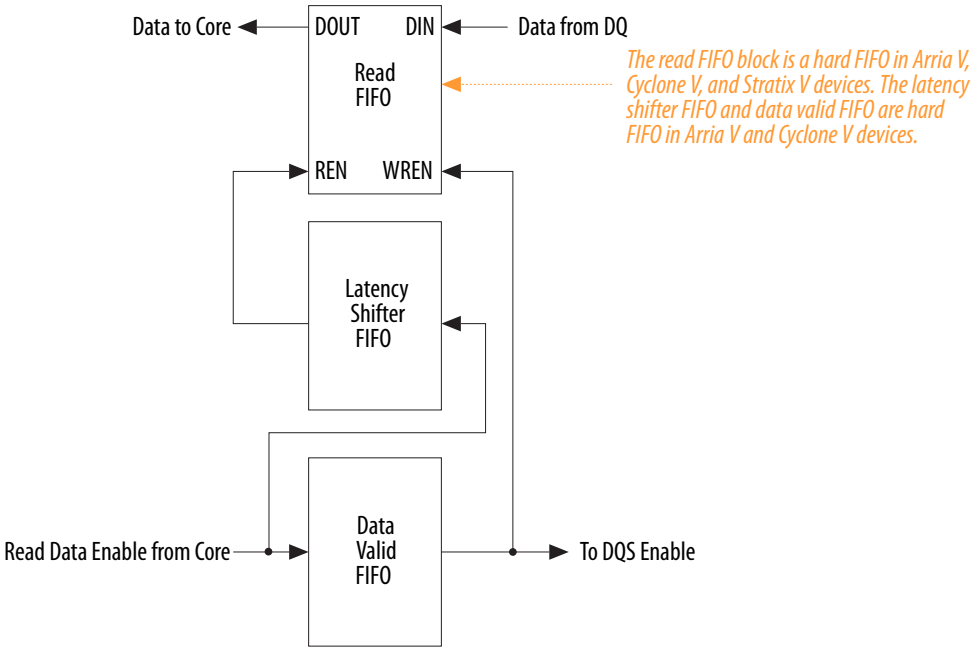
For Arria V and Cyclone V devices, in addition to the read FIFO and the data valid FIFO, the location of the latency shifter FIFO is in each DQS group. The latency shifter FIFO takes in a read-enable command from the core and implements a programmable latency of up to 32 cycles before feeding into the read-enable port of the read FIFO.

You can use the output of the data valid FIFO to perform the following tasks:

- To ungate the DQS logic when a strobe signal is capturing the data. In this case, the write-enable port must always be '1' on the read FIFO.
- To enable the read FIFO write-enable port when a clock is in use.

The following figure shows the three FIFOs interconnection.

Figure 4: Data Valid FIFO, Latency Shifter FIFO, and Read FIFO Interconnection



When a read command is sent to the memory device, a read-data-enable token is pushed through the data valid FIFO and the latency shifter FIFO. The data valid FIFO implements a latency equal to the read command to data latency. When the token comes out of the data valid FIFO, the DQS signal is ungated. The latency shifter FIFO then creates enough space between write and read pointers in the read FIFO to ensure that the data read on the read side is correct. If the read FIFO is read at half-rate, the read FIFO also implements a full-rate to half-rate conversion.

The determination of the correct latencies to implement at each of these FIFOs is important and cannot be done during compilation. When you attempt to implement your own custom memory solution, you must also implement some form of calibration algorithm.

To determine if the data coming from the read FIFO is valid, you must implement the read data valid latency in soft logic.

Related Information

UniPHY External Memory Interface Debug Toolkit

DQ and DQS Output Path

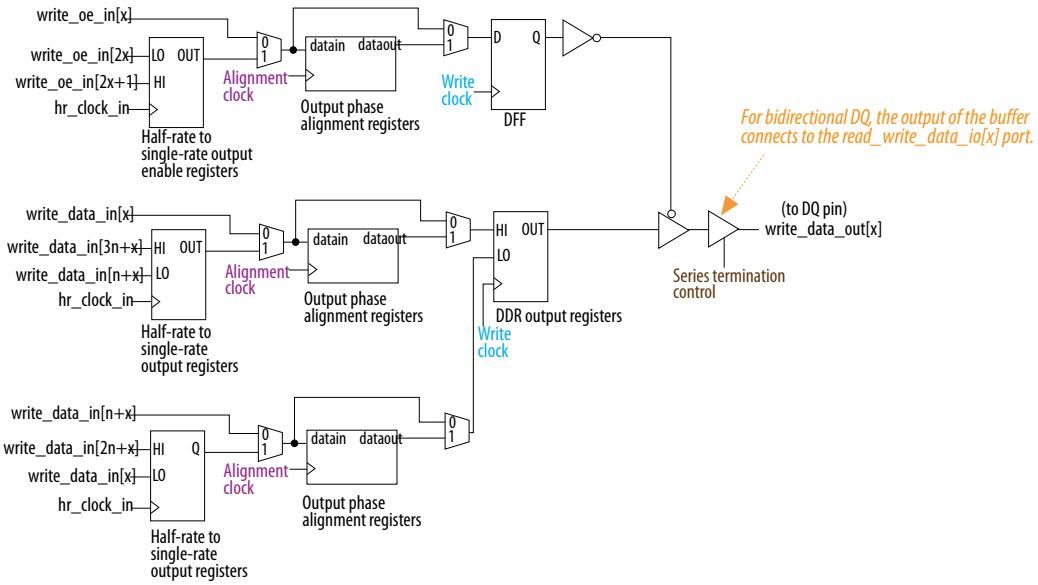
The DQ and DQS output path sends the DQ and DQS signal to the external device during write operations.

DQ and DQS Output Path for Stratix V Devices

The following figure shows the output path where n = the number of DQ pins and x = 0 to (n-1). is only applicable for Stratix V devices.

Note: This figure is only applicable for Stratix V devices. For Arria V and Cyclone V DQ and DQS output path, refer to **Figure 7**.

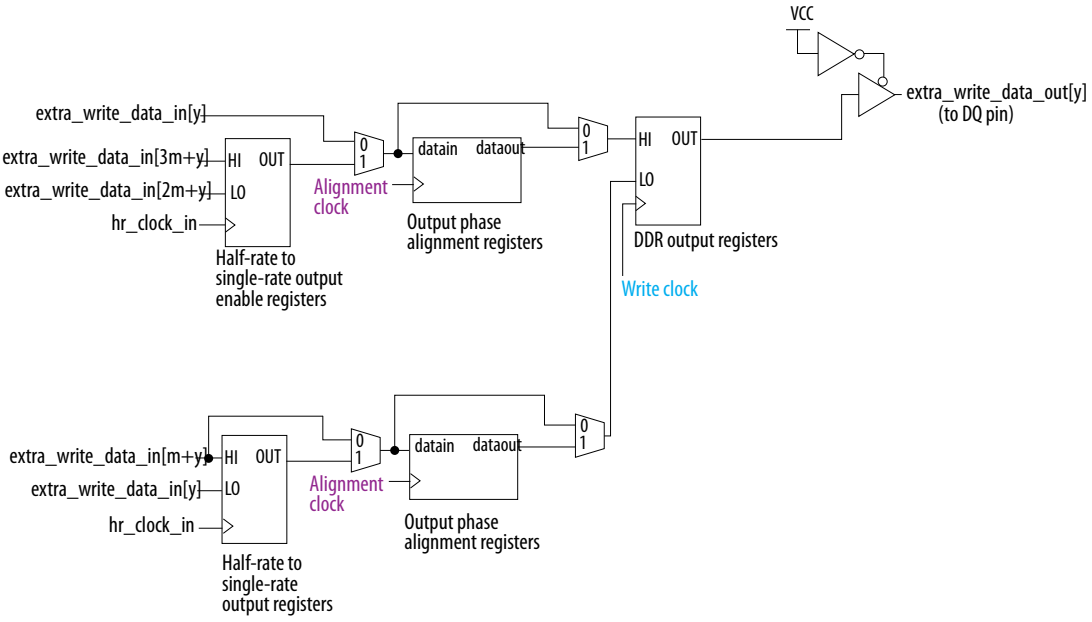
Figure 5: DQ and DQS Output Path for Stratix V Devices



Alignment clock	*The alignment clock comes from the write-leveling delay chains. For more information, refer to "Leveling Circuitry" section in the External Memory Interfaces in Stratix V Devices chapter in the Stratix V Device Handbook.*
Write clock	*The write clock comes from the PLL or the write-leveling delay chains. For more information, refer to "Leveling Circuitry" section in the External Memory Interfaces in Stratix V Devices chapter in the Stratix V Device Handbook.*
Series termination control	*The series termination control connects to the ALTOCT megafunction.*

The following figure shows the DQ and DQS output path for additional DQ pins usage, where y = 0 to (m-1) and m= the number of DQ pins

Figure 6: DQ and DQS Output Path (for Additional DQ Pins Usage) for Stratix V Devices



Alignment clock	The alignment clock comes from the write-leveling delay chains.
Write clock	The write clock comes from the PLL or the write-leveling delay chains.

Related Information

External Memory Interfaces in Stratix V Devices

Blocks in DQ and DQS Output Path

The following table lists the blocks in the DQ and DQS output path.

Table 3: Blocks in the DQ and DQS Output Path

Block Name	Description
Half-rate to single-rate output enable registers	Represents a group of registers that convert half-rate data to single-rate data.
Output phase alignment registers	Represents the circuitry required to phase shift the DQ-output signals. Use this block for write-leveling purposes in DDR3 SDRAM interfaces.
DDR output registers	Represents the DDIO registers that transfer DDR signals from the core to the DQ/DQS pins.

Related Information

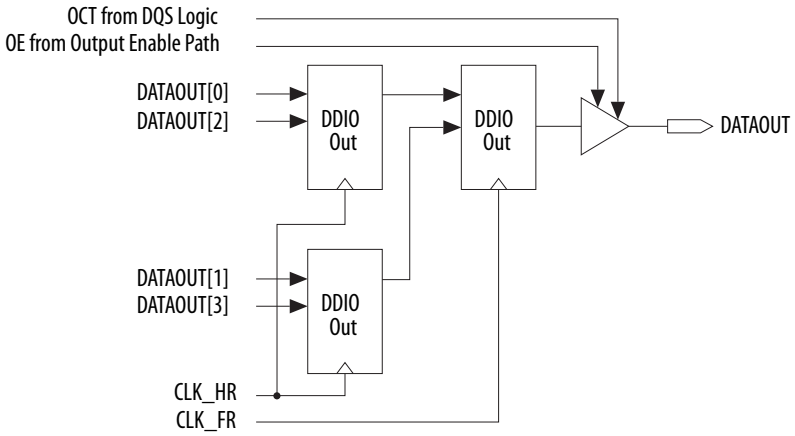
- External Memory Interfaces in Arria V Devices
- External Memory Interfaces in Cyclone V Devices
- External Memory Interfaces in Stratix V Devices

DQ and DQS Output Path for Arria V and Cyclone V Devices

The data output path for Arria V and Cyclone V families is similar to the output paths for Stratix V and earlier families, except for the output phase alignment registers. These registers are not available in Arria V and Cyclone V devices and do not support leveled interfaces.

The following figure shows the DQ and DQS output path for Arria V and Cyclone V devices.

Figure 7: DQ and DQS Output Path for Arria V and Cyclone V Devices

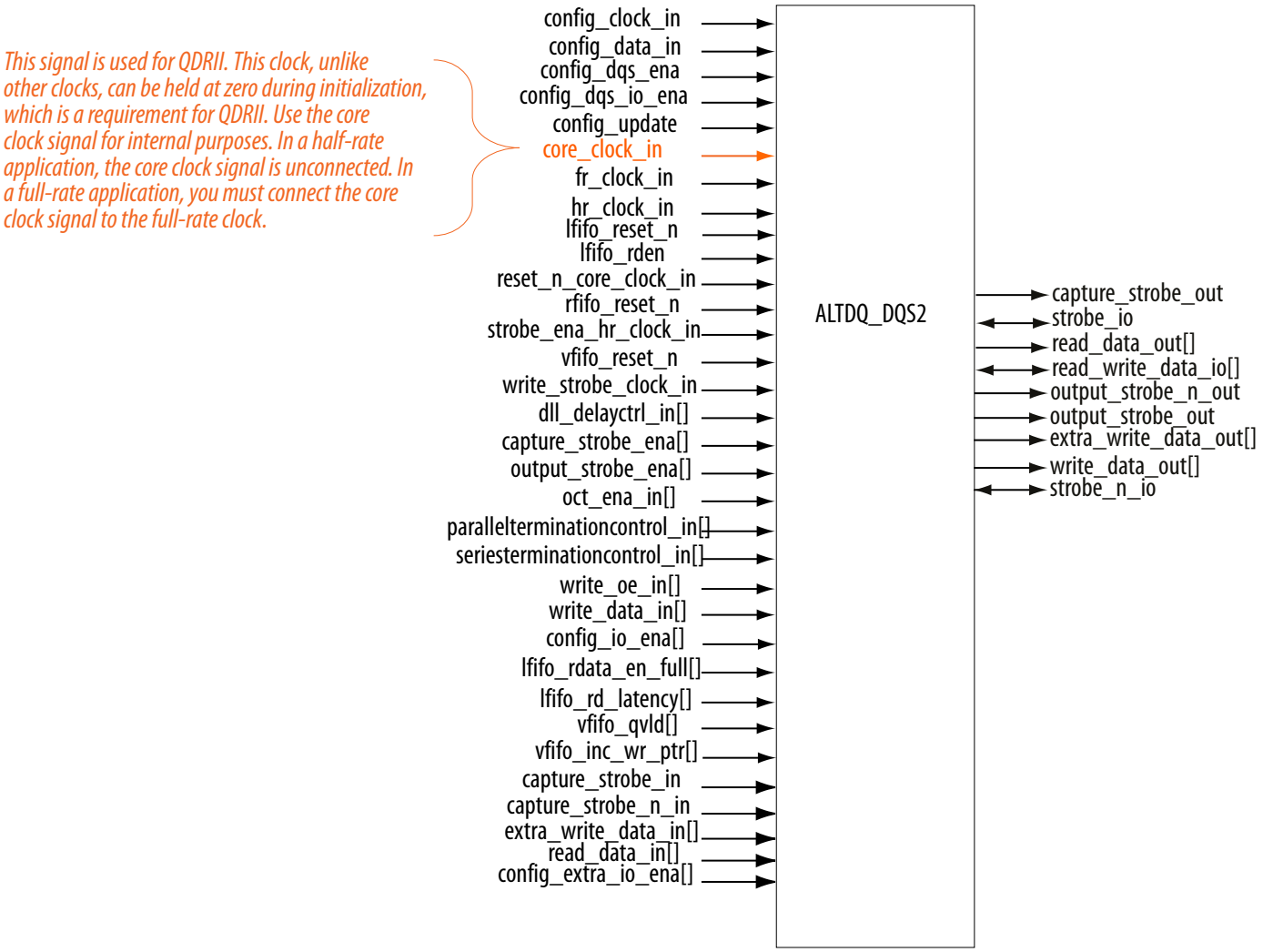


You must connect the ALTDQ_DQS2 IP core to the ALTOCT, ALTDLL, and ALTERA_PLL IP cores to utilize their features.

ALTDQ_DQS2 Ports

The following figure shows the data strobe, data, termination control, PLL, DLL, hard FIFO, and dynamic configuration ports of the IP core.

Figure 8: ALTDQ_DQS2 Block Diagram by Port Types



ALTDQ_DQS2 Data Strobe Ports

Table 4: ALTDQ_DQS2 Data Strobe Ports

Port Name	Type	Width	Description
capture_strobe_ena	Input	1	Controls the DQS enable control block by acting as the gating signal for signals coming from the input registers (capture_strobe_in) to reach the DQS delay chain block. This port is only supported in Stratix V devices.
capture_strobe_n_in	Input	1	Receives the negative polarity clock signal from the external device. For example, a DQSn signal from the external memory. This port is available when the capture strobe type is set to differential or complementary. This port is supported in Arria V, Cyclone V, and Stratix V devices.
capture_strobe_in	Input	1	Receives the clock signal from the external device, for example, a DQS signal from the external memory. This port is supported in Arria V, Cyclone V, and Stratix V devices.
capture_strobe_out	Output	1	Sends the delayed clock signal to the core. For example, a delayed DQS signal from the DQS delay chain. This port is supported in Arria V, Cyclone V, and Stratix V devices.
output_strobe_ena	Input	2 = half-rate 1 = full-rate	The gating signal for the output_strobe_out port. This port is supported in Arria V, Cyclone V, and Stratix V devices.

Port Name	Type	Width	Description
oct_ena_in	Input	2 = half-rate 1 = full-rate	Controls the dynamic on-chip-termination signal for all data and strobe ports. This port is supported in Arria V, Cyclone V, and Stratix V devices.
reset_n_core_clock_in	Input	1	Asynchronous reset used in QDRII-like interfaces to reset the write strobe. This port is supported in Arria V, Cyclone V, and Stratix V devices.
write_strobe_clock_in	Input	1	Receives the clock signal from the core. For example, a DQS signal from the core. Clocks the DDIO that generates the output strobe signal. This port is supported in Arria V, Cyclone V, and Stratix V devices. **Note:** This signal is the main full-rate input clock when you set the IP type to **Input** for Arria V and Cyclone V devices.
strobe_ena_hr_clock_in	Input	1	Receives the clock signal from the clock pin or the PLL to clock the DQS enable control block. Also a half-rate signal that, after going through the DQS_ENABLE_CTRL input, controls the gating of the input strobe. This port is supported in Arria V, Cyclone V, and Stratix V devices.
strobe_io	Bidirectional	1	Sends and receives the bidirectional clock signal. This port is supported in Arria V, Cyclone V, and Stratix V devices.

Port Name	Type	Width	Description
strobe_n_io	Bidirectional	1	Sends and receives the negative polarity clock signal for differential or complementary strobe configuration. This port is supported in Arria V, Cyclone V, and Stratix V devices.
output_strobe_out	Output	1	Sends clock signal to the external device. For example, a DQS signal to the external memory. This port is supported in Arria V, Cyclone V, and Stratix V devices.
output_strobe_n_out	Output	1	Sends the negative polarity clock signal to the external device (For example, DQSn signal to the external memory). This port is available when you set the output strobe type to differential or complementary. This port is supported in Arria V, Cyclone V, and Stratix V devices.

ALTDQ_DQS2 Data Ports

The following table lists the ALTDQ_DQS2 data ports where *n*= number of DQ pins, *m*= number of additional output-only DQ pins, *x* = 0 to (*n*-1), and y= 0 to (m-1).

Table 5: ALTDQ_DQS2 Data Ports

Port Name	Type	Width (5)	Description
`extra_write_data_in[]`	Input	2*m* = full-rate 4*m* = half-rate	Receives data signal from the core. This port connects to the input port of the half-rate data to single-rate data output registers block (**Figure 6**). In full-rate mode, only the `extra_write_data_in[y]` and `extra_write_data_in[m+y]` ports are used. This port is supported in Arria V, Cyclone V, and Stratix V devices.
`extra_write_data_out[]`	Output	*m*	Sends data to the external device. This port connects to the output port of the output buffer (**Figure 6**). This port is supported in Arria V, Cyclone V, and Stratix V devices.
`read_data_in[]`	Input	*n*	Receives data from the external device. This port connects to the input port of the input buffer located between the DQ pin and the DDR input registers block. This is an input-only DQ port that receives data from the external device (**Figure 1**). This port is supported in Arria V, Cyclone V, and Stratix V devices.

(5) The port width applies to full-rate mode, unless otherwise specified.

Port Name	Type	Width (5)	Description
`read_data_out[]`	Output	$2n$ = full-rate $4n$ = half-rate	Sends the captured data from the external device to the core. This port connects to the output port of the DDR input register block (**Figure 1**). The `read_data_out[x]` port outputs the positive-edge triggered data, and the `read_data_out[n+x]` port outputs the negative-edge triggered data. This port is supported in Arria V, Cyclone V, and Stratix V devices.
`read_write_data_io[]`	Bidirectional	n	Receives and sends data between the core and the external device. You must assign the bidirectional DQ port with the output termination and input termination assignments. This port is supported in Arria V, Cyclone V, and Stratix V devices.
`write_data_in[]`	Input	$2n$ = full-rate $4n$ = half-rate	Receives DDR data signal from the core to be sent out to the external device. For example, data to be written to the external memory during write operation. This port connects to the input port of the half-rate data to single-rate data output registers block (**Figure 5**). In full-rate mode, the IP core uses only the `write_data_in[x]` and `write_data_in[n+x]` ports. This port is supported in Arria V, Cyclone V, and Stratix V devices.

(5) The port width applies to full-rate mode, unless otherwise specified.

Port Name	Type	Width (5)	Description
`write_data_out[]`	Output	*n*	Sends the DDR data signal to the external device. For example, data to be written to the external memory during write operation. This port connects to the output port of the output buffer located between the DDR output registers block and the DQ-out pin (**Figure 5**). This port is supported in Arria V, Cyclone V, and Stratix V devices.
`write_oe_in[]`	Input	*n* = full-rate 2*n* = half-rate	Receives the gating signal from the core to control the output buffer. For example, gating control when writing data to the external memory during write operation. This port connects to the input port of the half-rate data to single-rate data output-enable registers block (**Figure 5**). In full-rate mode, the IP core uses only the `write_oe_in[x]` port. This port is supported in Arria V, Cyclone V, and Stratix V devices.

Note: To understand how these ports connect to the IOEs, refer to "I/O Elements" in the **External Memory Interfaces in Stratix V Devices** chapter of the Stratix V Device Handbook.

(5) The port width applies to full-rate mode, unless otherwise specified.

ALTDQ_DQS2 Termination Control Ports

Table 6: ALTDQ_DQS2 Termination Control Ports

Port name	Type	Width	Description
`parallelterminationcontrol_in[]`	Input	16	Controls the calibrated parallel termination ports of the input buffers. You must connect this port to the `parallelterminationcontrol[15:0]` port of the ALTOCT IP core. Ensure that the termination block located in the ALTOCT instance is assigned with the termination control block assignment. This port is supported in Arria V, Cyclone V, and Stratix V devices.
`seriesterminationcontrol_in[]`	Input	16	Controls the calibrated series termination ports of the output buffers. You must connect this port to the `seriesterminationcontrol[15:0]` port of the ALTOCT IP core. Ensure that the termination block located in the ALTOCT instance is assigned with the termination control block assignment. This port is supported in Arria V, Cyclone V, and Stratix V devices.

Related Information

DC and Switching Characteristics for Stratix V Devices
Describes the dynamic OCT control in Stratix V devices.

ALTDQ_DQS2 PLL and DLL Ports

Table 7: ALTDQ_DQS2 PLL and DLL Ports

Port name	Type	Width	Description
`dll_delayctrl_in[]`	Input	7	Receives the 7-bit delay settings from the `dll_delayctrlout` port of the ALTDLL instance. This 7-bit signal controls delay through the DQS delay chains. Compilation error occurs if this port is not connected to a DLL. This port is supported in Arria V, Cyclone V, and Stratix V devices.
`fr_clock_in`	Input	1	Receives the full-rate clock signal from a clock pin, or the PLL clock output port. This port is supported in Arria V, Cyclone V, and Stratix V devices.
`hr_clock_in`	Input	1	Receives the half-rate clock signal from a clock pin, or the PLL clock output port. This port is supported in Arria V, Cyclone V, and Stratix V devices.

Note: For more information about DLL in Stratix V device, refer to "Delay-Locked Loop" in the **External Memory Interfaces in Stratix V Devices** chapter of the Stratix V Device Handbook.

Note: For more information about PLL in Stratix V devices, refer to "PLL Specifications" in **DC and Switching Characteristics for Stratix V Devices** chapter of the Stratix V Device Handbook.

ALTDQ_DQS2 Hard FIFO Ports

Table 8: Hard FIFO Ports

Ports	Type	Width	Description
lfifo_rdata_en_full	Input	2	Data input to the latency shifter FIFO. This signal is the full read enable token generated by user logic and is asserted for the length of the desired read burst. This token is delayed by a variable number of integer cycles inside the latency shifter FIFO and used to feed the read enable signal of the read FIFO. This port is only supported in Arria V and Cyclone V devices.
lfifo_rden	Input	1	Data input to the Read FIFO Read Enable. This signal is the full read enable token generated by user logic and is asserted for the length of the desired read burst. This port is only supported in Stratix V devices.
lfifo_reset_n	Input	1	Active high reset to the latency shifter FIFO This port is only supported in Arria V and Cyclone V devices.
lfifo_rd_latency[]	Input	5	The number of cycles to delay data inputs feeding the latency shifter FIFO. A maximum of 31 cycles is supported. This port is only supported in Arria V and Cyclone V devices.

Ports	Type	Width	Description
vfifo_qvld	Input	Arria V and Cyclone V devices: 2 Stratix V devices: 1	Data input to the data valid FIFO. This signal is the full read enable token generated by user logic and is asserted for the length of the desired read burst. This signal is driven by the same user logic that drives the lfifo_rdata_en_full signal. In general applications, you can leave this port unconnected. This port is supported in Arria V, Cyclone V, and Stratix V devices.
vfifo_inc_wr_ptr	Input	2	Increments the latency implemented by the data valid FIFO by one cycle. This port is only supported in Arria V and Cyclone V devices.
vfifo_reset_n	Input	1	Active high reset to the data valid FIFO This port is only supported in Arria V and Cyclone V devices.
rfifo_reset_n	Input	1	Active high reset to the read FIFO . This port is only supported in Arria V and Cyclone V devices.

The I/O and DQS configuration blocks represent a set of serial-to-parallel shift registers that dynamically changes the settings of various device configuration bits. The I/O and DQS configuration blocks shift a serial configuration data stream into the shift registers, and then load the data stream into the configuration registers. The shift registers power-up low. Every I/O pin contains an I/O configuration block. Every DQS group contains a DQS configuration block and an I/O configuration block.

ALTDQ_DQS2 Dynamic Configuration Ports

The following table lists the dynamic configuration ports where *n*= number of DQ pins and *m*= number of additional DQ pins.

Table 9: ALTDQ_DQS2 Dynamic Configuration Ports

Port name	Type	Width	Description
`config_clock_in`	Input	1	The ALTDQ_DQS2 dynamic configuration interface consists of this input port. Receives the clock signal to clock all dynamic configuration blocks. You can connect this port to a clock pin, or the PLL clock output port. This is the clock signal. All other input signals must be treated as synchronous to this clock. This port is supported in Arria V, Cyclone V, and Stratix V devices.
`config_data`	Input	1	The ALTDQ_DQS2 dynamic configuration interface consists of this input port. The 1-bit serial input through which data is scanned into the calibration blocks. It is common to all configuration blocks, but it will only be scanned into calibrations blocks whose enable input is asserted. This port is supported in Arria V, Cyclone V, and Stratix V devices. For Stratix V GX and Arria V GZ devices, the bitstream for `config_data` must be LSB first (from LSB to MSB). For Arria V and Cyclone V devices, the bitstream for `config_data` must be MSB first (from MSB to LSB).

Port name	Type	Width	Description
`config_io_ena[]`	Input	*n*	An input port that controls the enable input on the DQ I/O configurations. Receives the clock enable signal for the I/O configuration block. Refer to **Dynamic Reconfiguration for ALTDQ_DQS2** on page 31 This port is supported in Arria V, Cyclone V, and Stratix V devices.
`config_dqs_io_ena`	Input	1	An input port that controls the enable input on the DQS I/O configurations. Receives the clock enable signal for the DQS I/O configuration block. Refer to **Dynamic Reconfiguration for ALTDQ_DQS2** on page 31 This port is supported in Arria V, Cyclone V, and Stratix V devices.
`config_dqs_ena`	Input	1	An input port that controls the enable input on the DQS logic. Receives the clock enable signal for the DQS configuration block. Refer to **Dynamic Reconfiguration for ALTDQ_DQS2** on page 31 This port is supported in Arria V, Cyclone V, and Stratix V devices.

Port name	Type	Width	Description
config_update	Input	1	The ALTDQ_DQS2 dynamic configuration interface consists of this input port. Receives the signal to load the bits from the serial-to-parallel shift registers to the configuration registers. After scanning all the bits into the desired scan chain blocks, the bits can be copied at once into the configuration register by asserting the config_update signal for one clock cycle. This port is supported in Arria V, Cyclone V, and Stratix V devices.
config_data_in	Input	1	Receives the serial configuration data stream that shifts into the serial-to-parallel shift registers. This port is supported in Arria V, Cyclone V, and Stratix V devices.
config_extra_io_ena[]	Input	*m*	Receives the clock enable signal for the additional I/O configuration block. This port is supported in Arria V, Cyclone V, and Stratix V devices.

Note: For more information about the dynamic configuration blocks in Stratix V device, refer to "I/O Configuration Block and DQS Configuration Block" in the **External Memory Interfaces in Stratix V Devices** chapter of the Stratix V Device Handbook.

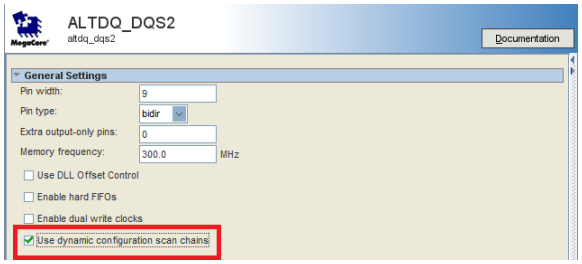
Dynamic Reconfiguration for ALTDQ_DQS2

When static timing closure is challenging (for example, high frequency, high board trace skew, and high timing uncertainty), dynamically reconfiguring the ALTDQ_DQS2 IP core may provide additional timing margin. Arria V, Cyclone V, and Stratix V devices contain reconfigurable logic, allowing you to adjust the delay of several datapaths at runtime.

The I/O configuration block and the DQS configuration block are shift registers that you can use to dynamically change the settings of various device configuration bits. The shift registers are configured with the rest of the device when the Programmer Object File (**.pof**). In dynamic mode, you can override the static values at runtime with a scan chain. You can reconfigure the I/Os by turning on the **Use dynamic configuration scan chains** option.

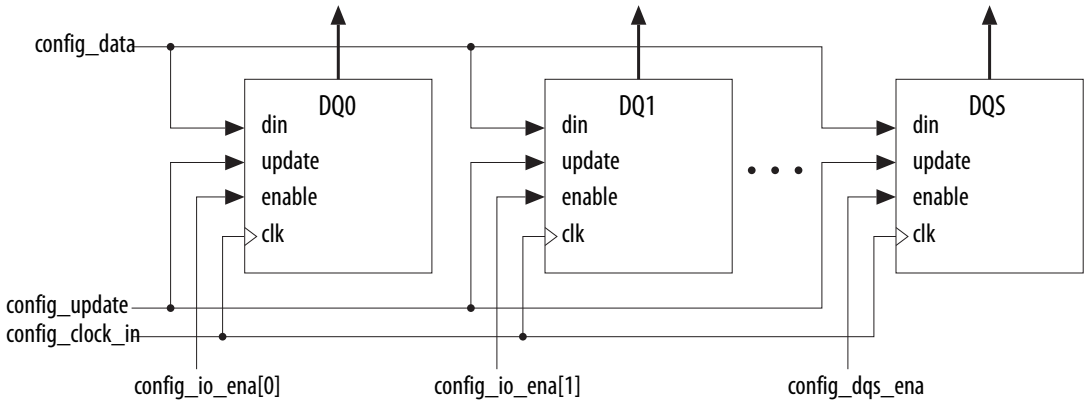
The following figure shows the **Use dynamic configuration scan chains** option.

Figure 9: Use Dynamic Configuration Scan Chains Option



The following figure shows the dynamic reconfiguration scan chain implementation.

Figure 10: Reconfiguration Scan Chain



Each I/O contains a scan chain block. The DQS logic also contains its own scan chain block. You can use I/O scan chain blocks to configure DQ and DQS I/O configuration registers (for example, delay chain) and you can use the DQS logic scan chain to configure DQS logic configuration (for example, DQS

postamble phase). You can serially scan configuration bits into each scan chain block with the following operating sequence:

The ALTDQ_DQS2 dynamic configuration interface is made of four input ports:

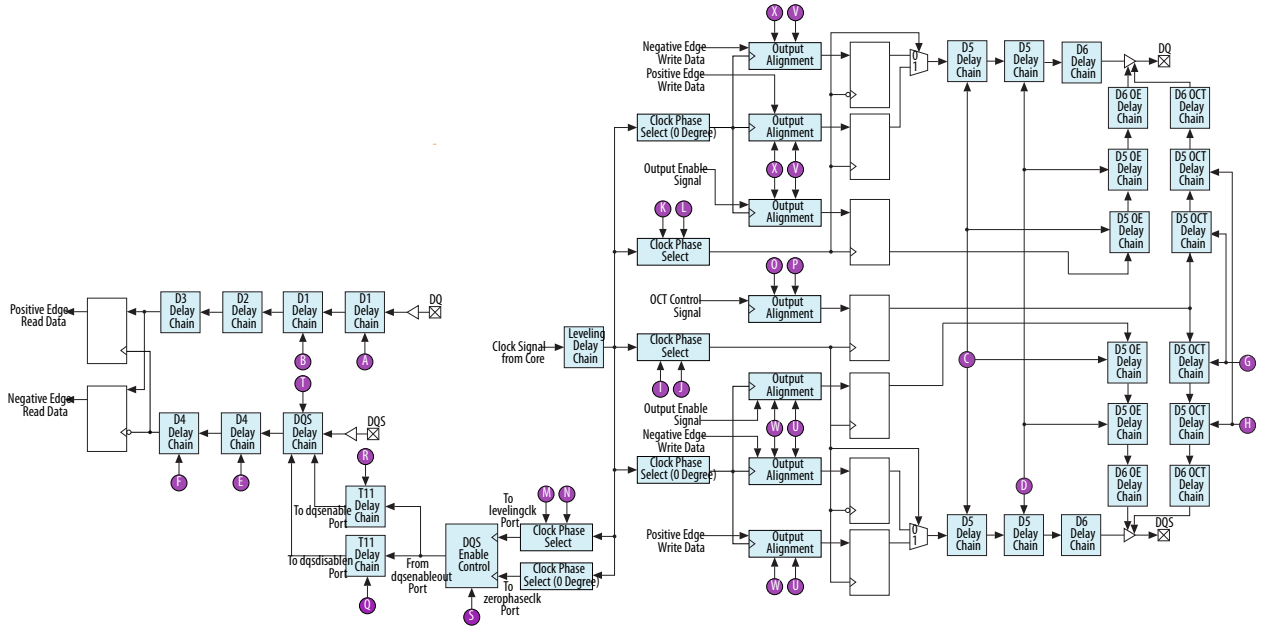
- `config_clock_in`—This is the clock signal. All other input signals must be treated as synchronous to this clock. The typical frequency is 25 MHz.
- `config_data`—This is the 1-bit serial input through which data is scanned into the calibration blocks. This is common to all configuration blocks, but it will only be scanned into calibrations blocks whose enable input is asserted. For Stratix V GX and Arria V GZ, the configuration data must be input in LSB first ordering. For example, the Stratix V I/O configuration block data must start with `padtoinputregisterdelaysetting[0]`. For Arria V and Cyclone V devices, the configuration data must be input in MSB first ordering.
- `config_enable`—In a generic ALTDQ_DQS2 IP core, the following three `config_enable` inputs are available:
  - `config_io_ena[]`—Controls the enable input on the DQ I/Os
  - `config_dqs_io_ena[]`—Controls the enable input on the DQS I/Os
  - `config_dqs_ena[]`—Controls the enable input on the DQS logic

  Note: Each of these inputs is wide to control all the scan chain blocks instantiated in the ALTDQ_DQS2 IP core. In a general application, you must assert only one enable input at a time to scan the desired data in the corresponding scan chain block. The enable input must be held high for the entire duration of the scanning process. All other inputs must be held at 0.

  Note: You must deassert the `config_enable` signal after the last bit of `config_data` to prevent further data from scanning in. Then, assert the `update` signal whenever you are ready to copy the scanned in data to the configuration registers.
- `config_update`—After scanning all the bits into the desired scan chain blocks, copy them into the configuration register by asserting the `config_update` signal for one clock cycle.

I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices

Figure 11: I/O and DQS Delay Chains for Arria V GZ and Stratix V Devices



Use the combination of D1, D2, D3, and D4 delay chains for calibration. Use D1, D2, and D3 to delay DQ, and D4 delay chain to delay DQS. D5 and D6 delay chains are the output delay chains.

The D2 and D3 delay chains are static input delay chains. The D6 delay chain is static output delay chain. You can only set the settings in the Quartus Prime Settings File (.qsf) or the Fitter sets the settings automatically based on the timing constraints. You cannot dynamically set the settings.

The following tables lists the I/O configuration block bit sequence, description, and settings for Arria V GZ and Stratix V devices.

Table 10: I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices

Legend in Figure 11	Bit	Bit Name	Description
A	5..0	padtoinputregisterdelaysetting	Connects to the `delayctrlin` port of the D1 delay chain to control the first I/O buffer-to-input register delay chain (D1). Sets to tune the DQ delay (read calibration) for DDR applications. For delay values, refer to the "Programmable IOE Delay" section in the **Stratix V Device Datasheet**.
B	11..6	padtoinputregisterrisefalldelaysetting	Connects to the `delayctrlin` port of the second D1 delay chain to control the second pad-to-input register delay chain (D1). For delay values, refer to the "Programmable IOE Delay" section in the **Stratix V Device Datasheet**.
C	17..12	outputdelaysetting1	Connects to the `delayctrlin` port of the D5 delay chain to control the output register-to-I/O buffer delay chain (D5) in the output path and output enable paths. This delay is for write calibration for DDR application. For delay values, refer to the "Programmable IOE Delay" section in the **Stratix V Device Datasheet**.
D	23..18	outputdelaysetting2	Connects to the `delayctrlin` port of the second D5 delay chain to control the output register-to-I/O buffer delay chain (second D5) in the output path and output enable paths. This delay is for write calibration for DDR application. For delay values, refer to the "Programmable IOE Delay" section in the **Stratix V Device Datasheet**.

Legend in Figure 11	Bit	Bit Name	Description
—	39..24	inputclkndelaysetting inputclkdelaysetting dutycycledelaymode dutycycledelaysetting	Unconfigurable bits. Always set bits to its default value.

Table 11: I/O Configuration Block Bit Value for Arria V GZ and Stratix V Devices

Bit	Bit Name/ Bit	Default Value (Binary)	Min. Value	Max. Value	Inc. Value
5..0	padtoinputregisterdelaysetting	000000	intrinsic delay	787.5 ps + intrinsic delay	12.5 ps
11..6	padtoinputregisterrisefalldelaysetting	000000	intrinsic delay	787.5 ps + intrinsic delay	12.5 ps
17..12	outputdelaysetting1	000000	intrinsic delay	787.5 ps + intrinsic delay	12.5 ps
23..18	outputdelaysetting2	000000	intrinsic delay	787.5 ps + intrinsic delay	12.5 ps
39..24	inputclkndelaysetting inputclkdelaysetting dutycycledelaymode dutycycledelaysetting	0000000000000000	—	—	—

Related Information

Stratix V Device Datasheet

DQS Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices

The following tables lists the DQS configuration block bit sequence, description, and settings for Arria V GZ and Stratix V devices.

Table 12: DQS Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices

Legend in I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices on page 34	Bit	Bit Name	Description
E	5..0	dqsbusoutdelaysetting	Connects to the delayctrlin port of the first D4 delay chain. Controls the first D4 delay chain in DQS delay chain path (after the DQS delay chain). This is the delay tuning of the DQS signal feeding into the DQS bus. For delay values, refer to the "Programmable IOE Delay" section in the **Stratix V Device Datasheet**.
F	11..6	dqsbusoutdelaysetting2	Connects to the delayctrlin port of the second D4 delay chain. Controls the second D4 delay chain in DQS delay chain path (after the first D4 delay chain). This is the delay tuning of the DQS signal feeding into the DQS bus. For delay values, refer to the "Programmable IOE Delay" section in the **Stratix V Device Datasheet**.
G	17..12	octdelaysetting1	Connects to the delayctrlin port of the D5 OCT delay chain. Controls the dynamic OCT output register-to-I/O buffer delay chain (D5). For delay values, refer to the "Programmable IOE Delay" section in the **Stratix V Device Datasheet**.

Legend in I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices on page 34	Bit	Bit Name	Description
H	23..18	octdelaysetting2	Connects to the `delayctrlin` port of the second D5 OCT delay chain. Controls the dynamic OCT output register-to-I/O buffer delay chain (second D5). For delay values, refer to the "Programmable IOE Delay" section in the **Stratix V Device Datasheet**.
—	27..24	addrphasesetting addrpowerdown addrphaseinvert	Unconfigurable bits. Always set bits to its default value.
I	29..28	dqsoutputphasesetting	Connects to the phasectrlin port of the clock phase select (in the DQS output path) to select between phase shifts of 0°, 45°, 90°, and 135°. Use this bit to level the DQS write output.
—	30	dqsoutputpowerdown	Unconfigurable bits. Always set bits to its default value.
J	31	dqsoutputphaseinvert	Connects to the phaseinvertctrl port of the clock phase select (in the DQS output path) to select between the non-inverted and inverted output. This setting allows the phase output from the delay chain to be inverted to gain additional phases.

Legend in I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices on page 34	Bit	Bit Name	Description
K	33..32	dqoutputphasesetting	Connects to the phasectrlin port of the clock phase select (in the DQ output path) to select between phase shifts of 0°, 45°, 90°, and 135°. DQ leveling clock select. n Use this bit to level the DQ write output.
—	35..34	dqoutputpowerdown	Unconfigurable bits. Always set bits to its default value.
L	36	dqoutputphaseinvert	Connects to the phaseinvertctrl port of the clock phase select (in the DQ output path) to select between the non-inverted and inverted output. This setting allows the phase output from the delay chain to be inverted to gain additional phases.
—	40..37	resyncinputphasesetting resyncinputpowerdown resyncinputphaseinvert	Unconfigurable bits. Always set bits to its default value.
M	42..41	postamblephasesetting	Connects to the phasectrlin port of the clock phase select (for the DQS enable control block) to select between phase shifts of 0°, 45°, 90°, and 135°. Use this clock phase select block to level the postamble (dqsenablein signal at the DQS enable control block).
—	44..43	postamblepowerdown	Unconfigurable bits. Always set bits to its default value.

Legend in I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices on page 34	Bit	Bit Name	Description
N	45	postamblephaseinvert	Connects to the phaseinvertctrl port of the clock phase select (for the DQS enable control block) to select between the non-inverted and inverted output. Use this clock phase select block to level the postamble (dqsenablein signal at the DQS enable control block). This setting allows the phase output from the delay chain to be inverted to gain additional phases.
—	65..46	dqs2xoutputphasesetting n dqs2xoutputpowerdown dqs2xoutputphaseinvert dq2xoutputphasesetting dq2xoutputpowerdown dq2xoutputphaseinvert ck2xoutputphasesetting ck2xoutputpowerdown ck2xoutputphaseinvert dqoutputzerophasesetting postamblezerophasesetting postamblepowerdown dividerioehratephaseinvert dividerphaseinvert	Unconfigurable bits. Always set bits to its default value.
O	68..66	enaoctcycledelaysetting	Connects to the enaoutputcycle-delay port of the output alignment block (in the dynamic OCT control path) to allow additional registers to be used. Use this bit to adjust the phase of the write-leveled OCT or output data signal.

Legend in I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices on page 34	Bit	Bit Name	Description
P	69	enaoctphasetransferreg	Connects to the enaphasetransferreg port of the output alignment block (in the dynamic OCT control path) to allow an additional negative edge-triggered register to be added to the OCT, output data, or output enable path to satisfy the setup or hold time requirement for the phase transfer.
Q	77..70	dqsdisablendelaysetting	Connects to the delayctrlin port of the T11 delay chain (located between the dqsenableout port of the DQS enable control block and the dqsdisablen port of the DQS delay chain). This is to align post-amble signal in terms of DQS signal by selecting different delays.
R	85..78	dqsenabledelaysetting	Connects to the delayctrlin port of the T11 delay chain (located between the dqsenableout port of the DQS enable control block and the dqsenable port of the DQS delay chain). This is to align post-amble signal in terms of DQS signal by selecting different delays.
S	86	enadqsenablephasetransferreg	Connects to the enaphasetransferreg port of the DQS enable control block to allow an additional negative edge-triggered register to be added to the DQS enable control path to satisfy the setup or hold time requirement for the phase transfer.

Legend in I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices on page 34	Bit	Bit Name	Description
T	88..87	dqsinputphasesetting	Connects to the phasectrlin port of the DQS delay chain block. To control the phase selection for the DQS delay chain. The frequency range that this works at is 300 MHz to 800 MHz.
U	89	enadqsphasetransferreg	Connects to the enaphasetransferreg port of the output alignment block (in the DQS output path and OE path) to allow an additional negative edge-triggered register to be used.
V	90	enaoutputphasetransferreg	Connects to the enaphasetransferreg port of the output alignment block (in the DQ output path and OE path) to allow an additional negative edge-triggered register to be added to the output data or output enable path to satisfy the setup or hold time requirement for the phase transfer.
W	93..91	enadqscycledelaysetting	Connects to the enaoutputcycledelay port of the output alignment block (in the DQS output path and OE path) to allow additional registers to be enabled in the output alignment block of the output data or output enable path of a DQS I/O. This is normally used to adjust the phase of the write-leveled OCT or output data signal.

Legend in I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices on page 34	Bit	Bit Name	Description
X	96..94	enaoutputcycledelaysetting	Connects to the enaoutputcycle-delay port of the output alignment block (in the DQ output path and OE path) to allow additional registers to be enabled in the output alignment block of the output data or output enable path of a DQ I/O. Use this bit to adjust the phase of the write-leveled OCT or output data signal.
—	100..97	enainputcycledelaysetting enainputphasetransferreg	Unconfigurable bits. Always set bits to its default value.

Table 13: DQS Configuration Block Bit Value for Arria V GZ and Stratix V Device

Bit	Bit Name	Default Value (Binary)	Min Value	Max Value	Inc. Unit
5..0	dqsbusoutdelaysetting	0	intrinsic delay	787.5 ps + intrinsic delay	12.5 ps
11..6	dqsbusoutdelaysetting2	0	intrinsic delay	787.5 ps + intrinsic delay	12.5 ps
17..12	octdelaysetting1	0	intrinsic delay	787.5 ps + intrinsic delay	12.5 ps
23..18	octdelaysetting2	0	intrinsic delay	787.5 ps + intrinsic delay	12.5 ps
27..24	addrphasesetting addrpowerdown addrphaseinvert	100	—	—	—

Bit	Bit Name	Default Value (Binary)	Min Value	Max Value	Inc. Unit
29..28	dqsoutputphasesetting	0	00 = 0° 01 = 45° 10 = 90° 11 = 135°		
30	dqsoutputpowerdown	1	—		
31	dqsoutputphaseinvert	0	0 = bypass 1 = enable		
33..32	dqoutputphasesetting	0	00 = 0° 01 = 45° 10 = 90° 11 = 135°		
35..34	dqoutputpowerdown	10	—		
36	dqoutputphaseinvert	0	0 = bypass 1 = enable		
40..37	resyncinputphasesetting resyncinputpowerdown resyncinputphaseinvert	100	—		
42..41	postamblephasesetting	0	00 = 0° 01 = 45° 10 = 90° 11 = 135°		
44..43	postamblepowerdown	10	—		
45	postamblephaseinvert	0	0 = bypass 1 = enable		

Bit	Bit Name	Default Value (Binary)	Min Value	Max Value	Inc. Unit
65..46	dqs2xoutputphasesetting dqs2xoutputpowerdown dqs2xoutputphaseinvert dq2xoutputphasesetting dq2xoutputpowerdown dq2xoutputphaseinvert ck2xoutputphasesetting ck2xoutputpowerdown ck2xoutputphaseinvert dqoutputzerophasesetting postamblezerophasesetting postamblepowerdown dividerioehratephaseinvert dividerphaseinvert	0010_0000_1000_1000_0100	—		
68..66	enaoctcycledelaysetting	10	000: Not supported 001: Not supported 010: No delay 011: 1 cycle delay 100: 2 cycle delay 101: 3 cycle delay 110: Not supported 111: Not supported		
69	enaoctphasetransferreg	0	0 = bypass 1 = enable		
77..70	dqsdisablendelaysetting	0	intrinsic delay	3.2 ns + intrinsic delay	12.5 ps
85..78	dqsenabledelaysetting	0	intrinsic delay	3.2 ns + intrinsic delay	12.5 ps
86	enadqsenablephasetrans-ferreg	0	0 = bypass 1 = enable		

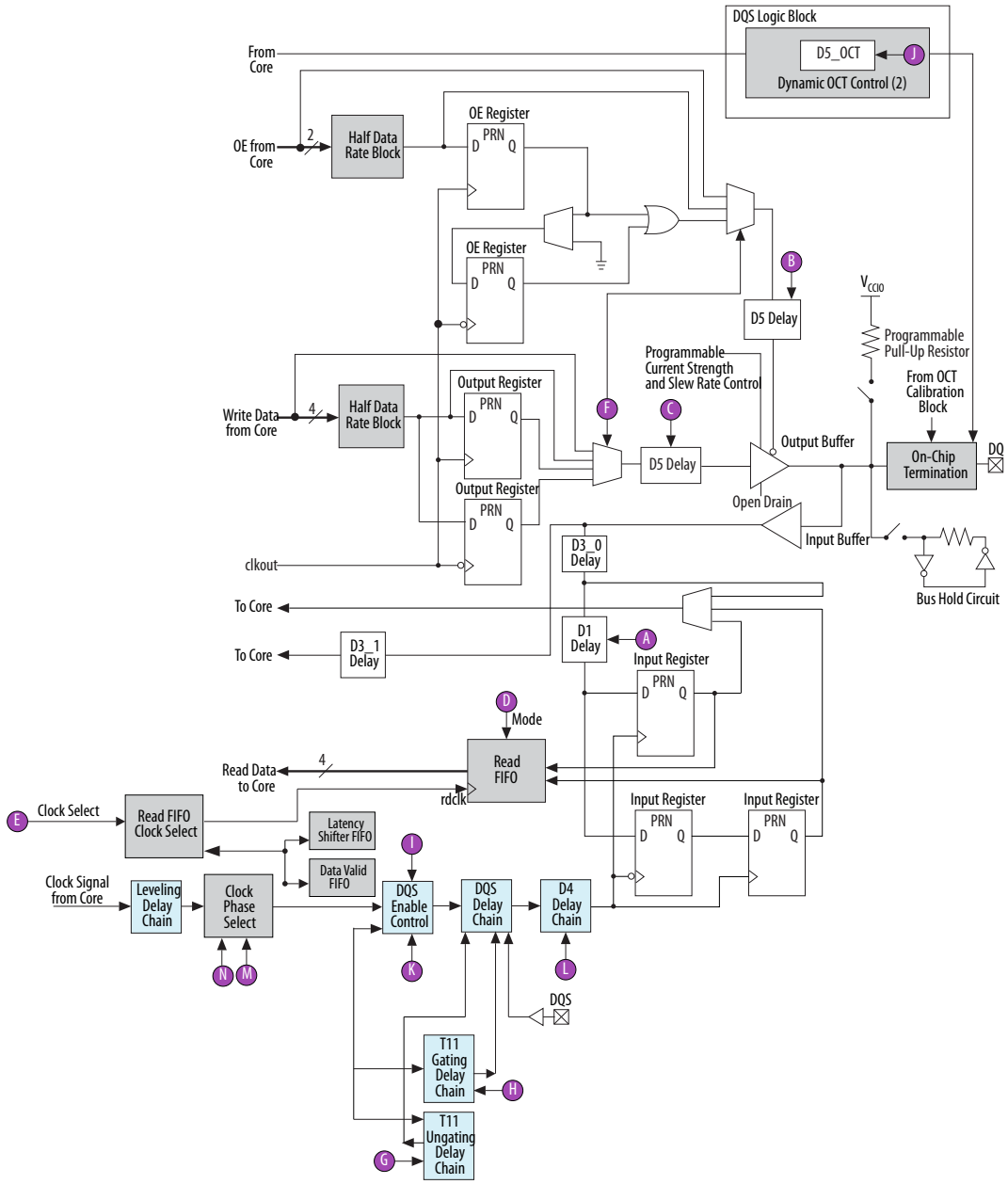
Bit	Bit Name	Default Value (Binary)	Min Value	Max Value	Inc. Unit
88..87	dqsinputphasesetting	0	00 = 0° 01 = 45° 10 = 90° 11 = 135°		
89	enadqsphasetransferreg	0	0 = bypass 1 = enable		
90	enaoutputphasetransferreg	0	0 = bypass 1 = enable		
93..91	enadqscycledelaysetting	10	000: Not supported 001: Not supported 010: No delay 011: 1 cycle delay 100: 2 cycle delay 101: 3 cycle delay 110: Not supported 111: Not supported		
96..94	enaoutputcycledelaysetting	10	000: Not supported 001: Not supported 010: No delay 011: 1 cycle delay 100: 2 cycle delay 101: 3 cycle delay 110: Not supported 111: Not supported		
100..97	enainputcycledelaysetting enainputphasetransferreg	0	—		

Related Information

Stratix V Device Datasheet

I/O Configuration Block Bit Sequence for Arria V and Cyclone V Devices

Figure 12: I/O and DQS Delay Chains for Arria V and Cyclone V Devices



The following table lists the I/O configuration block bit sequence, description, and settings for Arria V and Cyclone V devices.

Table 14: I/O Configuration Block Bit Sequence for Arria V and Cyclone V Devices

Legend in Figure 12	Bit	External Bit Name	Description
A	4..0	padtoinputregisterdelaysetting	Connects to the delayctrlin port of the D1 delay chain. Controls the I/O buffer-to-input register delay chain (D1). Tunes the DQ delay (read calibration) for DDR applications. For delay values, refer to the "Programmable IOE Delay" sections in the **Arria V Device Datasheet** and the **Cyclone V Device Datasheet**, respectively.
B	9..5	outputenabledelaysetting	Connects to the delayctrlin port of the D5 delay chain. Controls the output register-to-I/O buffer delay chain (D5) in the output enable paths. This delay is used for write calibration for DDR application. For delay values, refer to the "Programmable IOE Delay" sections in the **Arria V Device Datasheet** and the **Cyclone V Device Datasheet**, respectively.
C	14..10	outputregdelaysetting	Connects to the delayctrlin port of the D5 delay chain. Controls the output register-to-IO Buffer delay chain (D5) in the output path paths. This delay is used for write calibration for DDR application. For delay values, refer to the "Programmable IOE Delay" sections in the **Arria V Device Datasheet** and the **Cyclone V Device Datasheet**, respectively.

Legend in Figure 12	Bit	External Bit Name	Description
D	17..15	readfifomode	Connects to the dynfifomode port of input register read FIFO block. The read FIFO can be configured as a read FIFO or a Unified SerDes Block.
E	19..18	readfiforeadclockselect	Connects to the clksel port of the read FIFO clock select block. This controls the read FIFO Read clock source Select.
F	20	outputhalfratebypass	Sets the multiplexer in the output enable and output data path logic to dynamically bypass the half-rate to full-rate DDIO. Used only with the hard PHY.
—	24..21	Not mapped to any port	Unconfigurable bits. Always set bits to its default value.

Table 15: I/O Configuration Block Bit Sequence for Arria V and Cyclone V Devices

Bit	Bit Name	Default Value (Binary)	Min. Value	Max. Value	Inc. Value
4..0	padtoinputregisterdelaysetting	0	intrinsic delay	775 ps + intrinsic delay	25 ps
9..5	outputenabledelaysetting	0	intrinsic delay	775 ps + intrinsic delay	25 ps
14..10	outputregdelaysetting	0	intrinsic delay	775 ps + intrinsic delay	25 ps

Bit	Bit Name	Default Value (Binary)	Min. Value	Max. Value	Inc. Value
17..15	readfifomode	0	000: Half-rate Read FIFO Mode 001: Full-rate Read FIFO Mode 010: Deserializer Bit Slip Mode 011: Deserializer with Input from Bit Slip 100: Deserializer with Input from I/O 101: Serializer mode 110: Not supported 111: Not supported		
19..18	readfiforeadclockselect	0	00: Select Core CLKIN1 01: Select DQS_CLK (PHY_CLK) 10: Select SEQ_HR_CLK (PHY_CLK) 11: Select VCC (Disabled)		
20	outputhalfratebypass	0	0: Engage Half-Rate Register 1: Bypass Half-Rate Register		
24..21	Not mapped to any port	0	—	—	—

Related Information

- **Arria V Device Datasheet**
- **Cyclone V Device Datasheet**

DQS Configuration Block Bit Sequence for Arria V and Cyclone V Devices

The following tables lists the DQS configuration block bit sequence, description, and settings for Arria V and Cyclone V devices.

Table 16: DQS Configuration Block Bit Sequence for Arria V and Cyclone V Devices

Legend in Figure 12	Bit	Bit Name	Description
G	4..0	dqsenableungatingdelaysetting	Connects to the delayctrlin port of postamble T11 delay chain (ungated). Aligns the postamble signal in terms of DQS signal by selecting different delays.
H	9..5	dqsenablegatingdelaysetting	Connects to the delayctrlin port of the postamble T11 delay chain (gated). Aligns the postamble signal in terms of DQS signal by selecting different delays.
I	10	enadqsenablephasetransferreg	Connects to the enaphasetrans-ferreg port of the DQS Enable Control block to allow an additional negative edge-triggered register to be added to the DQS enable control path to satisfy the setup or hold time requirement for the phase transfer.
J	15..11	octdelaysetting	Connects to the delayctrlin port of the D5 delay chain. Controls the dynamic OCT output register-to-I/O buffer delay chain (D5). For delay values, refer to the "Programmable IOE Delay" sections in the **Arria V Device Datasheet** and the **Cyclone V Device Datasheet**, respectively.
K	16	dqshalfratebypass	Sets the multiplexers in the DQS enable logic, OCT logic, and FIFO control logic to dynamically switch from half-rate to full-rate configuration.
L	21..17	dqsbusoutdelaysetting	Connects to the delayctrlin port of the read DQS D4 delay chain. Controls the delay tuning of the DQS signal feeding into the DQS bus.

Legend in Figure 12	Bit	Bit Name	Description
M	22	postamblephaseinvert	Connects to the phaseinvertctrl port of the clock phase select block to select between the non-inverted and inverted output. This clock phase select block is used to level the postamble clock (in leveling multiplexer). This setting allows the phase output from the delay chain to be inverted to gain additional phases.
N	24..23	postamblephasesetting	Connects to the phasectrlin port of the clock phase select block to select between phase shifts of 0°, 45°, 90°, and 135°. This particular clock phase select block is used to level the postamble clock (in leveling multiplexer).
—	29..25	Not mapped to any port	Unconfigurable bits. Always set bits to its default value.

Table 17: DQS Configuration Block Bit Value for Arria V and Cyclone V Devices

Bit	External Bit Name	Default Value (Binary)	Min. Value	Max. Value	Inc. Value
4..0	dqsenableungatingdelaysetting	0	intrinsic delay	775 ps + intrinsic delay	25 ps
9..5	dqsenablegatingdelaysetting	0	intrinsic delay	775 ps + intrinsic delay	25 ps
10	enadqsenablephasetransferreg	0	0: Disable Neg-Edge Register 1: Enable Neg-Edge Register		
15..11	octdelaysetting	0	intrinsic delay	775 ps + intrinsic delay	25 ps
16	dqshalfratebypass	0	0: Engage Half-Rate Register 1: Bypass Half-Rate Register		

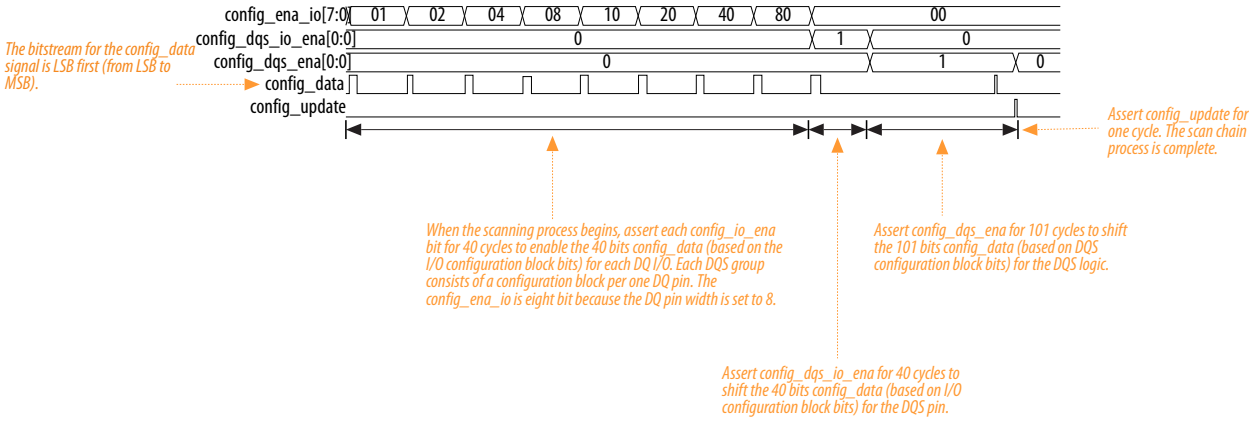
Bit	External Bit Name	Default Value (Binary)	Min. Value	Max. Value	Inc. Value
21..17	dqsbusoutdelaysetting	0	intrinsic delay	775 ps + intrinsic delay	25 ps
22	postamblephaseinvert	0	0 = Non-invert 1 = Invert		
24..23	postamblephasesetting	0	00 = 0° 01 = 45° 10 = 90° 11 = 135°		
29..25	Not mapped to any port	0	—		

Related Information

- Arria V Device Datasheet
- Cyclone V Device Datasheet

Example Usage of Dynamic Reconfiguration for ALTDQ_DQS2

Figure 13: Example Usage of ALTDQ_DQS2 Dynamic Reconfiguration for Stratix V and Arria V GZ Devices



Stratix V Design Example

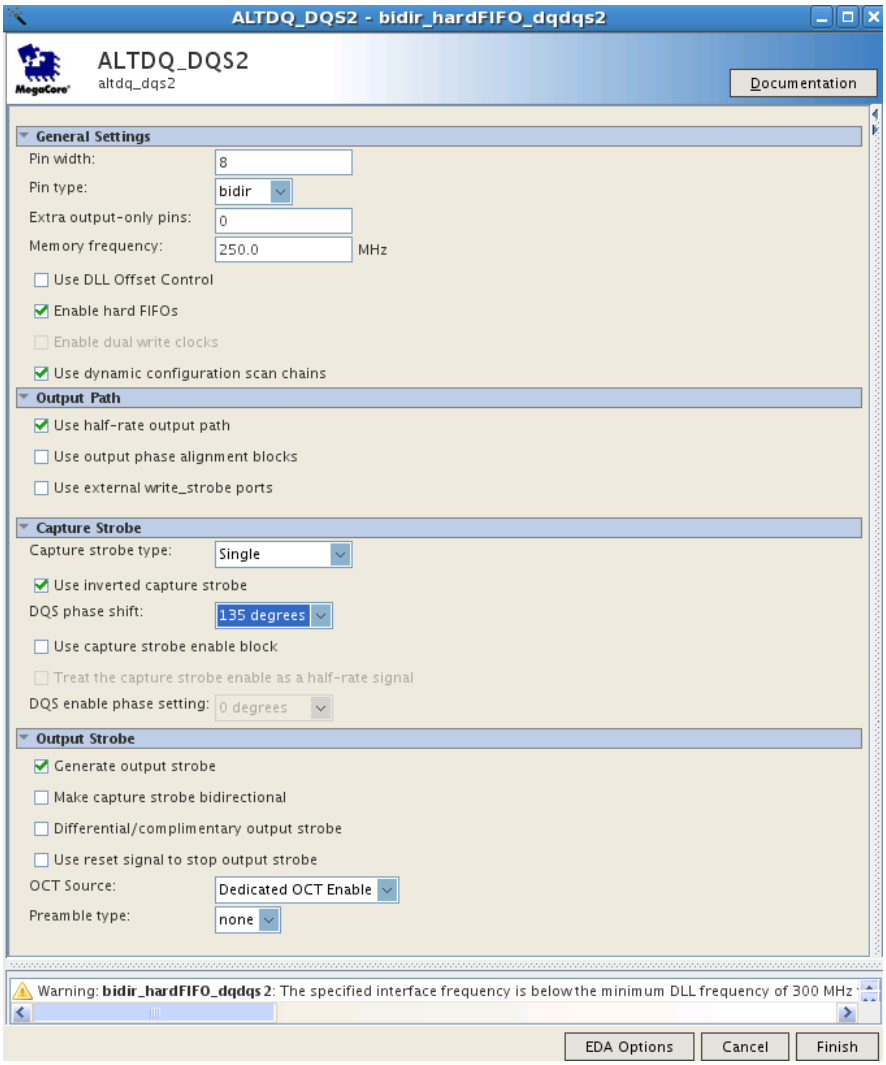
This section describes how to instantiate the ALTDQ_DQS2, ALTERA_PLL, ALTDLL, ALT_OCT IP cores using the **Top_SV_15_1.qar** design example.

Instantiating ALTDQ_DQS2 IP Core

To instantiate the ALTDQ_DQS2 IP core, perform the following steps:

1. In the Quartus Prime software, open the **Top_SV_15_1.qar** and restore the archived file into your working directory.
2. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the ALTDQ_DQS2 v15.1 IP core to customize. The parameter editor appears.
3. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the target Altera device family and output file HDL preference. Click **OK**.
4. On the **Parameter Settings** tab, on the General page, specify the parameters as shown in the following figure. These parameters configure the general settings for the ALTDQD_DQS2 instance.

Figure 14: ALTDQ_DQS2 Parameter Settings for Stratix V Devices



5. Click **Finish**.

   Because the memory frequency is less than the DLL minimum frequency, the DLL needs to be driven by a 2 x frequency (2 x 250 MHz = 500 MHz).

   Because the DLL is driven at doubled frequency, the effective DQS delay is only half of the memory frequency. In this case, to achieve 90 degree shift for memory frequency, the closest setting is 135 degree in the ALTDQ_DQS2 GUI (which is effectively only 135 degree of 500 MHz, or 67.5 degree of 250 MHz).

   Note: The settings in **Figure 14** enables free-running read and write clock. DQS enable block is not needed.

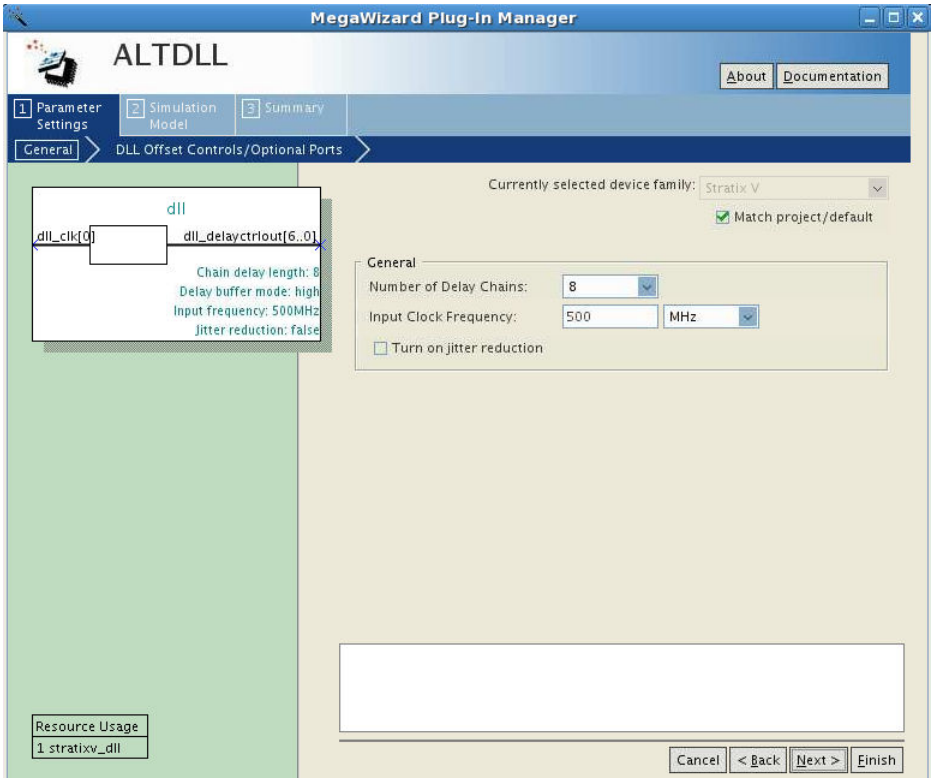
   Note: If your design requires bidirectional strobe, turn on the **Use capture strobe enable block** option. The timing of the capture strobe enables the DQS enable block to know the arrival of the capture strobe which requires round-trip delay information. However, the use of the DQS enable block requires runtime calibration which is not a feature of the ALTDQ_DQS2 IP core.

Instantiating the ALTDLL IP Core

To instantiate the ALTDLL IP core, follow these steps:

1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the target Altera device family and output file HDL preference. Click **OK**.
3. On the **Parameter Settings** tab, on the **General** page, specify the parameters as shown in the following figure.

Figure 15: ALTDLL Parameter Settings Tab

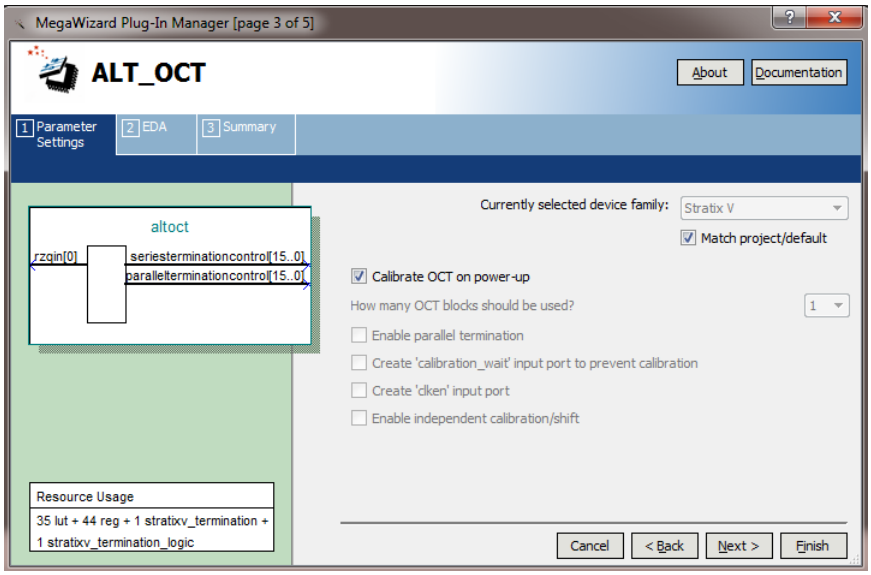


4. Click **Finish**.

Instantiating ALT_OCT IP Core

1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the target Altera device family and output file HDL preference. Click **OK**.
3. On the **Parameter Settings** tab, on the **General** page, specify the parameters as shown in the following figure.

Figure 16: ALT_OCT Parameter Settings Tab

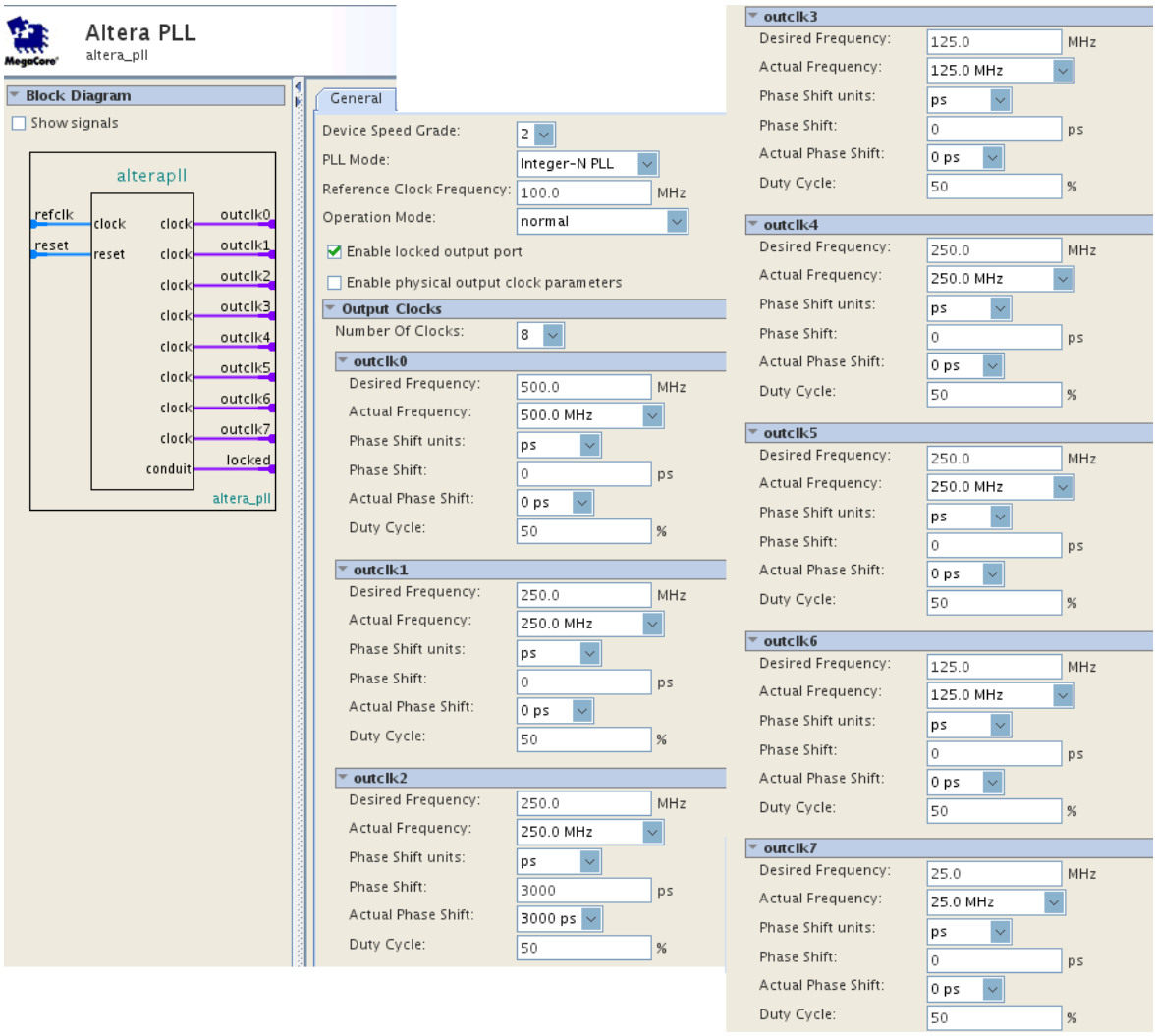


4. Click **Finish**.

Instantiating Altera PLL

1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the target Altera device family and output file HDL preference. Click **OK**.
3. On the **General** tab, specify the parameters as shown in the following figures.

Figure 17: Altera PLL Parameter Settings for Stratix V Devices



4. Click **Finish**.

Altera PLL Clock Settings Information

The following table lists the clock settings Information. You can either merge the similar frequency counters in their design, or the Fitter performs the merging automatically.

Table 18: Altera PLL Clock Settings Information

Clock	Description
outclk_0	600 MHz, used as 2x frequency if necessary.
outclk_1	300 MHz, used as strobe/dqs clock.
outclk_2	300 MHz, 270 degrees phase shifted. Used as data/dq clock.
outclk_3	150 MHz, used as half-rate clock.

Clock	Description
outclk_4	300 MHz, used to drive the ALTDLL IP core. The minimum frequency for the ALTDLL IP core for Stratix V devices is 300 MHz.
outclk_5	300 MHz, used to drive the full rate core clock.
outclk_6	150 MHz, used to drive the half rate core clock.
outclk_7	25 MHz, used as config_clk.

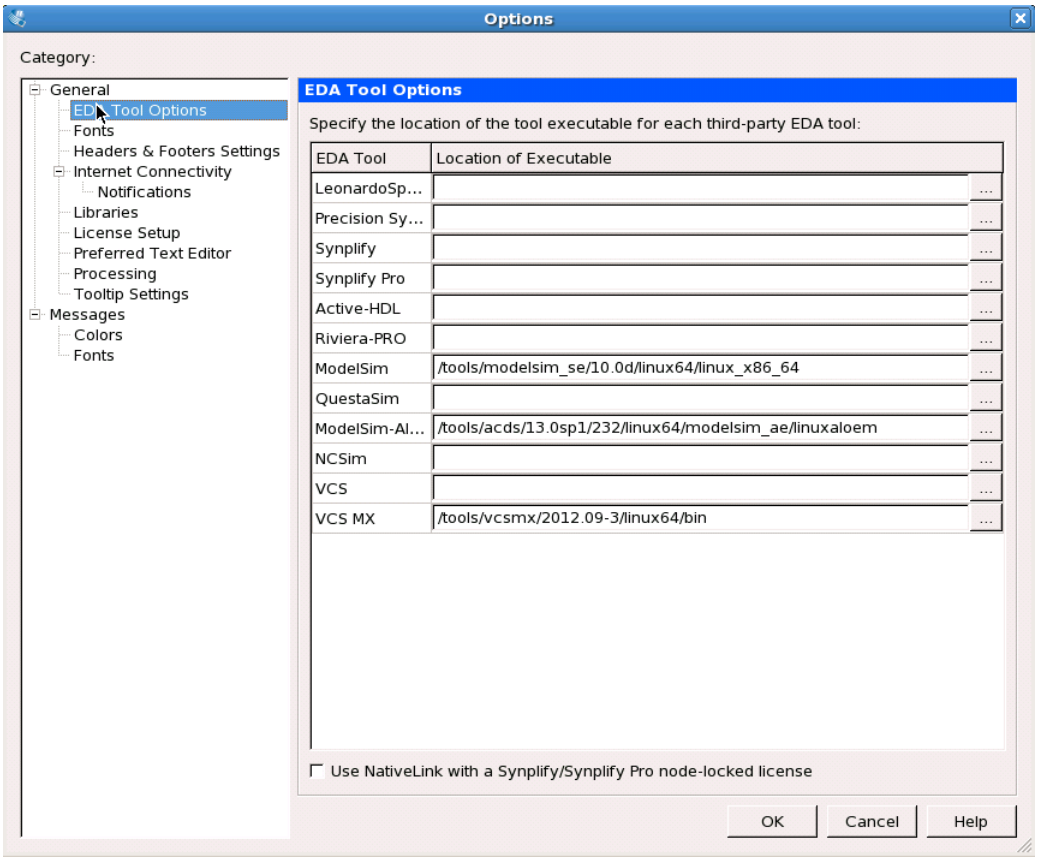
Note: If the memory frequency is less than the ALTDLL IP core minimum frequency, then drive the ALTDLL IP core at 2x or 4x of the memory frequency. The DQS phase settings decrease as well.

Setting Up NativeLink and Simulation Settings

To set up the NativeLink and simulation settings, follow these steps:

1. In the Quartus Prime software, on the Tools menu, select **Options**.
2. In the **Options** dialog box, under Category list, expand General and then select **EDA Tool Options**.
3. In the **EDA Tools Options** window, follow the settings as shown in the following figure:

Figure 18: EDA Tools Options Dialog Box



4. In the Quartus Prime software, on the Assignments menu, click **Settings** > **Simulation**.
5. Create a new testbench and name it `tb` and include all necessary files.

6. Enter the necessary NativeLink settings. The following figure shows an example settings.

Figure 19: Simulation Dialog Box

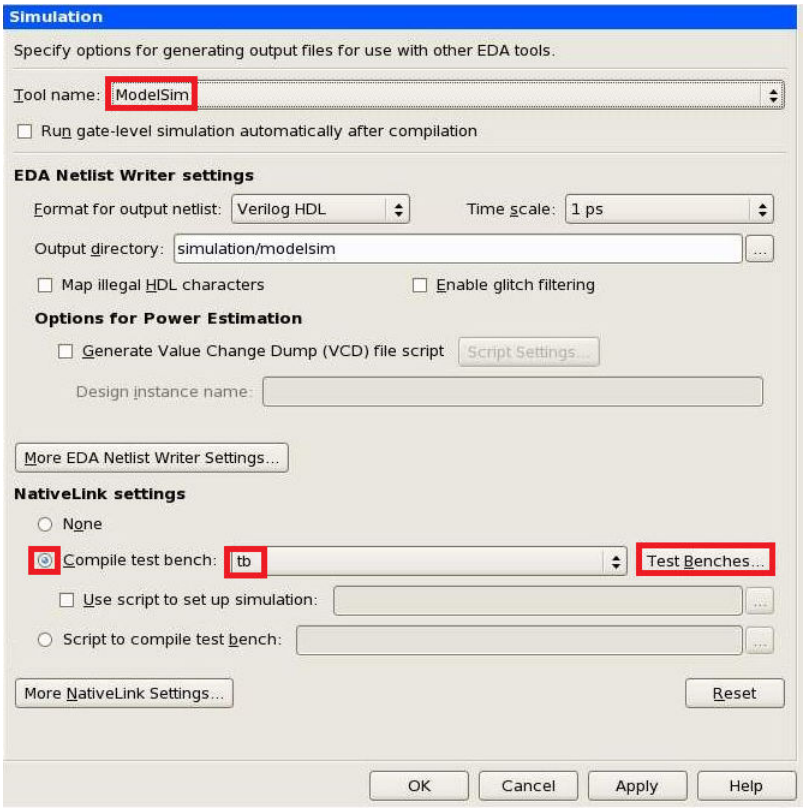
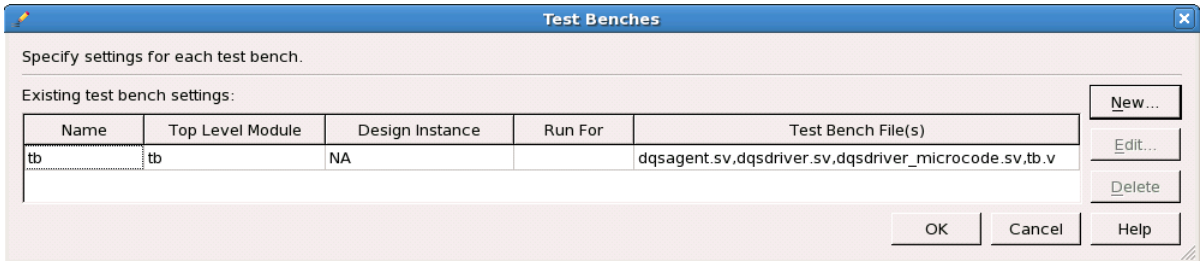


Figure 20: Test Benches Dialog Box



7. Run **Analysis and Synthesis**.
8. To view the simulation results, on the Tools menu, select **Run Simulation Tool** and then click **RTL Simulation.**
   For a successful simulation, you may need to manually change **alterapll.vo** to **alterapll.v** in the auto-generated **top_run_msim_rtl_verilog.do** file.
9. Before running the Fitter, ensure that the following settings are done in the Assignment Editor.

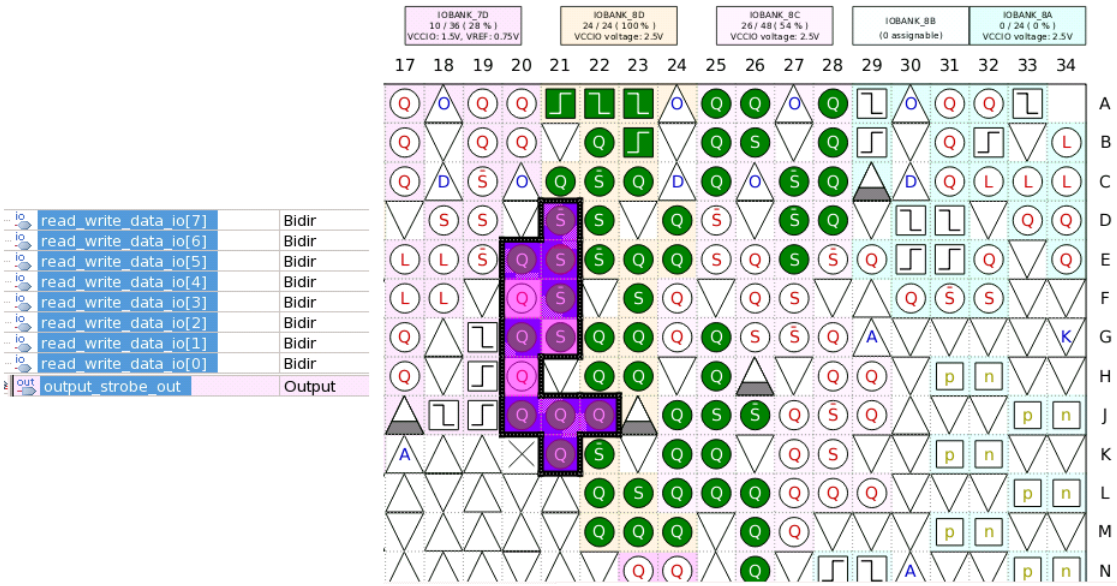
- I/O Standard
- Input Termination
- Output Termination
- DQ Group
- Location assignment for strobe pin—this helps the Fitter to fit the related DQ pins in the appropriate I/O sub-banks. You can then back-annotate the locations if desired.

The following figures show a setting example in the Assignment Editor and the Pin Planner result:

Figure 21: Setting Example in Assignment Editor

	ati	From	To	Assignment Name	Value	Enabled	Entity
1	✓		read_write_data_io[*]	I/O Standard	SSTL-15 Class I	Yes	
2	✓		read_write_data_io[*]	Input Termination	Parallel 50 Oh...th Calibration	Yes	top
3	✓		read_write_data_io[*]	Output Termination	Series 50 Ohm...h Calibration	Yes	top
4	✓		*alterapll_0002*|altera_pll:altera_pll_i*|*	PLL Compensation Mode	Normal	Yes	top
5	✓		*alterapll_0002*|altera_pll:altera_pll_i*|*	PLL Automatic Self-Reset	Off	Yes	top
6	✓		*alterapll_0002*|altera_pll:altera_pll_i*|*	PLL Bandwidth Preset	Auto	Yes	top
7	✓		output_strobe_out	Location	PIN_E21	Yes	
8	✓		capture_strobe_in	I/O Standard	SSTL-15 Class I	Yes	
9	✓		output_strobe_out	I/O Standard	SSTL-15 Class I	Yes	
10	✓		capture_strobe_in	Input Termination	Parallel 50 Oh...th Calibration	Yes	top
11	✓		output_strobe_out	Output Termination	Series 50 Ohm...t Calibration	Yes	top
12	✓	output_strobe_out	read_write_data_io[*]	DQ Group	9	Yes	top
13		<<new>>	<<new>>	<<new>>			

Figure 22: Pin Planner



10. Run the Fitter, Timing Analysis, and Assembler. Refer to SDC Walkthrough on page 69 for more elaboration on the SDC constraint examples included in this design example.

Related Information

Understanding Simulation Results—Stratix V Design Example on page 62

Understanding Simulation Results—Stratix V Design Example

ln the Stratix V design example, a generic testbench is used to test the write and read operations in the ALTDQ_DQS2 IP core. The following table lists the components in the testbench.

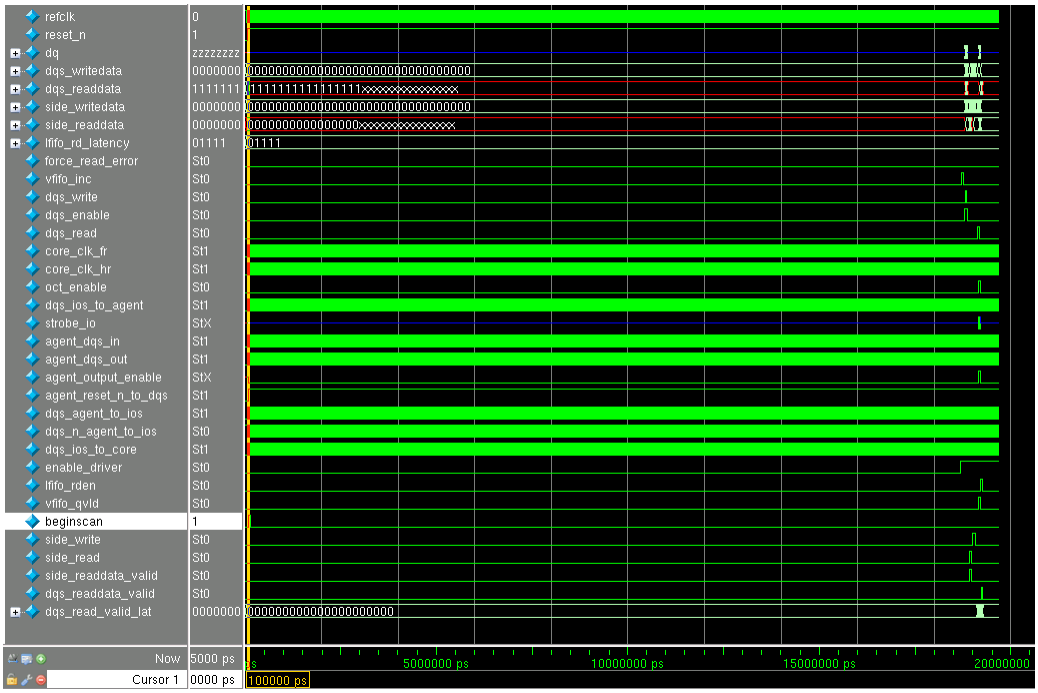
Table 19: Testbench Components

Component	Description
DQS Driver	• Acts as a host controller, sends read/write commands to the ALTDQ_DQS2 IP core. • Compares data read back from a DQS agent to what it should be. • Has a side channel (side reads/writes) communicating directly with the DQS agent, bypassing the ALTDQ_DQS2 IP core. Use the data in the side reads/writes to compare with the data sent to or received from the ALTDQ_DQS2 IP core.
DQS Agent	• Acts as an external memory device. • Has a side channel (side reads/writes) communicating directly with the DQS driver, bypassing the ALTDQ_DQS2 IP core. Use the data in the side reads/writes to compare with the data sent to or received from the ALTDQ_DQS2 IP core.

Note: Random data is generated and used in the testbench. You may see other data values if you are using a different operating system and seeds.

The following figure shows the waveform for the testbench generated after executing the **top_run_msim_rtl_verilog.do** file.

Figure 23: Waveform Example



All ports are in reset mode until the `reset_n` signal is asserted at 70 ns. Then, the core_clk_fr and core_clk_hr clocks start to toggle. The `agent_reset_n_to_dqs` signal is asserted at 91 ns to reset the ALTDQ_DQS2 IP core, which is located in **top_inst**.

Dynamic Configuration

At 0.0001 ms, there is a high pulse on the `beginscan` signal. When the `agent_output_enable` signal is pulled low, some internal calibration is being carried out. Dynamic configuration is the main feature used. At 18.675 ms, the `enable_driver` signal is asserted to specify that the internal calibration is completed. The DQS driver, which acts as the host controller, performs a read/write operation.

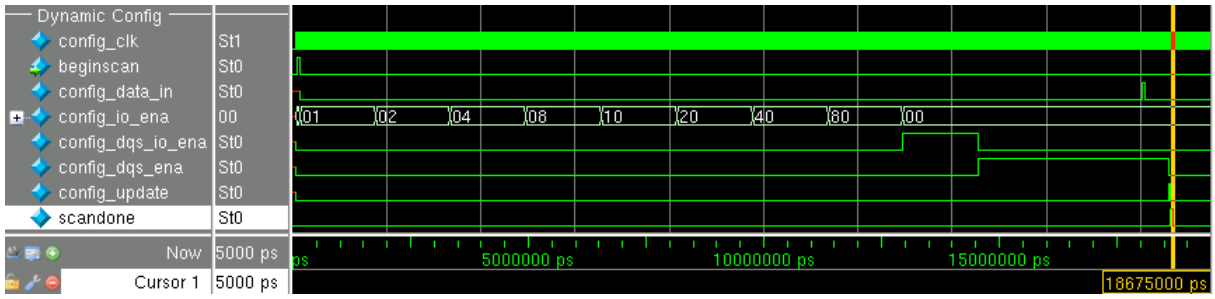
Note: For RTL related to dynamic configuration operations in this design example, refer to **config_controller.sv** file.

To help you achieve static timing closure, the dynamic configuration feature allows you to override the static values at runtime with a scan chain. Each I/O and the DQS logic contains its own scan chain block (shift registers). This section shows you how to serially scan configuration bits into each scan chain block, between 0.0001 ms and 18.675 ms.

The following figure shows the waveform for the dynamic configuration simulation generated after executing **topwave.do**, in between the high pulses of `beginscan` and `scandone`. There is a pulse between 18.035 ms and 18.075 ms on the `config_data_in` signal. This is because the `dqsinputphasesetting` of the DQS configuration is set to 2'b01 in the testbench.

Note: For the results of this settings, refer to **DQS Delay Chain** on page 66.

Figure 24: Dynamic Configuration Waveform



Because the **enable_driver** signal is asserted at 18.675 ms, the DQS driver performs the following operations:

- **DQS Write Operation** on page 64
- **Side Read Operation** on page 65
- **Side Write Operation** on page 65
- **DQS Read Operation** on page 66

Note: The `driver_clk` clock is running at the same rate as the core).

Related Information

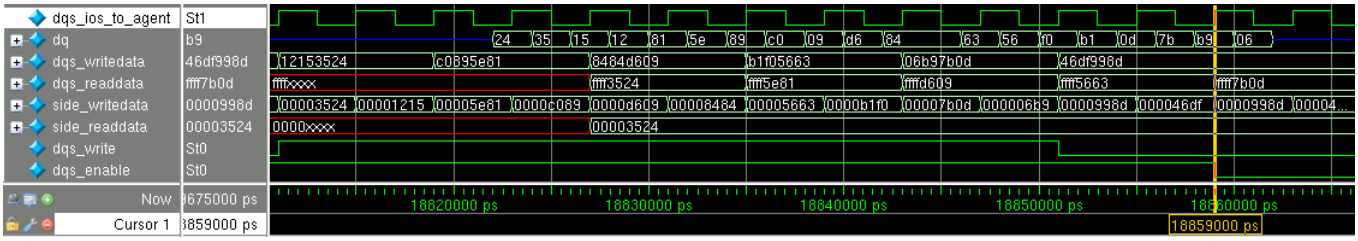
Dynamic Reconfiguration for ALTDQ_DQS2 on page 31

DQS Write Operation

The driver asserting the `dqs_enable` signal at 18.803 ms begins the write operation. The `dqs_write` signal is asserted at 18.811 ms. The `write_oe_in` signal of the ALTDQ_DQS2 IP core is set to high and is ready to send data to the DQS agent. Data are written to the `dqs_writedata` of the DQS driver and then reflected in the `dq` signal of the ALTDQ_DQS2 IP core. The data written out from the DQS driver are stored in `check_fifo`. The `dqs_write` signal deasserts at 18.851 ms. The `dqs_enable` deasserts at 18.859 ms. Outgoing data from the ALTDQ_DQS2 (`dq`) is center aligned to the write clock (`dqs_ios_to_agent`).

The following figure shows the DQS write operation waveform.

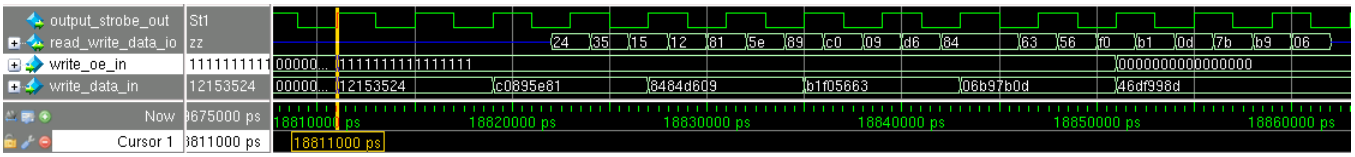
Figure 25: DQS Write Operation Waveform



Note: Before and after the DQS write operation, the `dq` signal is in Hi-Z mode to filter any unwanted glitch on the bidirectional port.

The following figure shows the waveform after executing the **topwave.do** file. The `write_oe_in` signal is held high throughout the five sets of valid `write_data_in` between 18.811 ms and 18.851 ms. Centre aligned output data appears on `read_write_data_io` between 18.822 ms and 18.86 2 ms. The `output_strobe_out` is a free running clock while the `read_write_data_io` is driven at Hi-Z when there is no data transaction.

Figure 26: DQS Write Operation Waveform After Executing the topwave.do File

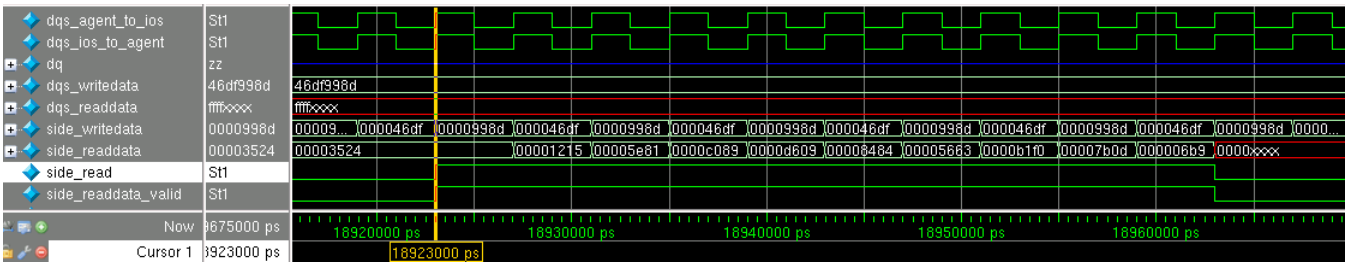


Side Read Operation

The side read operation is performed between 18.923 ms and 18.963 ms. The DQS agent sends data to the DQS driver with the `side_readdata` signal. Data validation is carried out in parallel, by comparing the `side_readdata` signal against the content of the `check_fifo` (data which was written out during the DQS write operation). If there is a mismatch, the software generates an error message.

The following figure shows the waveform for the side read operation.

Figure 27: Side Read Operation Waveform

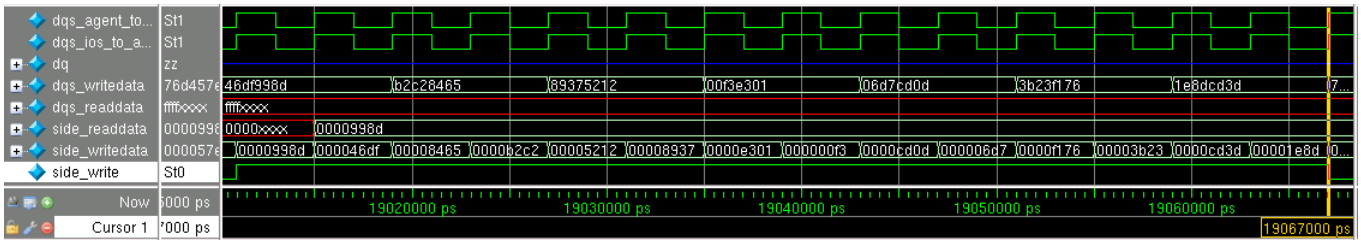


Side Write Operation

The side write operation begins between 19.011 ms and 19.067 ms. The data written out from the DQS driver is also stored in `check_fifo`.

The following figure shows the waveform for the side write operation.

Figure 28: Side Write Operation Waveform



Note: The incoming data at dq is edge-aligned to the dqs_agent_to_ios.

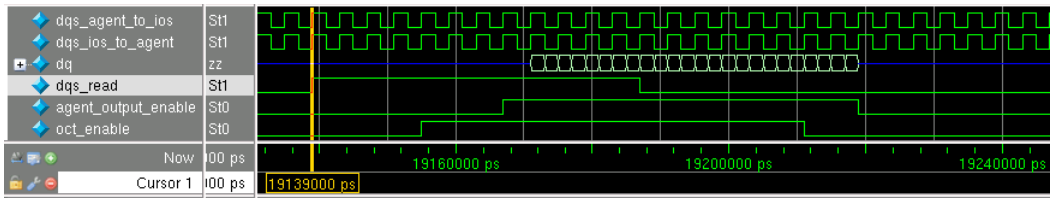
DQS Read Operation

Between 19.227 ms and 19.275 ms when the dqs_readdata_valid signal is held high, data validation is done by comparing data received by the ALTDQ_DQS2 core (read_data_out) against the content in the check_fifo.

ln Stratix V devices, you can choose to enable or disable the hard read FlFO. The read FlFO is in every input data paths. The read FlFO handles full-rate and half-rate conversion only. This example design uses the hard read FlFO. The testbench determines the timing to assert the write enable and read enable ports via the v/i/o_qvld and l/i/o_rden. ln an actual application, you must design your own logic to do so.

The DQS driver begins the DQS read operation when the dqs_read signal is asserted at 19.139 ms, for the entire length of the desired read burst, which in this case is 12 full-rate cycles. The DQS agent also receive the read command, and is ready to send out data. After a specific latency, the agent_output_enable signal is asserted beginning from 19.167 ms to 19.219 ms. During this period, the DQS agent drives clock and data lines of the external memory interface. The oct_enable signal is asserted between 19.155 ms and 19.211 ms. The incoming data (dq) is edge-aligned to the clock (dqs_agent_to_ios).

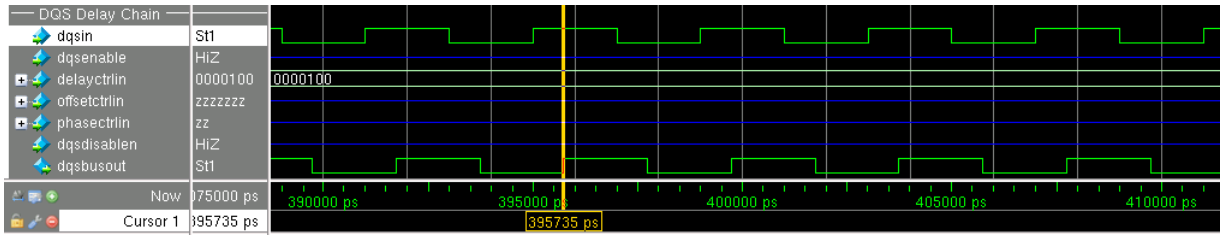
Figure 29: DQS Read Operation Waveform



DQS Delay Chain

Because the DQS enable block is not applicable for this example design, the dqsenable signal is held at Hi-Z at all time. The dqsbusout signal is the delayed dqsin signal that drives to the dedicated DQS clock network to clock the DQ capture registers, so that data are captured at the center of the eye. If you disable the dynamic configuration feature, you should see a 67.5° phase shift (or 735 ps) between dqsin and dqsbusout, as expected due to the settings in **Figure 14**. The following figure shows that the phasectrlin signal is held at Hi-Z.

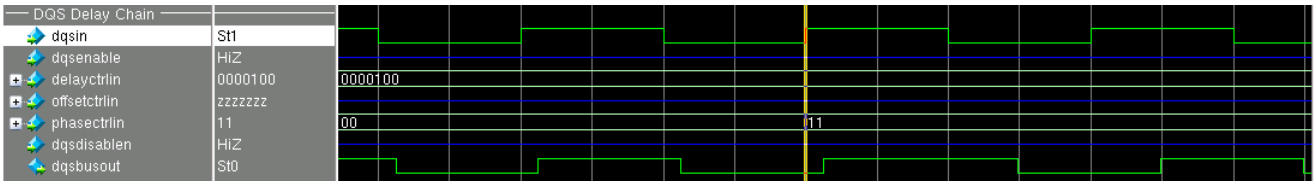
Figure 30: DQS Delay Chain Waveform



However, when you enable the dynamic configuration feature, `phasectrlin` (which is set via the `dqsinputphasesetting` port of the DQS configuration) determines the phase applied to the `dqsbusout` output.

Observe the following waveform. Before dynamic configuration, `phasectrlin` was set to 2'b00 and the shift between `dqsin` and `dqsbusout` is about 245 ps. Meanwhile after performing dynamic configuration, `phasectrlin` was set to 2'b11 and the shift between `dqsin` and `dqsbusout` is about 980 ps. The differences between 2'b11 (135°) and 2'b00 (0°) is 735 ps (67.5°). This is consistent with the settings in **Figure 14** .

Figure 31: DQS Delay Chain Waveform



Hard Read FIFO

As seen in the waveform below, the incoming read data is available on `read_write_data_io` between 19.171 ms and 19.219 ms. The capture DDIO block captures input data (DQ) on the rising and falling edges of the capture clock (DQS). For Stratix V devices, the capture DDIO block feeds the hard read FIFO or bypasses the hard read FIFO and goes directly to the core. The data transfer from the capture DDIO block and the next stage is referred to as zero-cycle transfer. This means that the transfer must happen on the same clock edge.

Note: To ensure correct timing analysis, use the `set_multicycle` SDC command.

This design example uses a hard read FIFO. When the first data is available at 19.174 ms, the `v/i/o_qvld` is asserted. This signal passes through the `write_enable_ctrl` of the DDIO OUT before driving the read enable port of the read FIFO. As the write enable signal of the read FIFO block is asserted, data is written to the read FIFO between 19.174 ms and 19.222 ms . The `l/i/o_rden`, which is connected to the read enable port of the read FIFO block through a fifo_enable block, is then asserted between 19.219 ms and 19.267 ms. Read data is available in the core between 19.179 ms and 19.275 ms on `read_data_out`. You may further optimize the timing to read the read FIFO by adjusting the `lfifo_rden` to create enough space between the read and write pointers in the read FIFO to maximize the throughput.

Figure 32: Hard Read FIFO Implementation

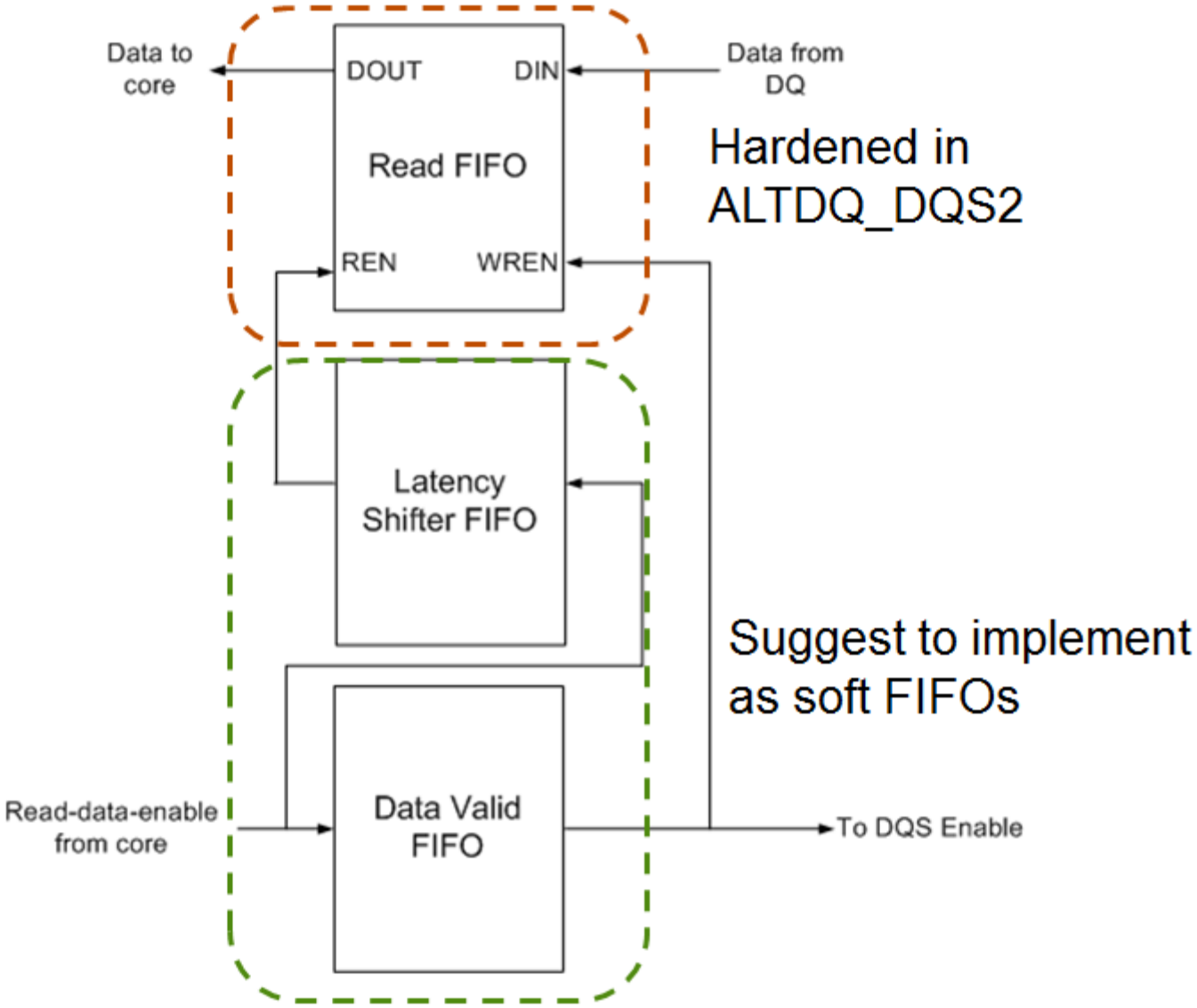
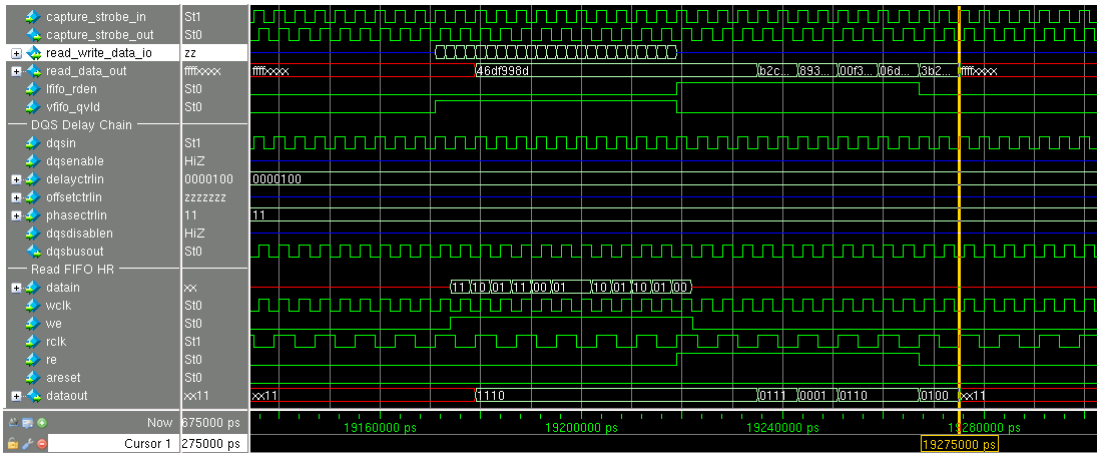


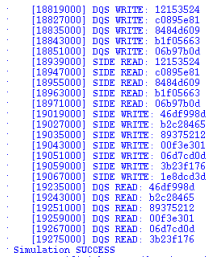
Figure 33: FIFO Read Waveform



Simulation Results

The following figure shows the simulation results in the message panel. lf the simulation failed, it is due to the data sent/received at the ALTDQ_DQS2 is not the same as the expected ones.

Figure 34: Simulation Results



SDC Walkthrough

To create a new **.sdc**, follow these steps:

1. Constrain the clocks coming into the FPGA with a `create_clock` command. The following command creates the base clock for the input clock port driving the PLL:

   ```
   create_clock -name refclk -period 10.000 [get_ports {refclk}]
   ```

2. Create the generated clocks for the PLL with the following command:

   ```
   derive_pll_clocks
   ```

3. Apply inter-clock, intra-clock and I/O interface uncertainties based on timing model characterization using the following command:

   ```
   derive_clock_uncertainty
   ```

4. Constraint the virtual input clock (for incoming DQS strobe) and the `capture_strobe_in` port. ln this example design, it is based on a 250 MHz input clock, with a 50% duty cycle, where the first rising edge occurs at 0 ns.

```
create_clock -name virtual_dqs_in -period 4.000 -waveform {0 2.000}

create_clock -name dqs_in -period 4.000 -waveform {0 2.000}[get_ports
{capture_strobe_in}]
```

Analyzing Same Edge Transfer

The following `set_false_path` commands ensure that you are analyzing only the same edge transfers, by removing the opposite edge transfers. Using multicycle commands ensure that the analysis is done on the correct transfer cycle.

Example 1: set_false_path Commands

```
set_false_path -setup -rise_from [get_clocks {virtual_dqs_in}] -fall_to
[get_clocks {dqs_in}]

set_false_path -setup -fall_from [get_clocks {virtual_dqs_in}] -rise_to
[get_clocks {dqs_in}]

set_false_path -hold -rise_from [get_clocks {virtual_dqs_in}] -rise_to
[get_clocks {dqs_in}]

set_false_path -hold -fall_from [get_clocks {virtual_dqs_in}] -fall_to
[get_clocks {dqs_in}]
```

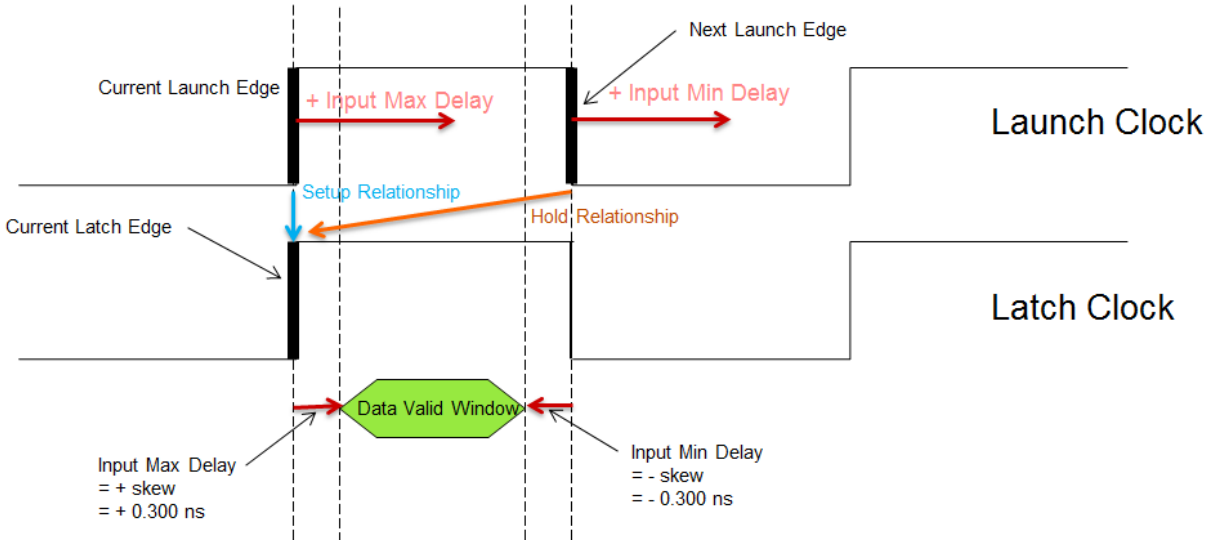
Example 2: set_multicycle_path Commands

```
set_multicycle_path -rise_from [get_clocks {virtual_dqs_in}] -rise_to
[get_clocks {dqs_in}] -setup -end 0

set_multicycle_path -fall_from [get_clocks {virtual_dqs_in}] -fall_to
[get_clocks {dqs_in}] -setup -end 0
```

Example 3: set_input_delay Commands

```
set_input_delay -clock {virtual_dqs_in} -max -add_delay 0.300
[get_ports{read_write_data_io[*]}]

set_input_delay -clock{virtual_dqs_in} -min -add_delay -0.300
[get_ports{read_write_data_io[*]}]

set_input_delay -clock {virtual_dqs_in} -clock_fall -max -add_delay 0.300
[get_ports{read_write_data_io[*]}]

set_input_delay -clock{virtual_dqs_in} -clock_fall -min -add_delay -0.300
[get_ports{read_write_data_io[*]}]
```

Figure 35: Same Edge Transfer Analysis



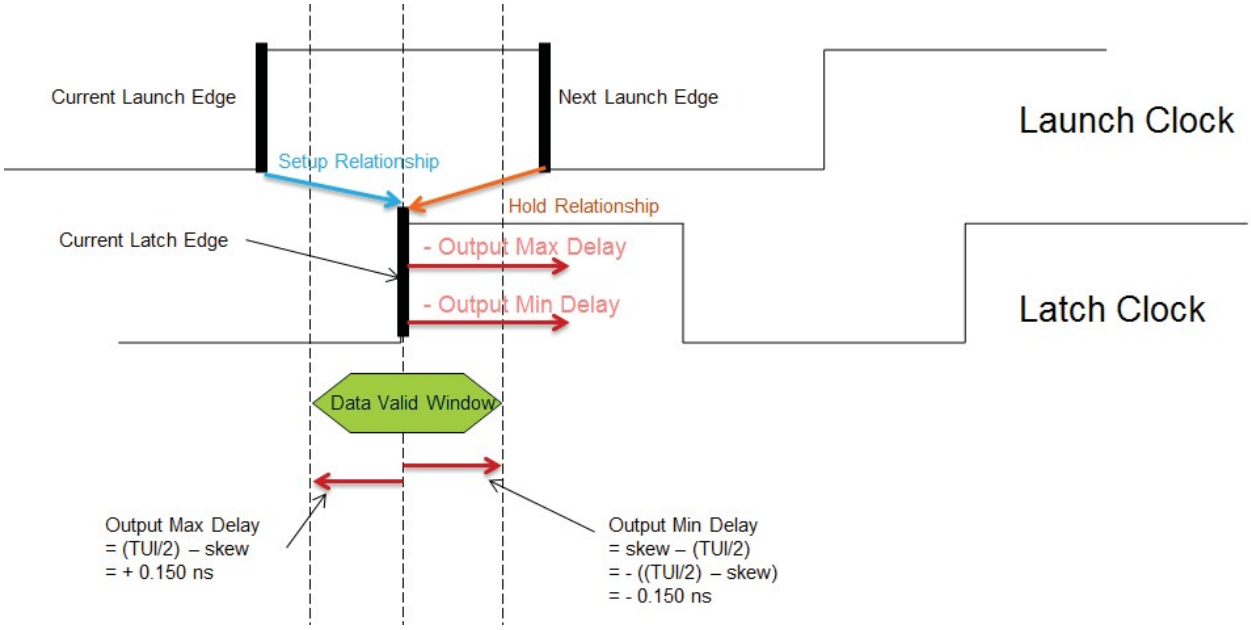
Constraining Outgoing DQS Strobe

The following commands constraint the outgoing DQS strobe. The `create_generated_clock` command requires the `-add` option because the `strobe_io` port is bidirectional. The design sends the data out which centre-aligned to the DQS strobe. The minimum and maximum input delay is assumed to be ±0.25 ns in this reference design. Use the `-add` option in the `set_output_delay` command to add the user-defined clock uncertainty values.

Example 4: Constraining DQS Strobe Commands

```
create_generated_clock -name dqs_out -source [get_pins{pll_inst/alterapll_inst/
altera_pll_i/general[1].gpllPLL_OUTPUT_COUNTER/divclk}] -phase 0
[get_ports{output_strobe_out}]

set_output_delay -clock { dqs_out } -max 0.150 [get_ports
{read_write_data_io[*]}] -add_delay

set_output_delay -clock { dqs_out } -max 0.150 -clock_fall [get_ports
{read_write_data_io[*]}] -add_delay

set_output_delay -clock { dqs_out } -min -0.150 [get_ports
{read_write_data_io[*]}] -add_delay

set_output_delay -clock { dqs_out } -min -0.150 -clock_fall [get_ports
{read_write_data_io[*]}] -add_delay
```

Figure 36:



The following `set_false_path` commands ensure that we are analyzing only the same edge transfers, by removing the opposite edge transfers.

Example 5: set_false_path Commands

```
set_false_path -setup -rise_from [get_clocks{pll_inst/alterapll_inst/
altera_pll_i/general[2].gpllPLL_OUTPUT_COUNTER/divclk}] -fall_to
[get_clocks{dqs_out}]

set_false_path -setup -fall_from [get_clocks{pll_inst/alterapll_inst/
altera_pll_i/general[2].gpllPLL_OUTPUT_COUNTER/divclk}] -rise_to
[get_clocks{dqs_out}]

set_false_path -hold -rise_from [get_clocks{pll_inst/alterapll_inst/
altera_pll_i/general[2].gpllPLL_OUTPUT_COUNTER/divclk}] -rise_to
[get_clocks{dqs_out}]

set_false_path -hold -fall_from [get_clocks{pll_inst/alterapll_inst/
altera_pll_i/general[2].gpllPLL_OUTPUT_COUNTER/divclk}] -fall_to
[get_clocks{dqs_out}]
```

The following `set_multicycle_path` commands ensure that the correct transfer between the DDlO and Read FlFO.

Example 6: set_multicycle_path Commands

```
set_multicycle_path -from {*/altdq_dqs2_stratixv:altdq_dqs2_inst/
input_path_gen[*].capture_regHIGH_DFF} -to {*/
```

```
altdq_dqs2_stratixv:altdq_dqs2_inst/input_path_gen[*].read_fifo_hrINPUT_DFF_*} -
setup -end 0

set_multicycle_path -from {*/altdq_dqs2_stratixv:altdq_dqs2_inst/
input_path_gen[*].capture_regLOW_DFF} -to {*/
altdq_dqs2_stratixv:altdq_dqs2_inst/input_path_gen[*].read_fifo_hrINPUT_DFF_*} -
setup -end 0
```

FIFO control algorithm is necessary. Consider designing some soft FlFOs for this purpose. The following paths can only be set to false path or multicycle if there is calibration algorithm in the system to ensure correct functionalities.

Example 7: set_false_path Commands

```
#set_false_path -from {*/altdq_dqs2_stratixv:altdq_dqs2_inst/
input_path_gen[*].read_fifo_hrWRITE_LOAD_DFF_*} -to {*/
altdq_dqs2_stratixv:altdq_dqs2_inst/
input_path_gen[*].read_fifo_hrREAD_LOAD_DFF_*}
```

Related Information

FIFO Control on page 13
Describes the hard data valid FlFO and hard latency shifter FlFO of the Arria V and Cyclone V devices.

Arria V Design Example

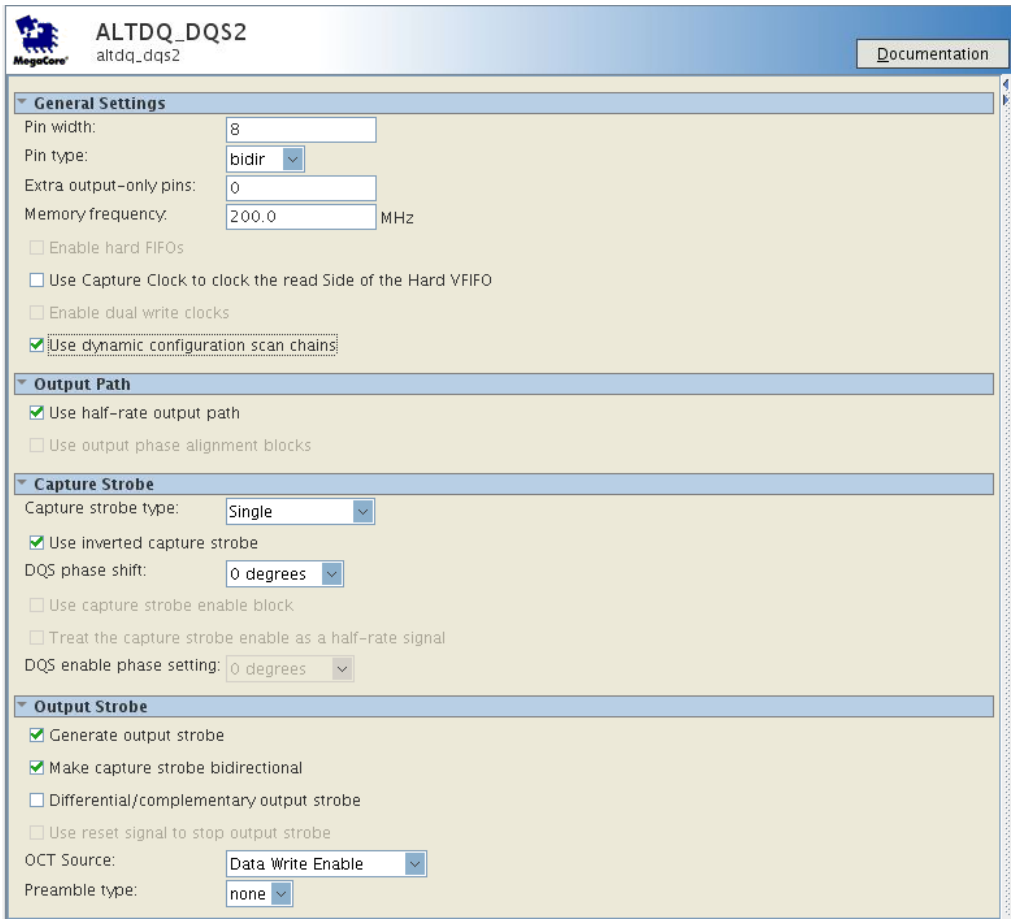
This section describes how to instantiate the ALTDQ_DQS2, ALTERA_PLL, ALTDLL, ALT_OCT IP cores using the **Top_AV_15_1.qar** design example.

Instantiating the ALTDQ_DQS2 IP Core

To instantiate the ALTDQ_DQS2 IP core, perform the following steps:

1. In the Quartus Prime software, open the **Top_AV_15_1.qar** design example and restore the archived file into your working directory.
2. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
3. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the target Altera device family and output file HDL preference. Click **OK**.
4. On the **Parameter Settings** tab, on the **General** page, specify the parameters as shown in the following figure. These parameters configure the general settings for the ALTDQD_DQS2 instance.

Figure 37: ALTDQ_DQS2 Parameter Settings for Arria V Devices



5. Click **Finish**.

Note: If your design requires bidirectional strobe, you must set the DQS phase shift to 0 degree to bypass the DQS delay chain.

Related Information

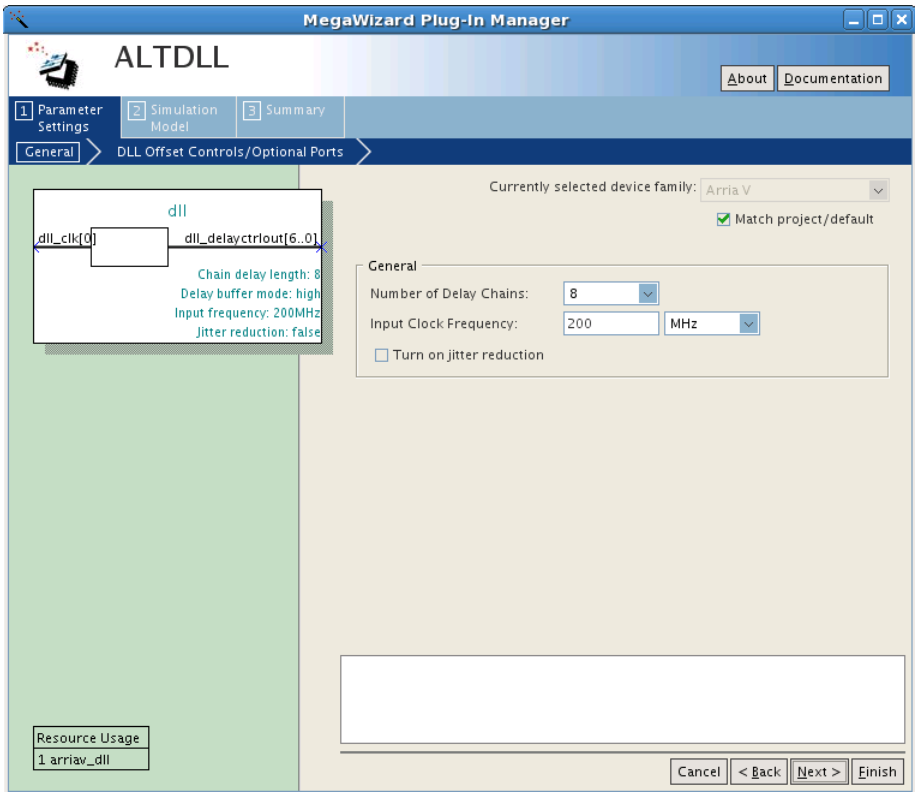
Why do I see random read errors when using the ALTDQ_DQS2 megafunction?
Link to ALTDQ_DQS2 IP core KDB

Instantiating the ALTDLL IP Core

To instantiate the ALTDLL IP core, follow these steps:

1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the target Altera device family and output file HDL preference. Click **OK**.
3. On the **Parameter Settings** tab, on the **General** page, specify the parameters as shown in the following figure.

Figure 38: ALTDLL Parameter Settings

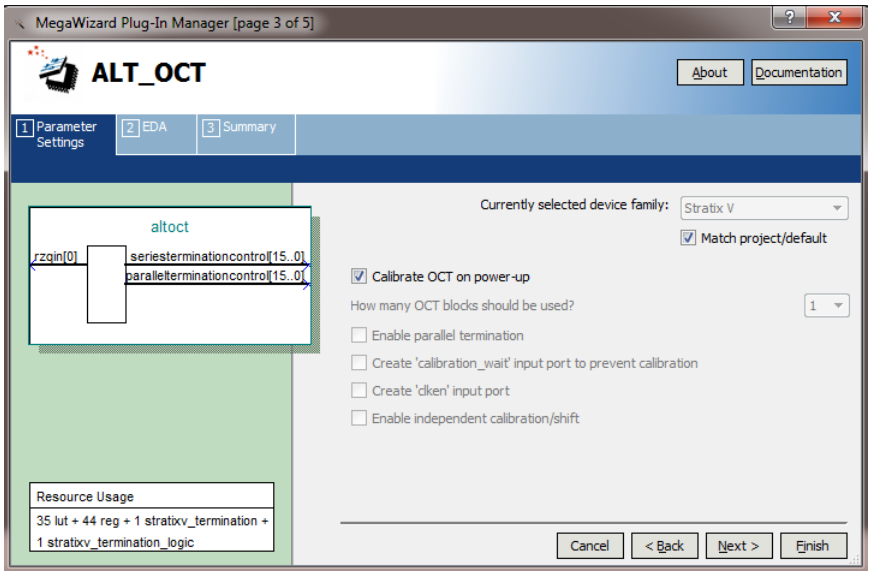


4. Click **Finish**.

Instantiating ALT_OCT IP Core

1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the target Altera device family and output file HDL preference. Click **OK**.
3. On the **Parameter Settings** tab, on the **General** page, specify the parameters as shown in the following figure.

Figure 39: ALT_OCT Parameter Settings Tab

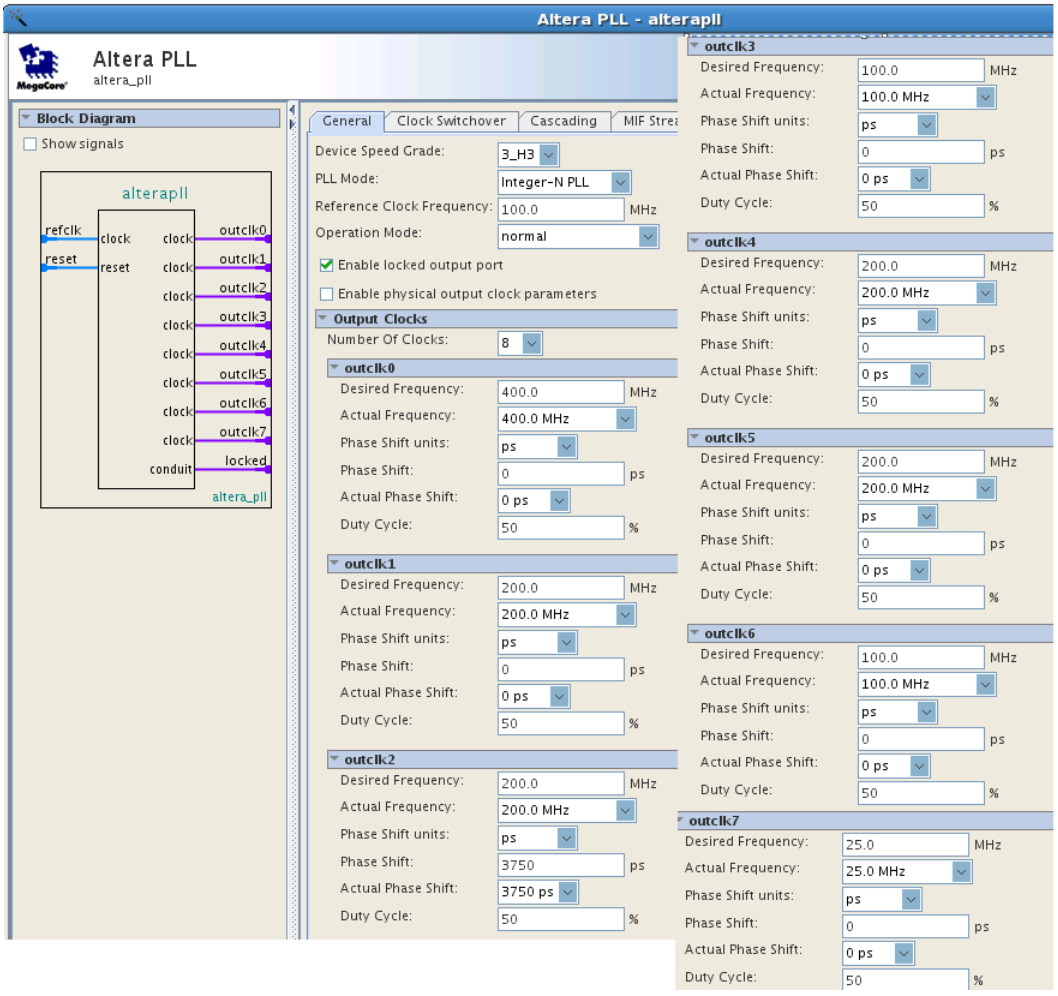


4. Click **Finish**.

Instantiating Altera PLL

1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name for your custom IP variation. This name identifies the IP core variation files in your project. If prompted, also specify the target Altera device family and output file HDL preference. Click **OK**.
3. On the **General** tab, specify the parameters as shown in the following figures.

Figure 40: Altera PLL Parameter Settings for Arria V Devices



4. Click **Finish**.

Altera PLL Clock Settings Information

The following table lists the clock settings Information. You may merge the similar frequency counters in their design, or the Fitter performs the merging automatically.

Table 20: Altera PLL Clock Settings Information

Select **Operation Mode: Normal** in the Altera PLL parameters before configuring the following clocks.

Clock	Description
outclk_0	400 MHz. Used as 2x frequency if necessary.
outclk_1	200 MHz. Used as strobe/dqs clock.
outclk_2	200 MHz. 270° phase shifted. Used as data/dq clock.
outclk_3	100 MHz. Used as half-rate clock.

Clock	Description
outclk_4	200 MHz. Used to drive the ALTDLL IP core. The following are the ALTDLL minimum frequency: • Arria V devices: 200 MHZ • Arria V GZ: 300 MHz
outclk_5	200 MHz. Used to drive the full-rate core clock.
outclk_6	100 MHz. Used to drive the half-rate core clock.
outclk_7	25 MHz. Used as config_clk.

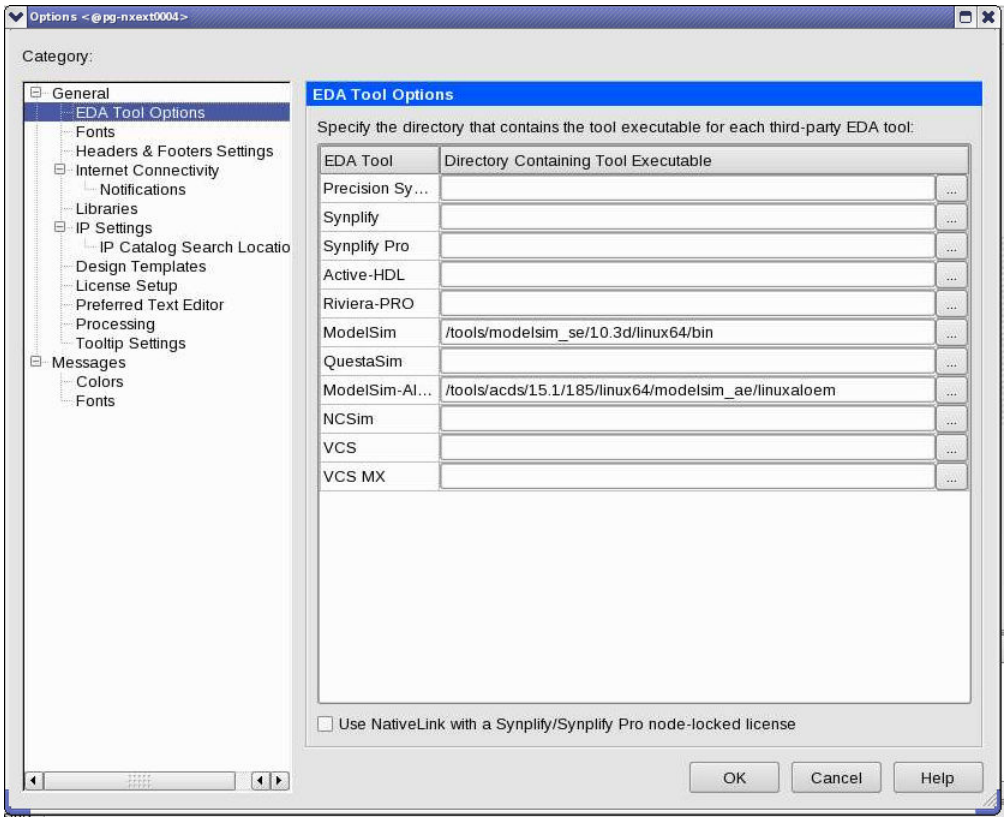
Note: lf the memory frequency is less than the ALTDLL IP core minimum frequency, then drive the ALTDLL IP core at 2x or 4x of the memory frequency. Relatively, the DQS phase settings decrease as well.

Setting Up NativeLink and Simulation Settings

To set up the NativeLink and simulation settings, follow these steps:

1. In the Quartus Prime software, on the Tools menu, select **Options**.
2. In the **Options** dialog box, under Category list, expand General and then select **EDA Tool Options**.
3. In the **EDA Tools Options** window, follow the settings as shown in the following figure:

Figure 41: EDA Tools Options Dialog Box



4. In the Quartus Prime software, on the Assignments menu, click **Settings**.
5. In the Settings dialog box, under the Category list, expand **EDA Tool Settings**. Click **Simulation**.
6. Enter the necessary NativeLink settings. The following figure shows an example settings. In this design example, a testbench (**tb.v**) is provided together with other supporting files.

Figure 42: Simulation Dialog Box

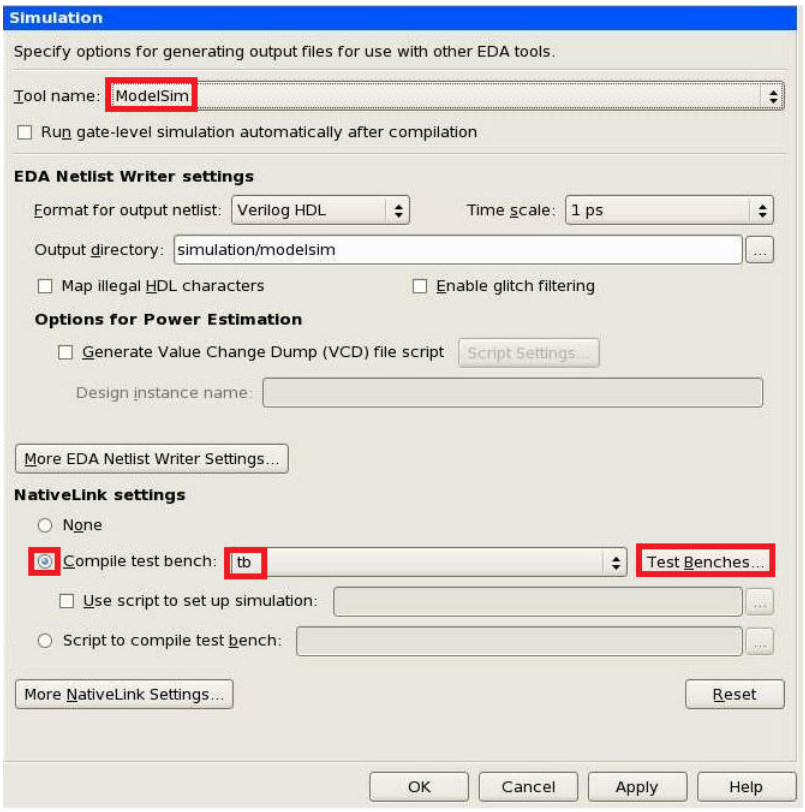


Figure 43: Test Benches Dialog Box

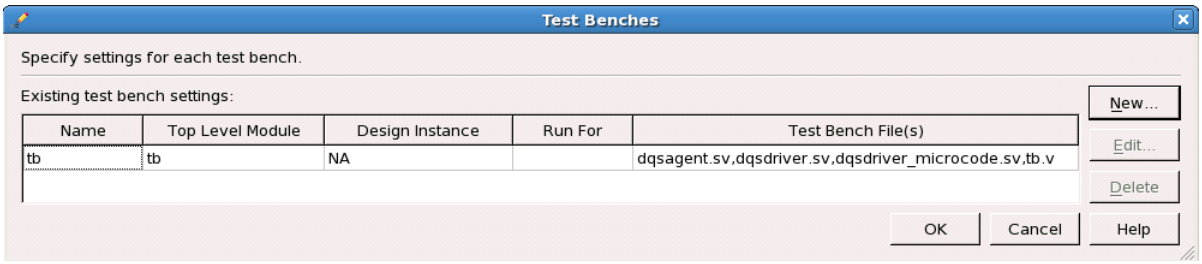
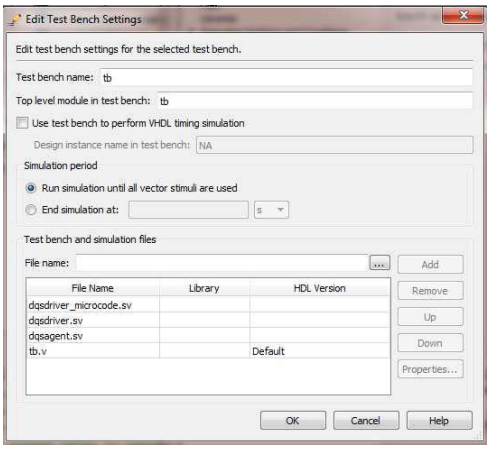


Figure 44: Edit Test Bench Settings Dialog Box



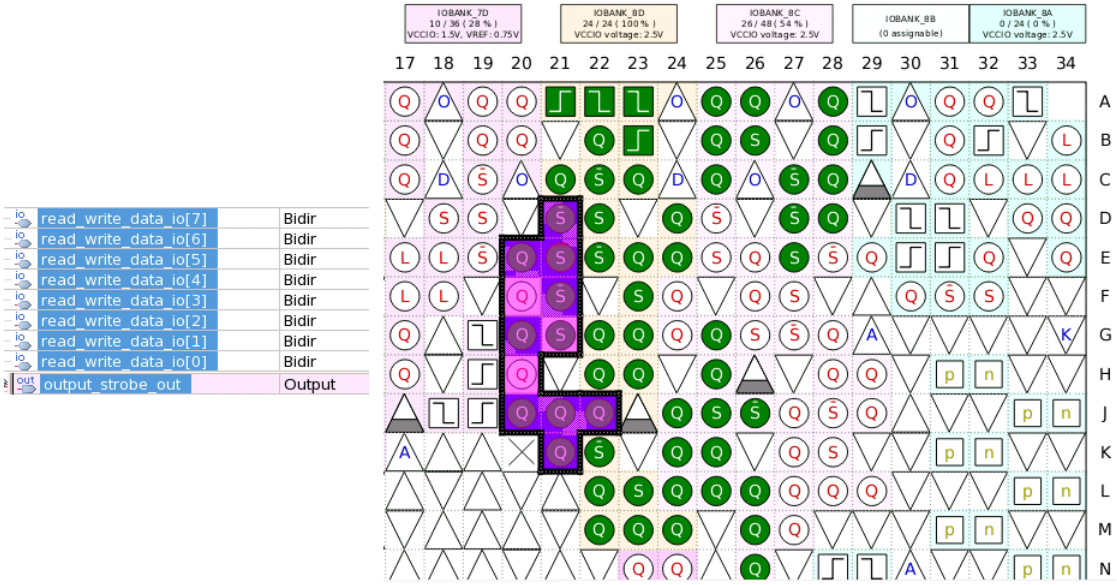
7. Run **Analysis and Synthesis**.
8. To view the simulation results, on the Tools menu, select **Run Simulation Tool** and then click **RTL Simulation.**
   For a successful simulation, you may need to manually change **alterapll.v** to **alterapll.vo** in the auto-generated **top_run_msim_rtl_verilog.do** file.
9. Before running the Fitter, ensure that the following settings are done in the Assignment Editor.
   - I/O Standard
   - Input Termination
   - Output Termination
   - DQ Group
   - Location assignment for strobe pin—this helps the Fitter to fit the related DQ pins in the appropriate I/O sub-banks. You can then back-annotate the locations if desired.

   The following figure shows an example setting in the Assignment Editor and the Pin Planner results:

Figure 45: Assignment Editor Window

	at	From	To	Assignment Name	Value	Enabled	Entity
1	✓		read_write_data_io[*]	I/O Standard	SSTL-15 Class I	Yes	
2	✓		read_write_data_io[*]	Input Termination	Parallel 50 Oh...th Calibration	Yes	top
3	✓		read_write_data_io[*]	Output Termination	Series 50 Ohm...h Calibration	Yes	top
4	✓		*alterapll_0002*|altera_pll:altera_pll_i*|*	PLL Compensation Mode	Normal	Yes	top
5	✓		*alterapll_0002*|altera_pll:altera_pll_i*|*	PLL Automatic Self-Reset	Off	Yes	top
6	✓		*alterapll_0002*|altera_pll:altera_pll_i*|*	PLL Bandwidth Preset	Auto	Yes	top
7	✓		output_strobe_out	Location	PIN_E21	Yes	
8	✓		capture_strobe_in	I/O Standard	SSTL-15 Class I	Yes	
9	✓		output_strobe_out	I/O Standard	SSTL-15 Class I	Yes	
10	✓		capture_strobe_in	Input Termination	Parallel 50 Oh...th Calibration	Yes	top
11	✓		output_strobe_out	Output Termination	Series 50 Ohm...t Calibration	Yes	top
12	✓	output_strobe_out	read_write_data_io[*]	DQ Group	9	Yes	top
13		<<new>>	<<new>>	<<new>>			

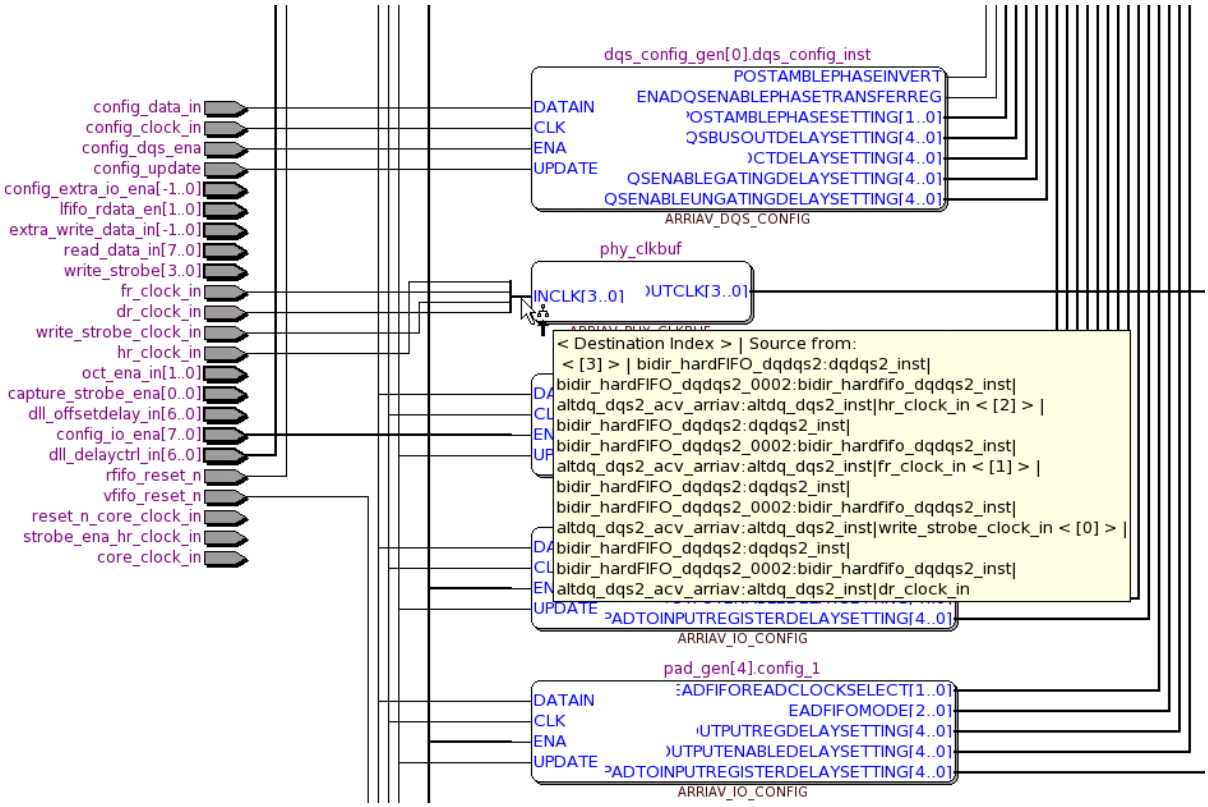
Figure 46: Pin Planner



10. Run the Fitter, Timing Analysis, and Assembler. An SDC example (**top.sdc**) is included in the example design.

Arria V device uses the PHYCLK by default.

Figure 47: Design Example Clock Routing in Arria V Device



Related Information

Understanding Simulation Results—Arria V Design Example on page 83

Understanding Simulation Results—Arria V Design Example

ln the Arria V design example, a generic testbench is used to test the write and read operations in the ALTDQ_DQS2 IP core. The following table lists the components in the testbench.

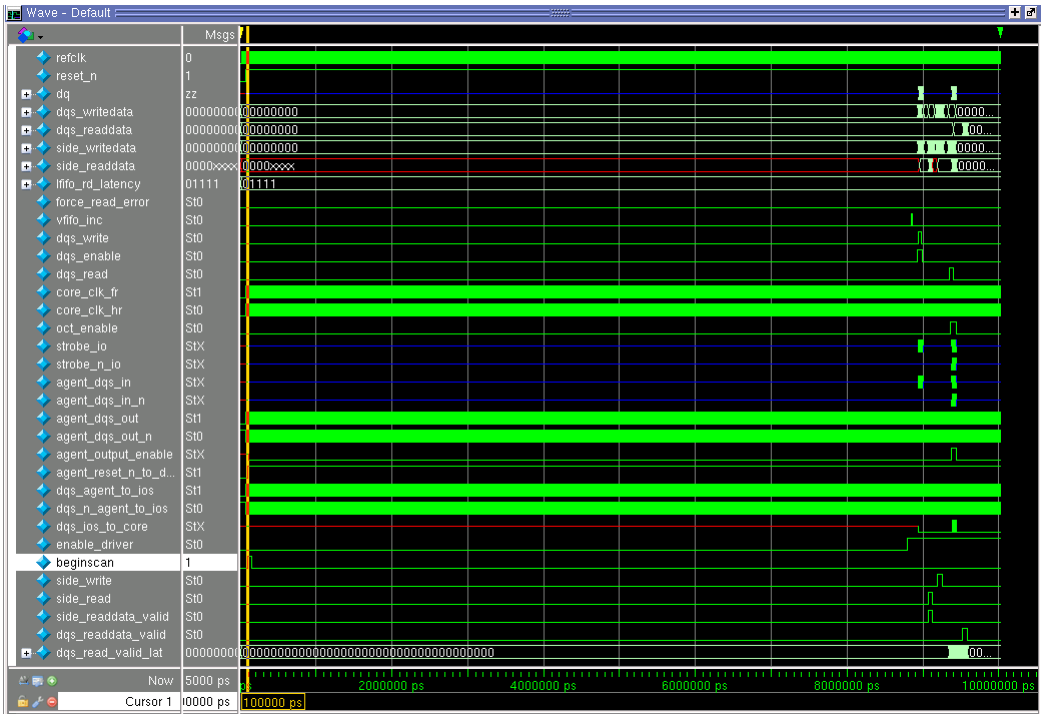
Table 21: Testbench Components

Component	Description
DQS Driver	• Acts as a host controller, sends read/write commands to the ALTDQ_DQS2 IP core. • Compares data read back from the DQSAgent to what it should be • Has a side channel (side reads/writes) communicating directly with the DQS agent, bypassing the ALTDQ_DQS2 IP core. Use the data in the side reads/writes to compare with the data sent to or received from the ALTDQ_DQS2 IP core.
DQS Agent	• Acts as an external memory device. • Has a side channel (side reads/writes) communicating directly with the DQS Driver, bypassing the ALTDQ_DQS2 IP core. Use the data in the side reads/writes to compare with the data sent to or received from the ALTDQ_DQS2 IP core.

Note: Random data is generated and used in the testbench. You may see other data values if you are using different operating system and seeds.

The following figure shows the waveform for the testbench generated after executing the **top_run_msim_rtl_verilog.do** file.

Figure 48: Example Waveform



All ports are in reset mode until the `reset_n` signal is asserted at 70 ns. Then, the `core_clk_fr` and `core_clk_hr` clocks start to toggle. The `agent_reset_n_to_dqs` signal is asserted at 95 ns to reset the ALTDQ_DQS2 IP core located in **top_inst**.

Dynamic Configuration

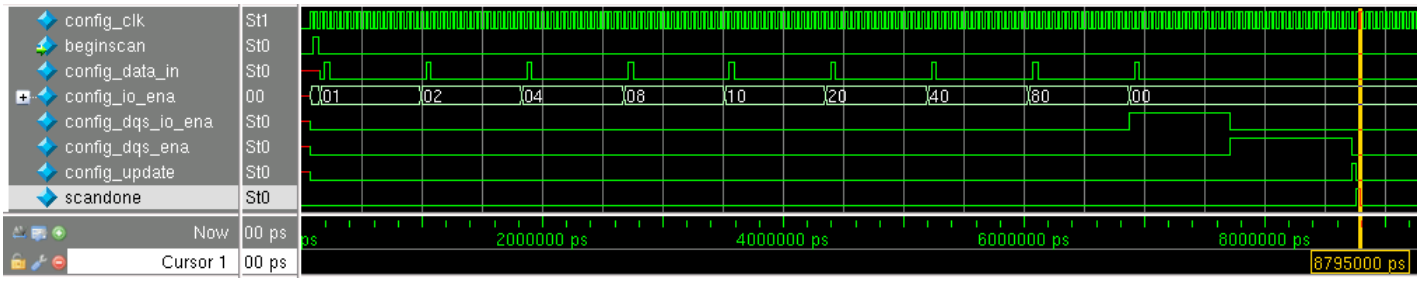
At 100 ns, there is a high pulse on the `beginscan` signal. When the `agent_output_enable` signal is pulled low, the `strobe_io` and `agent_dqs_in` goes to Hi-Z between 110 ns and 8.94 ms time mark, some internal calibration is being carried out. Dynamic configuration is the main feature used. At 8.795 ms time mark, the `enable_driver` signal is asserted. This specifies that the internal calibration is completed, and the control is passed on to the host controller (the DQS driver) to perform the normal read/write operation.

Note: For RTL related to dynamic configuration operations in this design example, refer to **config_controller.sv** file.

You can override the static values at runtime with a scan chain using the dynamic configuration feature in the ALTDQ_DQS2 IP core. To help you achieve static timing closure, the dynamic configuration feature allows you to override the static values at runtime with a scan chain. Each I/O and the DQS logic contain its own scan chain block (shift registers). This section shows you how to serially scan configuration bits into each scan chain block, between 100 ns and 8.795 ms time mark.

The following figure shows the waveform for the dynamic configuration simulation generated after executing **topwave.do** (located in the **simulation/modelsim** folder), between the high pulses of `beginscan` and `scandone`.

Figure 49: Dynamic Configuration Waveform



Because the `enable_driver` signal is asserted at 8.795 ms time mark, following read and write operations will be executed by DQS driver.

Note: The `driver_clk` is running at the same rate as the core

Related Information

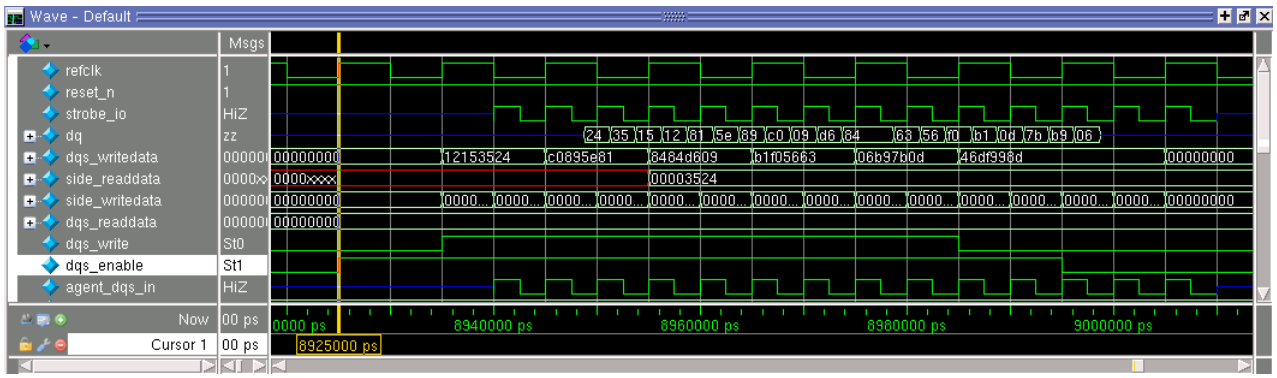
Dynamic Reconfiguration for ALTDQ_DQS2 Megafunction on page 31

DQS Write Operation

The driver asserting the `dqs_enable` signal at 8.925 ms begins the write operation. This will configure the ALTDQ_DQS2's `output_strobe_ena` to high, getting ready to send out strobe to the DQS agent. The `dqs_write` signal is asserted at 8.935 ms. This sets the `write_oe_in` signal of the ALTDQ_DQS2 IP core to high, getting ready to send out data to the DQS agent. The data are written to the `dqs_writedata` of the DQS driver, and then reflected in the `dq` signal of the ALTDQ_DQS2 IP core. The data written out from the DQS driver is also stored in `check_fifo`. During the DQS write operation, the `strobe_io` and `agent_dqs_in` is toggling. `Dqs_write` deasserts at 8.985 ms and `dqs_enable` deasserts at 8.995ms. Outgoing data from the dq signal of the ALTDQ_DQS2 IP core is center-aligned to the strobe (`strobe_io`).

The following figure shows the DQS write operation waveform.

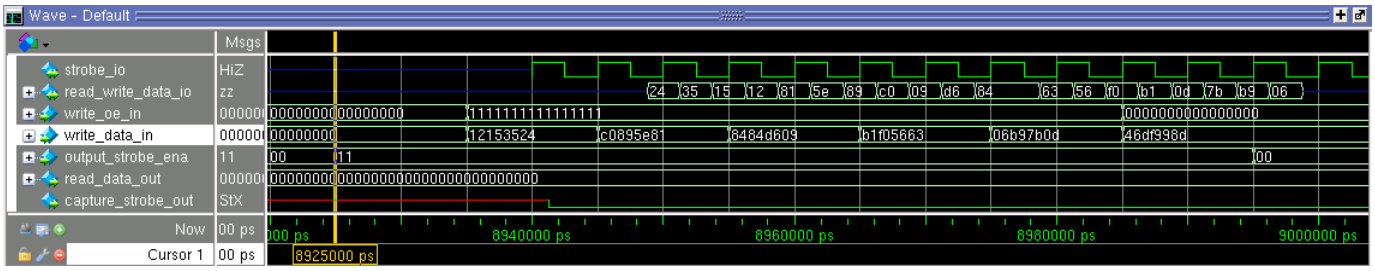
Figure 50: DQS Write Operation Waveform



Note: Before and after the DQS write operation, the `strobe_io` signal is in Hi-Z mode.

The following figure shows the waveform for the dynamic configuration simulation generated after executing **topwave.do** (located in the simulation/modelsim folder). The `output_strobe_ena` is held high from 8.925 ms to 8.995 ms while the `strobe_io` signal starts toggling only between 8.94 ms and 9.01 ms. The `write_oe_in` signal is held high throughout the five sets of valid `write_data_in`, which is between 8.935 ms and 8.955 ms. Center-aligned output data appears on the `read_write_data_io` signal between 8.949 ms and 8.999 ms.

Figure 51: DQS Write Operation Waveform After Executing the topwave.do File

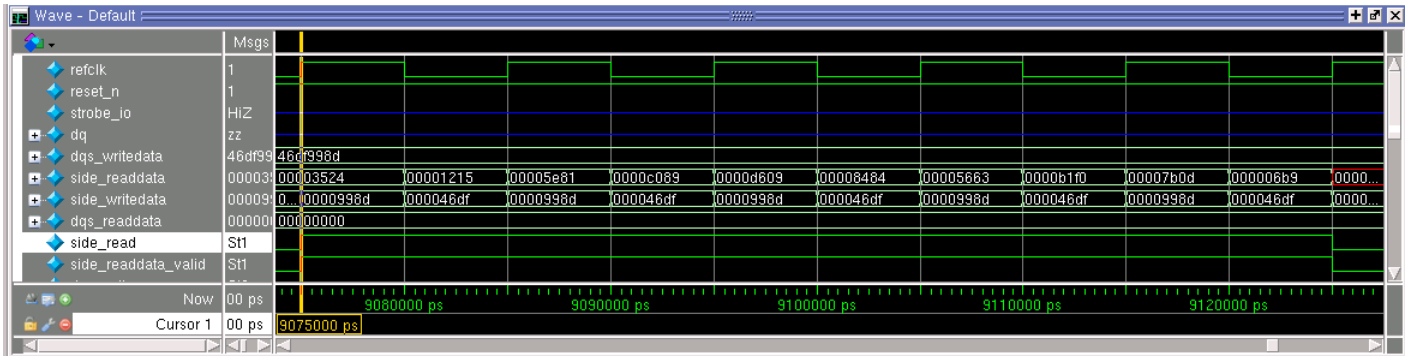


Side Read Operation

The side read operation is performed between 9.075 ms and 9.125 ms, in which the DQS agent sends data to the DQS driver with the `side_readdata` signal. Data validation is carried out in parallel, by comparing the `side_readdata` signal against the content of the `check_fifo` (data which was written out during the DQS write operation). If there is a mismatch, the software generates an error message.

The following figure shows the waveform for the side read operation.

Figure 52: Side Read Operation Waveform

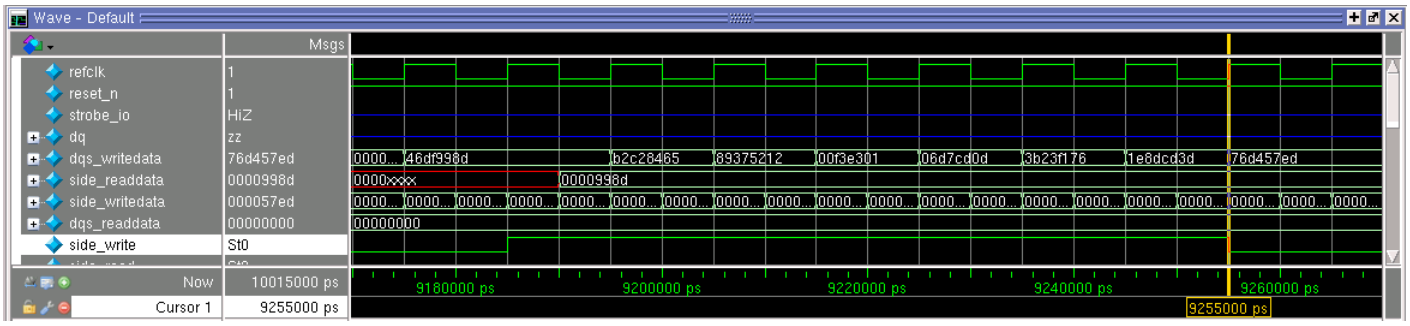


Side Write Operation

The side write operation begins between 9.185 ms and 9.255 ms. The data written out from the DQS driver is also stored in `check_fifo`.

The following figure shows the waveform for the side write operation.

Figure 53: Side Write Operation Waveform



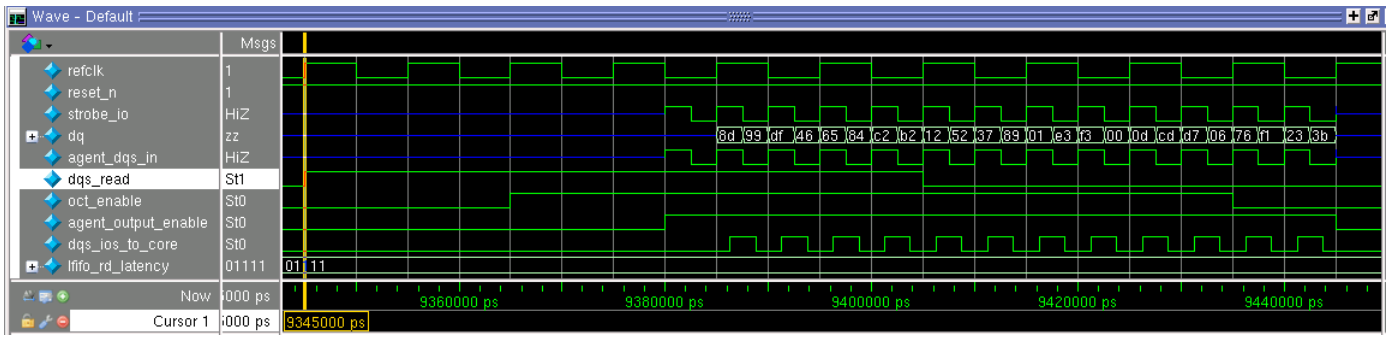
Note: The incoming data at `dq` is edge-aligned to the `strobe_io`.

DQS Read Operation

Data validation is done by comparing data received by the ALTDQ_DQS2 core (`read_data_out`) against the content of the `check_fifo`.

DQS read is initiated by the DQS driver when `dqs_read` is asserted at 9.345 ms. The `lfifo_rdata_en_full` (`dqs_read`) of the ALTDQ_DQS2 IP core is also asserted for the entire length of the desired read burst, which in this case is 12 full-rate cycles. As the DQS agent receive the read command, it is then ready to sends out data as per requested. This is shown with `agent_output_enable` being asserted from 9.380 ms to 9.445 ms. During this period, the DQS agent drives the strobe and data lines of the external memory interface. In this test bench, `strobe_io` is driven by `dqs_agent_to_ios`. When output enable is deasserted, `strobe_io` will be set to Hi-Z.

Figure 54: DQS Read Operation Waveform



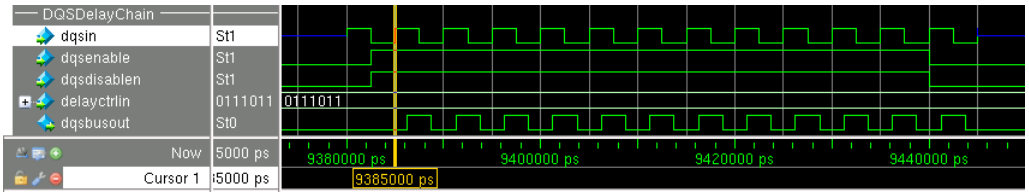
As the incoming data arrives at the ALTDQ_DQS2 IP core, the edge-aligned data on `read_write_data_io`, and strobe on `strobe_io` ports. Following will discuss how the data is captured in the FPGA before it is made available in the core.

DQS Delay Chain

The `dqsenable` signal grounds the DQS input strobe after the strobe goes to Hi-Z. This is important for bidirectional strobes, where glitches can be filtered effectively through the DQS enable. The `dqsbusout` signal is the delayed `dqsin` signal that drives into the dedicated DQS clock network to clock the DQ

capture registers so that data is captured at the centre of the eye. The following figure shows a 90° phase shift between the `dqsin` and `dqsbusout` signals. This is consistent with the settings in **Instantiating the ALTDQ_DQS2 IP Core** on page 73.

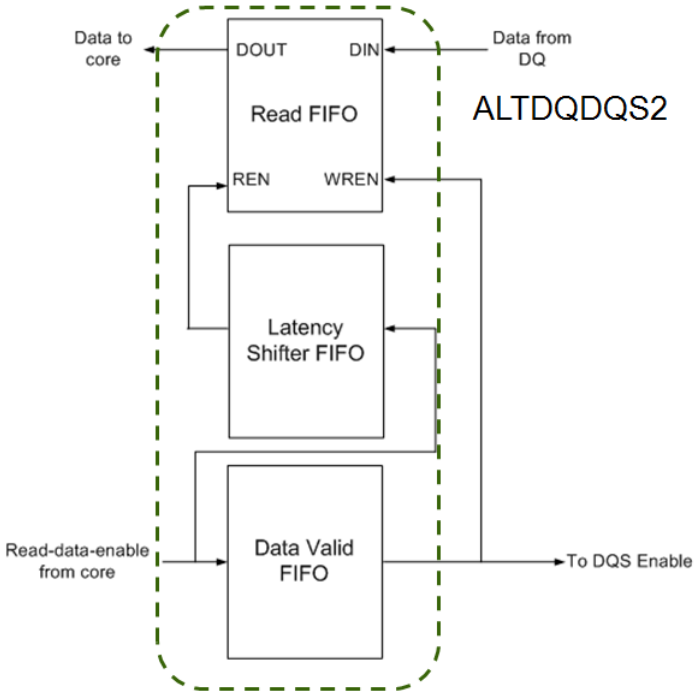
Figure 55: DQS Delay Chain Waveform



VFIFO, LFIFO, and Read FIFO

In Arria V and Cyclone V devices, the data valid FIFO (VFIFO) generates the input signal to the DQS_ENABLE_CTRL block and connects to the write enable port of the read FIFO. The latency shifter FIFO (LFIFO) connects to the read enable port of the read FIFO. The LFIFO and VFIFO implement each configurable latencies to determine the time to read enable and write enable for the Read FIFO respectively. The `lfifo_rd_latency` signal determines the latency setting in the LFIFO while the `vfifo_inc_wr_ptr` signal determines the latency setting for the VFIFO. The read FIFO is in every input data paths, and you can set the read FIFO to control the conversion between FR-FR or FR-HR in Arria V and Cyclone V devices.

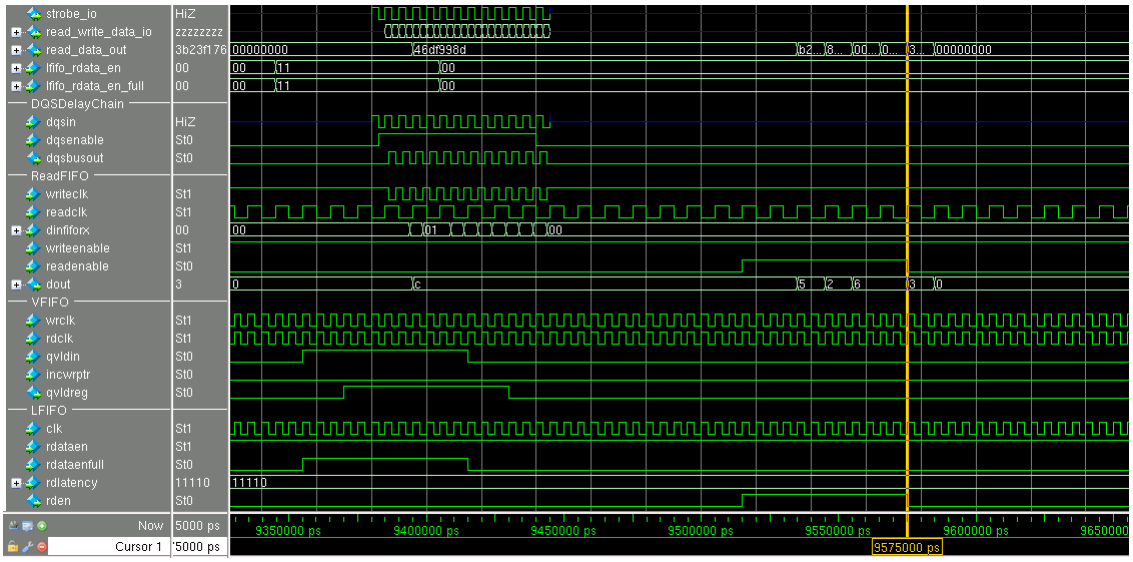
Figure 56: VFIFO, LFIFO and Read FIFO in Arria V and Cyclone V Devices



When the host sends out a read command, a token is also sent to the LFIFO and the VFIFO. This token should be asserted for the length of the desired read burst. This token is referred to `lfifo_rdata_en_full` (for LFIFO) and `vfifo_qvld` (for VFIFO). In this testbench, the `lfifo_rd_latency` is set to 15d. Note that this is consistent with the latency noticed between `rdataenfull` of the LFIFO and `readenable` of the read FIFO, which is 16 half- rate clock cycles. The `lfifo_rdata_en_full` and `vfifo_qvld` signals pass through each HR-FR DDIO before it reaches the `rdataenfull` of the LFIFO and the `qvldin` of the VFIFO at 9.355 ms. The delayed `qvldreg` of the VFIFO (1.5 half-rate cycles) feeds the DQSENABLEIN of the DQS_ENABLE_CTRL block. Meanwhile, the `rden` signal of the LFIFO, which is delayed by 16 half-rate cycles, feeds the READENABLE of the read FIFO.

The `dqsenable` signal is asserted at 9.3825 ms. Edge-aligned input data starts at 9.385 ms and ends at 9.445 ms. The `dqsenable` signal is deasserted at 9.44 ms. In an actual application, you are required to have some form of DQS enable runtime calibration to obtain the correct DQS gating-ungating window. The `strobe_io` signal goes to Hi-Z when the DQS read operation completes. Because the `writeenable` signal of the read FIFO is always asserted, data is written into the read FIFO whenever the read FIFO is available, which is as early as 9.395 ms. However, data is only read out when the `readenable` signal is asserted between 9.515 ms and 9.575 ms. The data are available in the `read_data_out` signal. You can further optimize the timing to read the read FIFO by adjusting the LFIFO latency delay to create sufficient space between the read and write pointers in the read FIFO to maximize the throughput.

Figure 57: FIFO Read Waveform



Note: The write enable (`we`) and read enable (`re`) signals of the hard read FIFO are different from the `wrreq` and `rdreq` signals of the DCFIFO. In DCFIFO, the data are only available at the FIFO output ports when the `rdreq` signal is asserted. In hard read FIFO, the `we` signal controls when to advance the write address counter while the `re` signal controls when to advance the read address counter. When the read and write address pointers are the same, write data appear at the read port as soon as a write operation is completed. This explains why we see the first written data available almost immediately at the FIFO output port. When the signal is asserted, it will advance to the next read address, and then only the second written data is available at the FIFO output port.

Note: Beginning from the Quartus Prime software version 13.1, the `lfifo_rdata_en` and `lfifo_rdata_valid` signals will be removed.

DQS Enable Control

The goal of DQS enable calibration is to find settings that satisfy the following conditions:

- The DQS enable signal rises before the first rising edge of DQS.
- The DQS enable signal sets to 1 after the second-last falling edge of DQS.
- The DQS enable signal falls before the last falling edge of DQS.

The ideal position for the falling edge of the DQS enable signal is centered between the second-last and last falling edges of DQS.

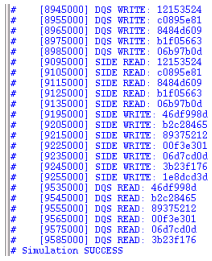
Related Information

Functional Description—UniPHY

Simulation Results

The following figure shows the simulation results in the message panel. lf the simulation failed, it is due to the data sent/received at the ALTDQ_DQS2 is not the same as the expected ones.

Figure 58: Simulation Results



SDC Walkthrough

To create a new **.sdc**, follow these steps:

1. Constrain the clocks coming into the FPGA with the `create_clock` command. The following command creates the base clock for the input clock port driving the PLL:

   ```
   create_clock -name refclk -period 10.000 [get_ports {refclk}]
   ```

2. Create the generated clocks for the PLL with the following command:

   ```
   derive_pll_clocks
   ```

3. Constraint the virtual input clock (for incoming DQS strobe) and the `strobe_in` port. ln this design example, it is based on a 200 MHz input clock, with a 50% duty cycle, where the first rising edge occurs at 0 ns.

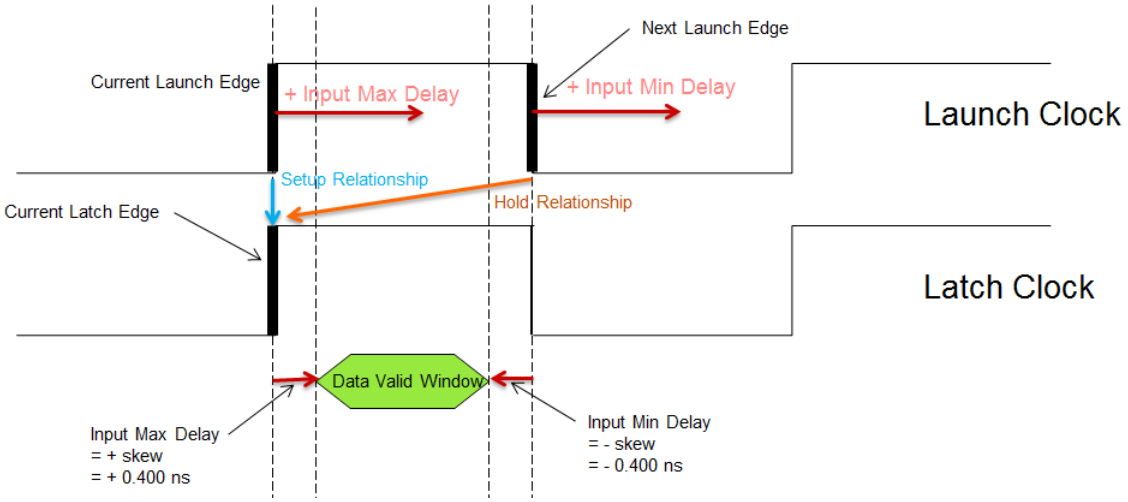
   ```
   create_clock -name virtual_dqs_in -period 5.000

   create_clock -name dqs_in -period 5.000[get_ports {strobe_in}]
   ```

4. lncoming data is edge-aligned to the DQS strobe, and the minimum and maximum input delay is assumed to be ± 0.4 ns in this design example. You must modify constraints to reflect the data and

clock relationship in the system. Use the `-add` option to add the your delay constraint instead of overriding previous constraints.

```
set_input_delay -clock {virtual_dqs_in} -max -add_delay 0.400 [get_ports
{read_write_data_io[*]}]

set_input_delay -clock {virtual_dqs_in} -min -add_delay -0.400
[get_ports{read_write_data_io[*]}]

set_input_delay -clock {virtual_dqs_in} -clock_fall -max -add_delay 0.400 [get_ports
{read_write_data_io[*]}]

set_input_delay -clock {virtual_dqs_in} -clock_fall -min -add_delay -0.400
[get_ports {read_write_data_data_io[*]}]
```

Figure 59: Input Delay Timing Analysis



Analyzing Same Edge Transfer

The following `set_false_path` commands ensure that you are analyzing only the same edge transfers, by removing the opposite edge transfers.

Note: These assignments are optional.

Example 8: set_false_path Commands

```
set_false_path -setup -rise_from [get_clocks {virtual_dqs_in}] -fall_to
[get_clocks {dqs_in}]

set_false_path -setup -fall_from [get_clocks {virtual_dqs_in}] -rise_to
[get_clocks {dqs_in}]

set_false_path -hold -rise_from [get_clocks {virtual_dqs_in}] -rise_to
[get_clocks {dqs_in}]

set_false_path -hold -fall_from [get_clocks {virtual_dqs_in}] -fall_to
[get_clocks {dqs_in}]
```

The default setup relationship is to latch data on the next edge. The following `set_multicycle_path` commands direct the TimeQuest Timing Analyzer to analyze the paths as

a same-edge transfer, whereby the same edge that launches data is going to latch it. The reason it is latched on the same edge is that latch edge will be delayed by the DQS circuitry (hardened 90° in this design example) into the middle of the data eye.

Example 9: set_multicycle_path Commands

```
set_multicycle_path -rise_from [get_clocks {virtual_dqs_in}] -rise_to
[get_clocks {dqs_in}] -setup -end 0

set_multicycle_path -fall_from [get_clocks {virtual_dqs_in}] -fall_to
[get_clocks {dqs_in}] -setup -end 0
```

Constraining Outgoing DQS Strobe

The design sends the data out by a clock shifted 270° so that the non-shifted clock is center-aligned. These constraints state that the external device adds ±250ps of skew, which could also be described as a setup requirement and hold requirement of 250 ps. These numbers are an example, and you must modify constraints to reflect the data and clock relationship in the system. Use the `-add` option to add your delay constraint instead of overriding previous constraints.

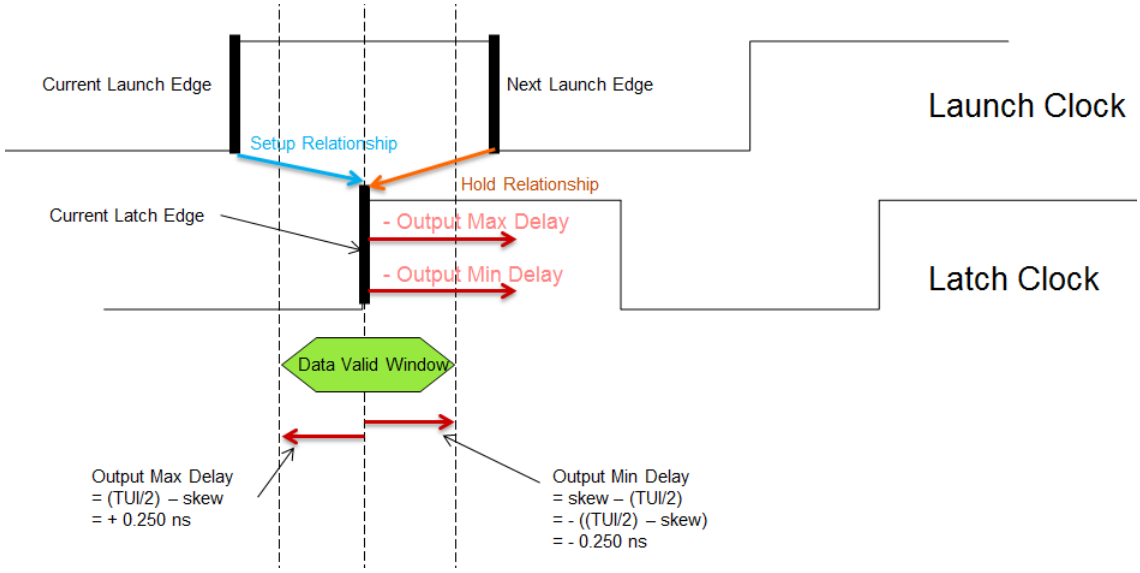
Note: Use the `–add` option for the `create_generated_clock` command for the `strobe_io` port because the port is bidirectional.

The following commands constraint the outgoing DQS strobe.

Example 10: Constraining DQS Strobe Commands

```
create_generated_clock -name dqs_out -source [get_pins{dqdqs2_inst|
bidir_hardfifo_dqdqs2_inst|altdq_dqs2_inst|phy_clkbuf|outclk[1] }] -phase 0
[get_ports {strobe_io}] -add

set_output_delay -clock {dqs_out} -max 0.250 [get_ports {read_write_data_io[*]}]
-add_delay

set_output_delay -clock { dqs_out } -max 0.250 -clock_fall [get_ports
{read_write_data_io[*]}] - add_delay

set_output_delay -clock {dqs_out} -min -0.250 [get_ports
{read_write_data_io[*]}] -add_delay

set_output_delay -clock { dqs_out } -min -0.250 -clock_fall [get_ports
{read_write_data_io[*]}] - add_delay
```

Figure 60: Output Delay Timing Analysis



The following `set_false_path` commands ensure that we are analyzing only the same edge transfers, by removing the opposite edge transfers.

Note: These assignments are optional.

Example 11: set_false_path Commands

```
set_false_path -setup -rise_from [get_clocks{pll_inst|alterapll_inst|
altera_pll_i|general[3].gpll~PLL_OUTPUT_COUNTER|divclk}] -fall_to [get_clocks
{dqs_out}]

set_false_path -setup -fall_from [get_clocks{pll_inst|alterapll_inst|
altera_pll_i|general[3].gpll~PLL_OUTPUT_COUNTER|divclk}] -rise_to [get_clocks
{dqs_out}]

set_false_path -hold -rise_from [get_clocks{pll_inst|alterapll_inst|
altera_pll_i|general[3].gpll~PLL_OUTPUT_COUNTER|divclk}] -rise_to [get_clocks
{dqs_out}]

set_false_path -hold -fall_from [get_clocks{pll_inst|alterapll_inst|
altera_pll_i|general[3].gpll~PLL_OUTPUT_COUNTER|divclk}] -fall_to [get_clocks
{dqs_out}]
```

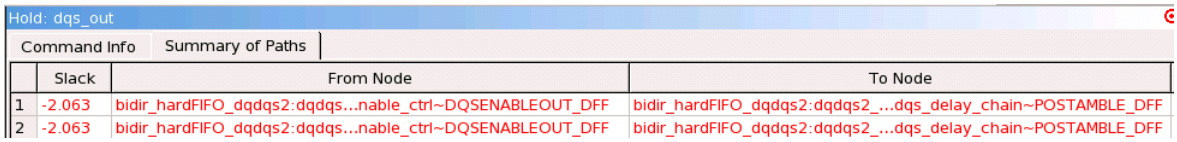
The strobe port functions as input or output at a time. Non-related transfers below should be set to false path and do not need to be analyzed.

```
set_false_path -from [get_clocks {virtual_dqs_in}] -to [get_clocks {dqs_out}]
```

Timing Violation

The following figure shows a timing violation in the example design. This path is related to DQS enable control and is valid. Some calibration algorithm is required to control the DQS enable block.

Figure 61: Timing Violation



Without any calibration algorithm in place, this path cannot be set as false path in the static timing analysis.

Example 12: set_false_path Command

```
#set_false_path -from [get_keepers {*|dqs_enable_ctrl~DQSENABLEOUT_DFF}] -to
[get_clocks{dqs_out}]
```

Related Information

- **External Memory Interfaces in Arria V Devices**

  For more information related to the PHY Clock and DQS Logic Blocks.
- **Functional Description—UniPHY**

  For more information on some basic calibration.

tq_analysis.tcl

The **tq_analysis.tcl** is a script that analyzes specific dqdqs I/O timing paths. Because you might be changing the l/O constraints for your specific implementation, this TCL script helps you to quickly run specific timing analysis.

IP-Generate Command

You can use the `ip-generate`command to create custom variations of the ALTDQ_DQS2 IP core.

Using IP-Generate Command

You can use **ip-generate.exe**, a command-line executable, to configure parameters. The command creates or modifies custom IP core variations, which you can then instantiate in a design file.

To run the `ip-generate` command, follow these steps:

1. Locate the **ip-generate.exe** executable file in the *<quartus_install_dir>***quartus\sopc_builder\bin** folder. Ensure that you include the directory path into your operating system environment.
2. To obtain the options for the `ip-generate` command, type the following command in the command prompt:

```
ip-generate –help
```

3. To instantiate the IP core using the executable file, type the following syntax:

```
ip-generate --component-name=altdq_dqs2 –-component-system-
param=DEVICE_FAMILY="Stratix V" –-file-set=QUARTUS_SYNTH --output-
name[=<file_name>] –-component- param[=<parameter_name>][=<parameter_value>]
```

The *<file_name>* is the instance name. For example, `my_dqdqs2`. The *<parameter_name>* is the name of the parameter that you configure with the value stated in *<parameter_value>*.

You must use the `--component-param[=<parameter_name>][=<parameter_value>]` option for every parameter assignment. Parameters that are not assigned take the default values. Ensure that you type the exact case and spaces for the parameter names and values.

The following list describes two examples of how you can use this command:

- To create an ALTDQ_DQS2 instance named `my_dqdqs2` with all the default parameter values, type the following command:

```
ip-generate --component-name=altdq_dqs2 –-component-system-
param=DEVICE_FAMILY="Stratix V" --output-name=my_dqdqs2 –-file-set=QUARTUS_SYNTH
```

- To create an ALTDQ_DQS2 instance named `mydq_dqs2` without output strobe ports, and with a 135° DQS phase shift, type the following command:

```
ip-generate --component-name=altdq_dqs2 –-component-system-
param=DEVICE_FAMILY="Stratix V" --output-name=my_dqdqs2 –-file-set=QUARTUS_SYNTH
--component-param=USE_OUTPUT_STROBE="False" --component-
param=DQS_PHASE_SETTING="3"
```

This command generates two files—**my_dqdqs2.v** and **my_dqdqs2_altdq_dqs2.sv**. The **my_dqdqs2.v** file contains the `my_dqdqs2` top level module, and the **my_dqdqs2_altdq_dqs2.sv** file contains the source code. The files are in Verilog HDL format.

The `ip-generate` command generates the ports for the instance based on the parameter values.

Related Information

- **ALTDQ_DQS2 Ports** on page 17
- **ALTDQ_DQS2 Parameter Settings** on page 2

ALTDQ_DQS2 IP Core User Guide Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
16.1	ALTDQ_DQS2 IP Core User Guide
16.0	ALTDQ_DQS2 IP Core User Guide
15.1	ALTDQ_DQS2 IP Core User Guide
14.1	ALTDQ_DQS2 IP Core User Guide

Document Revision History

Table 22: Document Revision History

Date	Version	Changes
May 2017	2017.05.08	• Updated the desciption for DQS enable block in Blocks in DQ and DQS Input Path table. • Updated the description for hard data valid FIFO in Capture DDIO to Read FIFO Path section. • Updated description for `config_data` port in ALTDQ_DQS2 Dynamic Configuration Ports table. • Updated the description for `config_data` port in Dynamic Reconfiguration for ALTDQ_DQS2 section. • Changed instances of *Quartus II* to *Quartus Prime*.
February 2017	2017.02.24	Updated Stratix V and Arria V design examples for Quartus II version 15.1.
May 2016	2016.05.02	• Updated Cyclone V devices supports only x8/x9 DQ/DQS groups and the maximum number of pins including strobes is 12. • Added ALTDQ_DQS2 IP Core User Guide Archives table.
November 2015	2015.11.25	• Clarified the input ordering for Stratix V GX, Arria V GZ, Arria V GX, Arria V SoC, Cyclone V GX, and Cyclone V SoC devices to `config_data` port description.
November 2015	2015.11.06	• Added a note to **Pin Width** in ALTDQ_DQS2 Parameter Settings table to clarify that Cyclone V devices support maximum pin width of 10. • Changed instances of *Quartus II* to *Quartus Prime*. • Removed topics on generating IP cores and added links to Introduction to Altera IP Cores, Creating Version-Independent IP and Qsys Simulation Scripts, and Project Management Best Practices.
December 2014	2014.12.17	Updated the description for `write_strobe_clock_in` signal to explain that the signal is a full-rate input clock when you set the IP type to **Input** for Arria V and Cyclone V devices.

Date	Version	Changes
July 2014	2014.07.07	• Replaced MegaWizard Plug-In Manager information with IP Catalog. • Added standard information about upgrading IP cores. • Added standard installation and licensing information. • Clarified that configuration data must be input in LSB first ordering and updated the typical frequency for `config_clock_in` from 50 MHz to 25 MHz in the **Dynamic Reconfiguration for ALTDQ_DQS2** on page 31 section. • Updated vfifo_qvld[] in **Table 8** to clarify that this signal is also supported in Stratix V devices. Also clarified that the this signal is 1-bit wide for Stratix V devices and 2-bits wide for Arria V and Cyclone V devices. • Updated **Table 10**, **Table 12**, and **Figure 11** to update IOE delay settings information. • Removed lfifo_rdata_en and lfifo_rdata_valid ports information from **Figure 8** and **Table 8**. • Added information about checking the validity of the data coming from the read FIFO in **FIFO Control** on page 13. • Added **Arria V Design Example** on page 73 and **Stratix V Design Example** on page 54 sections. • Updated **DQS Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices** on page 36 and **I/O Configuration Block Bit Sequence for Arria V GZ and Stratix V Devices** on page 34 to include Arria V GZ devices. • Clarified that the ALTDQ_DQS2 does not support DQS tracking.
January 2013	2.2	• Updated note in Figure 3–8 on page 3–10. • Added "Additional Information" section.
December 2012	2.1	• Replaced previous design examples with the following design examples: • For Arria V devices: 12.1_AV_BasicDesign.qar • For Stratix V devices: 12.1_SV_BasicDesign.qar • Made necessary changes in the document to reflect the design examples replacement.

Date	Version	Changes
December 2012	2.0	• Major enhancement to include: • Arria V and Cyclone V devices information. • Updated "Features" on page 1–1: • Included read FIFO, hard FIFO, latency shifter FIFO, and data valid FIFO. • "Device Support" on page 1–2 • Included Arria V and Cyclone V devices. • Updated "Parameter Settings" on page 2–1: • Updated Table 2–1 on page 2–1 to include new parameters and to update old parameters. • Updated "ALTDQ_DQS2 Datapaths" on page 3–1: • Updated Figure 3–1 and added notes *(3)*, *(4)*, and *(5)* to clarify the usage of soft and hard FIFO for different devices and to explain the location of an inversion. • Added the following new sections: "DQS Logic" on page 3–2, "Capture DDIO to Read FIFO Path" on page 3–3, and "FIFO Control" on page 3–4. • Updated "ALTDQ_DQS2 Ports" on page 3–10: • Updated "DQ and DQS Output Path" on page 3–6 to include DQS output path information. • Updated Figure 3–5 on page 3–6 and added notes to the figure to help distinguish the device support. • Updated Figure 3–6 on page 3–7 to fix the variable typo and to clarify that the figure is for additional pin usage for Stratix V devices. • Combined and converted IP cores information to Table 3–3 on page 3–8 for ease of reference. • Added Figure 3–7 on page 3–8 to shows the DQ and DQS output path for Arria V and Cyclone V devices.
December 2012	2.0	• Updated "ALTDQ_DQS2 Ports" on page 3–10: • Major update to Figure 3–8 on page 3–10 to clearly define the device family support and the port types. • Updated Table 3–3 on page 3–8 to include new ALTDQ_DQS2 data strobe ports and updated old ALTDQ_DQS2 ports. • Updated Table 3–5 on page 3–12 to include new ALTDQ_DQS2 data ports and updated old ALTDQ_DQS2 data ports. • Updated Table 3–7 on page 3–14 to update the description of the ALTDQ_DQS2 PLL and DLL ports • Added Table 3–8 on page 3–14 to introduce the new ALTDQ_DQS2 hard FIFO ports. • Updated Table 3–9 on page 3–15 to add new ALTDQ_DQS2 dynamic configuration ports and to update old ports. • Added new chapter: "Dynamic Reconfiguration for ALTDQ_DQS2 Megafunction" on page 4–1. • Added new chapter: "Instantiating Megafunctions" on page 5–1 • Added design examples.

Date	Version	Changes
September 2010	1.0	Initial release.