LatticeECP2/M High-Speed I/O Interface

June 2013 Technical Note TN1105

Introduction

LatticeECP2™ and LatticeECP2M™ devices support Double Data Rate (DDR) and Single Data Rate (SDR) interfaces using the logic built into the Programmable I/O (PIO). SDR applications capture data on one edge of a clock while the DDR interfaces capture data on both the rising and falling edges of the clock, thus doubling performance. The LatticeECP2/M I/Os also have dedicated circuitry to support DDR and DDR2 SDRAM memory interfaces. This technical note details the use of LatticeECP2/M devices to implement both a high-speed generic DDR interface and DDR and DDR2 memory interfaces.

DDR and DDR2 SDRAM Interfaces Overview

A DDR SDRAM interface will transfer data at both the rising and falling edges of the clock. The DDR2 is the second generation of the DDR SRDRAM memory.

The DDR and DDR2 SDRAM interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. The DDR SDRAM interface uses a single-ended DQS strobe signal, whereas the DDR2 interface uses a differential DQS strobe. Figures 12-1 and 12-2 show typical DDR and DDR2 SDRAM interface signals. SDRAM interfaces are typically implemented with eight DQ data bits per DQS. An x16 interface will use two DQS signals and each DQS is associated with eight DQ bits. Both the DQ and DQS are bi-directional ports used to both read and write to the memory.

When reading data from the external memory device, data coming into the device is edge-aligned with respect to the DQS signal. This DQS strobe signal needs to be phase-shifted 90 degrees before FPGA logic can sample the read data. When writing to a DDR/DDR2 SDRAM, the memory controller (FPGA) must shift the DQS by 90 degrees to center-align with the data signals (DQ). A clock signal is also provided to the memory. This clock is provided as a differential clock (CLKP and CLKN) to minimize duty cycle variations. The memory also uses these clock signals to generate the DQS signal during a read via a DLL inside the memory. Figures 12-3 and 12-4 show DQ and DQS timing relationships for read and write cycles. For other detailed timing requirements, please refer to the DDR SDRAM JEDEC specification (JESD79C).

During read, the DQS signal is LOW for some duration after it comes out of tristate. This state is called Preamble. The state when the DQS is LOW before it goes into Tristate is the Postamble state. This is the state after the last valid data transition.

DDR SDRAM also requires a Data Mask (DM) signal to mask data bits during write cycles. Note that the ratio of DQS to data bits is independent of the overall width of the memory. An 8-bit interface will have one strobe signal.

DDR SDRAM interfaces use the SSTL25 Class I/II I/O standards whereas the DDR2 SDRAM interface uses the SSTL18 Class I/II I/O standards. The DDR2 SDRAM interface also supports differential DQS (DQS and DQS#).

Figure 12-1. Typical DDR SDRAM Interface

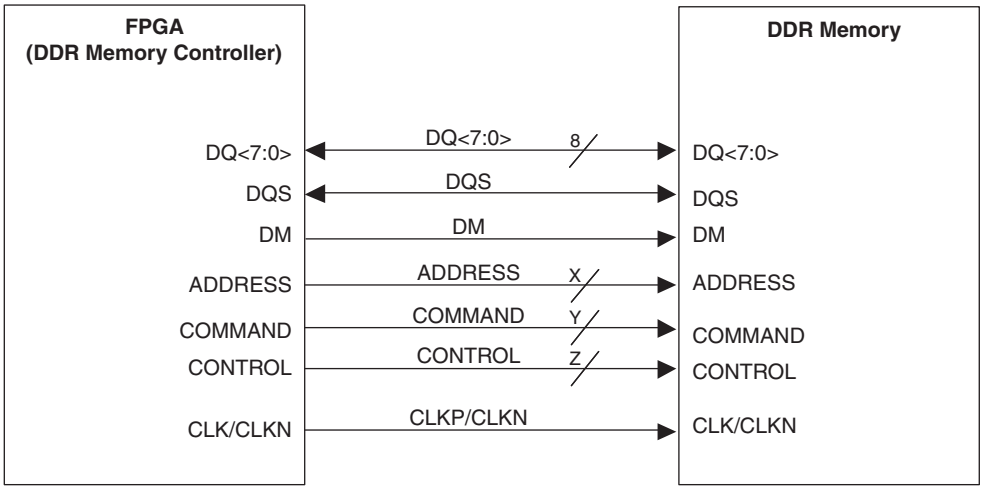
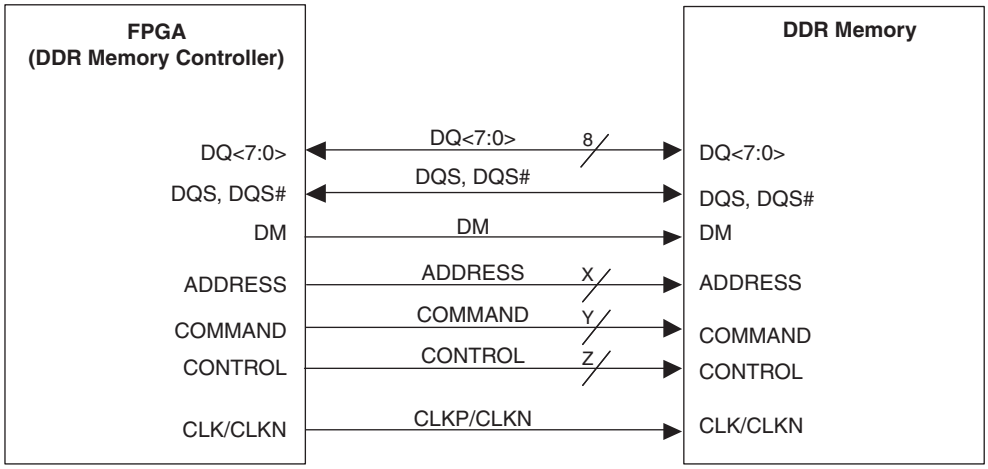


Figure 12-2. Typical DDR2 SDRAM Interface



The following two figures show the DQ and DQS relationship for memory read and write interfaces.

Figure 12-3. DQ-DQS During READ

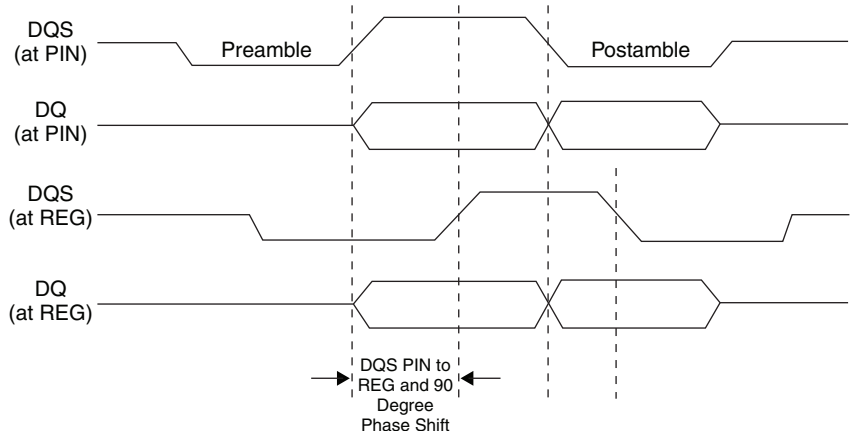
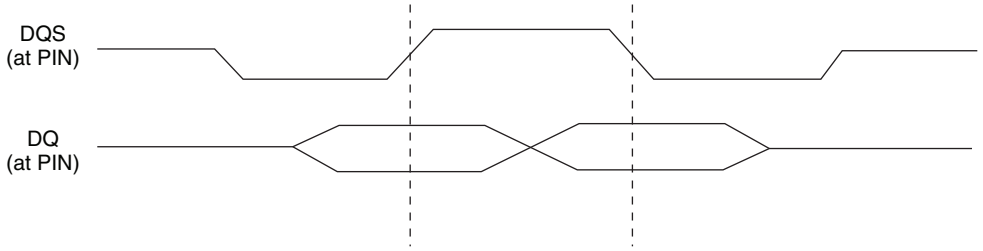


Figure 12-4. DQ-DQS During WRITE



Implementing DDR Memory Interfaces with LatticeECP2/M Devices

As described in the DDRSDRAM overview section, the DDR SDRAM interfaces rely primarily on the use of a data strobe signal called DQS for high-speed operation. When reading data from the external memory device, data coming into the LatticeECP2/M device is edge-aligned with respect to the DQS signal. Therefore, the LatticeECP2/M device needs to shift the DQS (a 90-degree phase shift) before using it to sample the read data. When writing to a DDR SDRAM, the memory controller from the LatticeECP2/M device must generate a DQS signal that is center-aligned with the DQ, the data signals. This is accomplished by ensuring the DQS strobe is 90 degrees ahead relative to DQ data.

LatticeECP2/M devices have dedicated DQS support circuitry for generating the appropriate phase shifting for DQS. The DQS phase shift circuit uses a frequency reference DLL to generate delay control signals associated with each of the dedicated DQS pins and is designed to compensate for process, voltage and temperature (PVT) variations. The frequency reference is provided through one of the global clock pins.

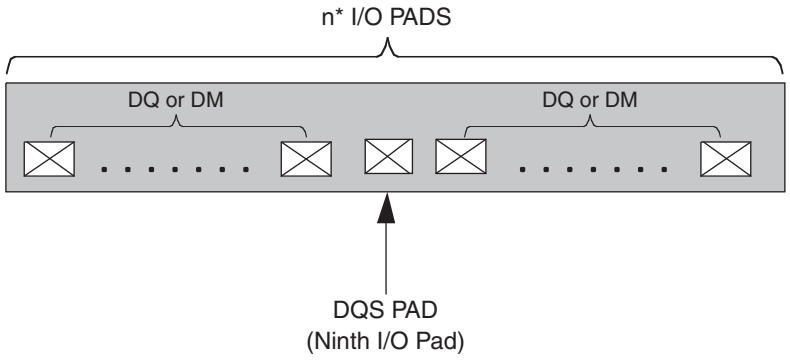
The dedicated DDR support circuit is also designed to provide comfortable and consistent margins for data sampling window.

This section describes how to implement the read and write sections of a DDR memory interface. It also provides details of the DQ and DQS grouping rules associated with the LatticeECP2/M devices.

DQS Grouping

Each DQS group generally consists of at least 10 I/Os (one DQS, eight DQ and one DM) to implement a complete 8-bit DDR memory interface. LatticeECP2/M devices support DQS signals on the bottom, left and right sides of the device. Each DQS signal on the bottom half of the device will span across 18 I/Os and on the left and right sides of the device will span across 16 I/Os. Any 10 of these I/Os spanned by the DQS can be used to implement an 8-bit DDR memory interface.

Figure 12-5. DQ-DQS Grouping



*n=18 on bottom banks and n=16 on the left and right side banks.

Figure 12-5 shows a typical DQ-DQS group for a LatticeECP2/M device. The ninth I/O of this group of 16 I/Os (on the left and right side banks) and 18 I/Os (on the bottom bank) is the dedicated DQS pin. All eight pads before of the DQS and 7 (on the left and right side) and 9 (on the bottom bank) pads after the DQS are covered by this DQS bus span. The user can assign any eight of these I/O pads to be DQ data pins. Therefore, to implement a 32-bit wide memory interface you would need to use four such DQ-DQS groups.

When not interfacing with the memory, the dedicated DQS pin can be used as a general purpose I/O. Each of the dedicated DQS pins is internally connected to the DQS phase shift circuitry. The pinout information contained in the LatticeECP2/M Family Data Sheet shows pin locations for the DQS pads. Table 12-1 shows an extract from the data sheet.

In this case, the DQS is marked as LDQS8 (L = left side, 8 = associated PFU row/column). Since DQS is always the fifth True Pad in the DQ-DQS group, counting from low to high PFU Row/Column number, LDQS6 will cover PL2A to PL11B. Following this convention, there are eight pads before and seven pads after DQS for DQ available following counter-clockwise for the left and bottom sides of the device and following clockwise for the top and right sides of the device. The user can assign any eight of these pads to be DQ data signals.

Table 12-1. ECP2-50 672 fpBGA Pinout from LatticeECP2/M Family Data Sheet

Ball Number	Ball Function	Bank	Dual Function	Differential
D2	PL2A	7	VREF2_7	T*
D1	PL2B	7	VREF1_7	C*
GND	GNDIO	7		
F6	PL5A	7		T
F5	PL5B	7		C
VCCIO	VCCIO	7		
E4	PL6A	7		T*
E3	PL6B	7		C*
VCC	VCC	7		
E2	PL7A	7		T
E1	PL7B	7		C
GND	GNDIO	7		
GND	GND	7		
H6	PL8A	7	LDQS8	T*
H5	PL8B	7		C*
F2	PL9A	7		T
VCCIO	VCCIO	7		
F1	PL9B	7		C
H8	PL10A	7		T*
J9	PL10B	7		C*
G4	PL11A	7		T
GND	GNDIO	7		
G3	PL11B	7		C
H7	PL12A	7		T*
VCCAUX	VCCAUX	7		

DDR Software Primitives

This section describes the software primitives that can be used to implement DDR interfaces. These primitives include:

- **DQSDLL** – The DQS delay calibration DLL
- **DQSBUFC** – The DQS delay function and the clock polarity selection logic
- **IDDRMX1A** – The DDR input and DQS to system clock transfer registers with half clock cycle transfer
- **IDDRMFX1A** – The DDR input and DQS to system clock transfer registers with full clock cycle transfer
- **ODDRMXA** – The DDR output registers

HDL usage examples for each of these primitives are listed in Appendices A and B.

DQSDLL

The DQSDLL generates a 90-degree phase shift required for the DQS signal. This primitive implements the on-chip DQSDLL. Only one DQSDLL should be instantiated for all the DDR implementations on one half of the device. The clock input to this DLL should be at the same frequency as the DDR interface. The DLL generates the delay based on this clock frequency and the update control input to this block. The DLL updates the dynamic delay control to the DQS delay block when this update control (UDDCNTL) input is asserted. Figure 12-6 shows the primitive symbol. The active low signal on UDDCNTL updates the DQS phase alignment and should be initiated at the beginning of READ cycles.

Figure 12-6. DQSDLL Symbol

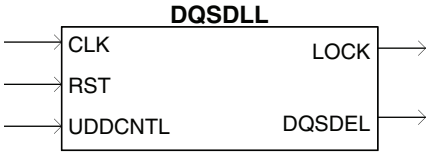


Table 12-2 provides a description of the ports.

Table 12-2. DQSDLL Ports

Port Name	I/O	Description
CLK	I	System CLK should be at the frequency of the DDR interface from the FPGA core.
RST	I	Resets the DQSDLL
UDDCNTL	I	Provides update signal to the DLL that will update the dynamic delay.
LOCK	O	Indicates when the DLL is in phase.
DQSDEL	O	The digital delay generated by the DLL, should be connected to the DQSBUF primitive.

DQSDLL Update Control: The DQS Delay can be updated for PVT variation using the UDDCNTL input. The DQSDEL is updated when the when the UDDCNTL is held LOW. The DQSDEL can be updated when variations are expected. DQSDEL can be updated anytime, except when the memory controller is receiving data from the memory.

DQSDLL Configuration: By default, this DLL will generate a 90-degree phase shift for the DQS strobe based on the frequency of the input reference clock to the DLL. The user can control the sensitivity to jitter by using the LOCK_SENSITIVITY attribute. This configuration bit can be programmed to be either “HIGH” or “LOW”.

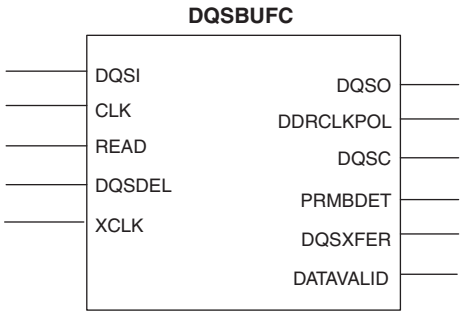
The DLL Lock Detect circuit has two modes of operation controlled by the LOCK_SENSITIVITY bit, which selects more or less sensitivity to jitter. If this DLL is operated at or above 150 MHz, it is recommended that the LOCK_SENSITIVITY bit be programmed “HIGH” (more sensitive). When running at or below 100 MHz, it is recom-

mended that the bit be programmed “LOW” (more tolerant). For 133 MHz, the LOCK_SENSITIVITY bit can go either way.

DQSBUFC

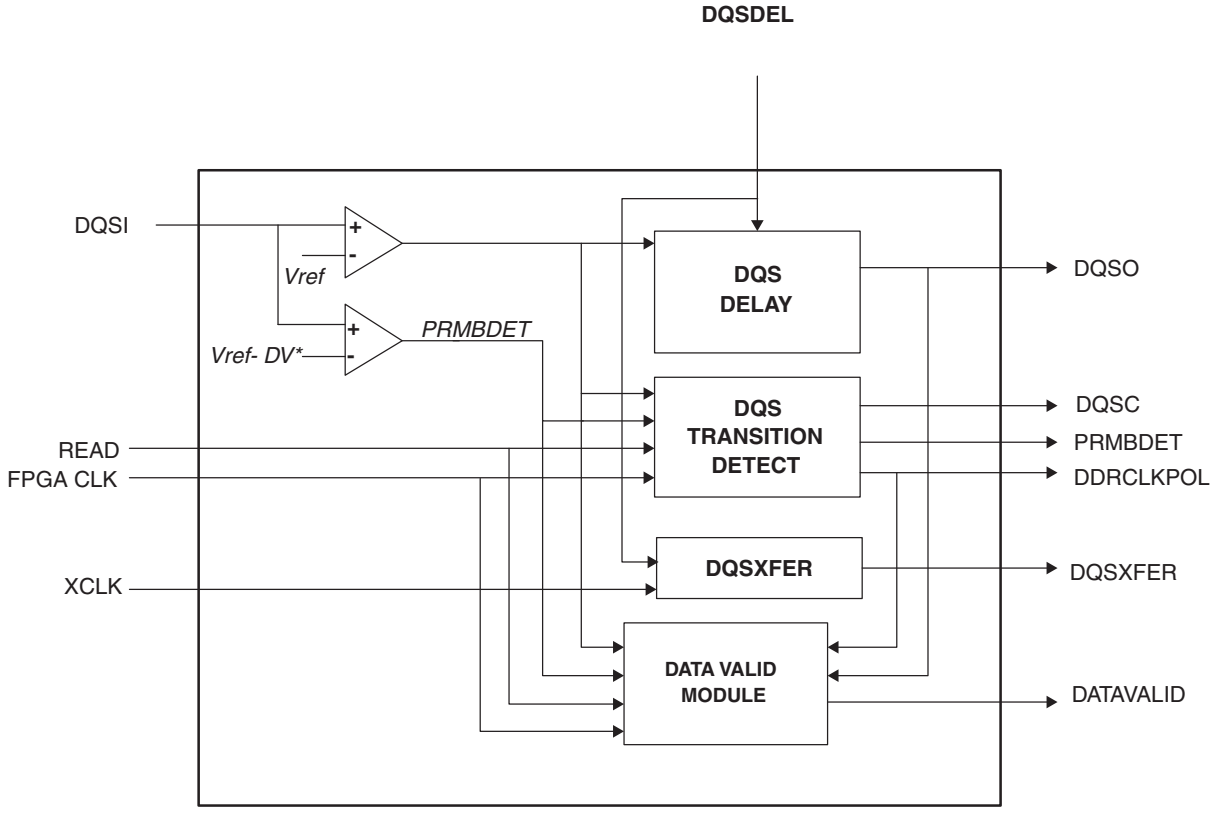
This primitive implements the DQS delay and the DQS transition detector logic. Figure 12-7 shows the primitive symbol.

Figure 12-7. DQSBUFC Symbol



The DQSBUFC is composed of the DQS Delay, the DQS Transition Detect and the DQSXFER block as shown in Figure 12-8. This block inputs the DQS and delays it by 90 degrees. It also generates the DDR Clock Polarity and the DQSXFER signal. The preamble detect (PRMBDET) signal is generated from the DQSI input using a voltage divider circuit.

Figure 12-8. DQSBUFC Function



**DV ~ 170mV for DDR1 (SSTL25 signaling)*
**DV ~ 120mV for DDR2 (SSTL18 signaling)*

DQS Delay Block: The DQS Delay block receives the digital control delay line (DQSDEL) coming from one of the two DQSDLL blocks. These control signals are used to delay the DQSI by 90 degrees. DQSO is the delayed DQS and is connected to the clock input of the first set of DDR registers.

DQS Transition Detect: The DQS Transition Detect block generates the DDR Clock Polarity signal based on the phase of the FPGA clock at the first DQS transition. The DDR READ control signal and FPGA CLK inputs to this coming and should be coming from the FPGA core.

DQSXFER: This block generates the 90-degree phase shifted clock to for the DDR Write interface. The input to this block is the XCLK. The user can choose to connect this either to the edge clock or the FPGA clocks. The DQSXFER is routed using the DQSXFER tree to all the I/Os spanned by that DQS.

Data Valid Module: The data valid module generates a DATAVALID signal. This signal indicates to the FPGA that valid data is transmitted out of the input DDR registers to the FPGA core.

Table 12-3 provides a description of the I/O ports associated with the DQSBUFC primitive.

Table 12-3. DQSBUFC Ports

Port Name	I/O	Description
DQSI	I	DQS Strobe signal from memory
CLK	I	System CLK
READ	I	Read generated from the FPGA core
DQSDEL	I	DQS Delay from the DQSDLL primitive
XCLK	I	Edge Clock or System CLK
DQSO	O	Delayed DQS Strobe signal, to the input capture register block
DQSC	O	DQS Strobe signal before delay, going to the FPGA core logic
DDRCLKPOL	O	DDR Clock Polarity signal
PRMBDET	O	Preamble detect signal, going to the FPGA core logic
DQSXFER	O	90 degree shifted clock going to the Output DDR register Block
DATAVALID	O	Signal indicating transmission of Valid data to the FPGA core

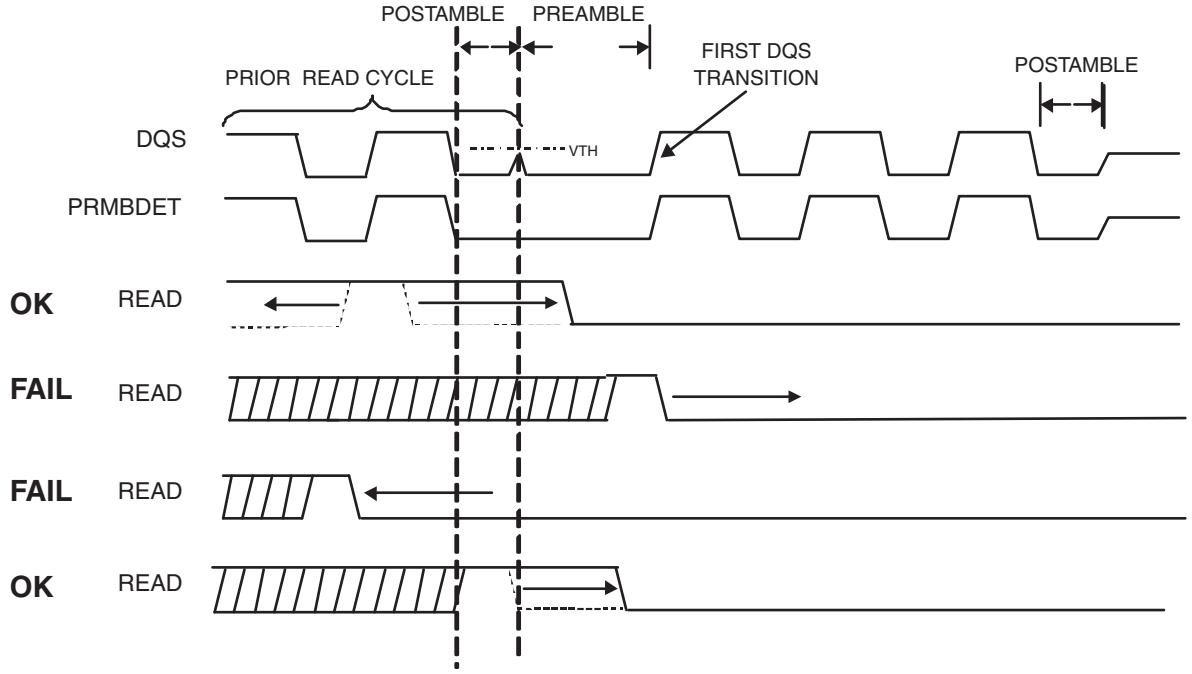
READ Pulse Generation

The READ signal to the DQSBUFC block is internally generated in the FPGA core. The READ signal goes high when the READ command to control the DDR-SDRAM is initially asserted. This precedes the DQS preamble by one cycle, yet may overlap the trailing bits of a prior read cycle. The DQS Detect circuitry of the LatticeECP2/M device requires the falling edge of the READ signal to be placed within the preamble stage.

The preamble state of the DQS can be detected using the CAS latency and the round trip delay for the signals between the FPGA and the memory device. Note that the internal FPGA core generates the READ pulse. The rise of the READ pulse should coincide with the initial READ command of the Read Burst and need to go low before the Preamble goes high.

Figure 12-9 shows a READ Pulse timing example with respect to the PRMBDET signal.

Figure 12-9. READ Pulse Generation



IDDRMX1A

This primitive will implement the input register block in memory mode. The DDR registers are designed to use edge clock routing on the I/O side and the primary clock on the FPGA side. The ECLK input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFC primitive). The SCLK input is connected to the system (FPGA) clock. DDRCLKPOL is an input from the DQS Clock Polarity tree. This signal is generated by the DQS Transition detect circuit in the hardware. The DDRCLKPOL signal is used to choose the polarity of the SCLK to the synchronization registers.

Figure 12-10. IDDRMX1A Symbol

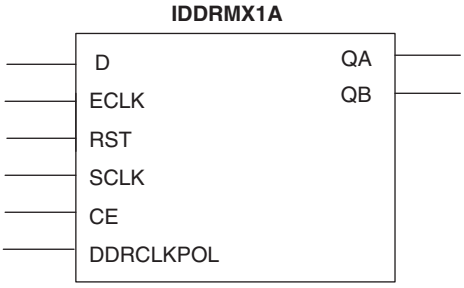


Table 12-4 provides a description of all I/O ports associated with the IDDRMX1A primitive.

Table 12-4. IDDRMX1A Ports

Port Name	I/O	Definition
D	I	DDR Data
ECLK	I	The phase shifted DQS should be connected to this input
RST	I	Reset
SCLK	I	System CLK
CE	I	Clock enable
DDRCLKPOL	I	DDR clock polarity signal
QA	O	Data at Positive edge of the CLK
QB	O	Data at the negative edge of the CLK

Note: The DDRCLKPOL input to IDDRMX1A should be connected to the DDRCLKPOL output of DQSBUFC.

Figure 12-11 shows the Input Register Block configured in the IDDRMX1A mode.

Figure 12-11. Input Register Block in IDDRMX1A Mode

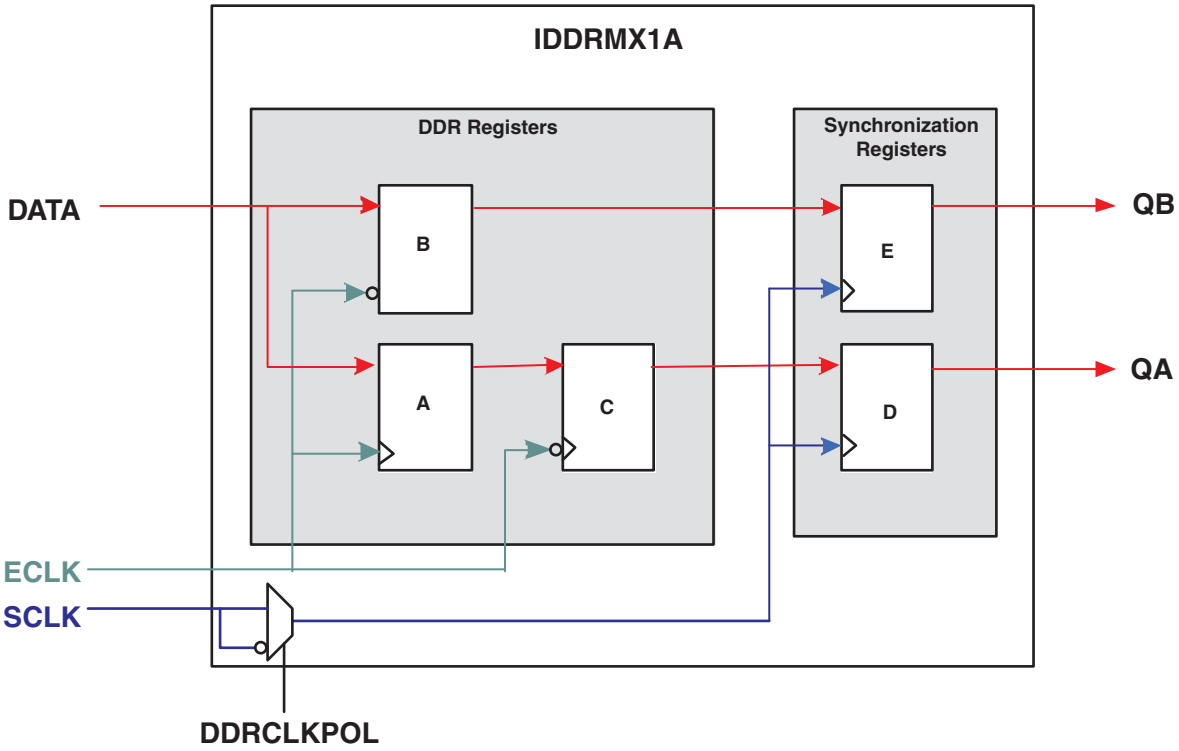
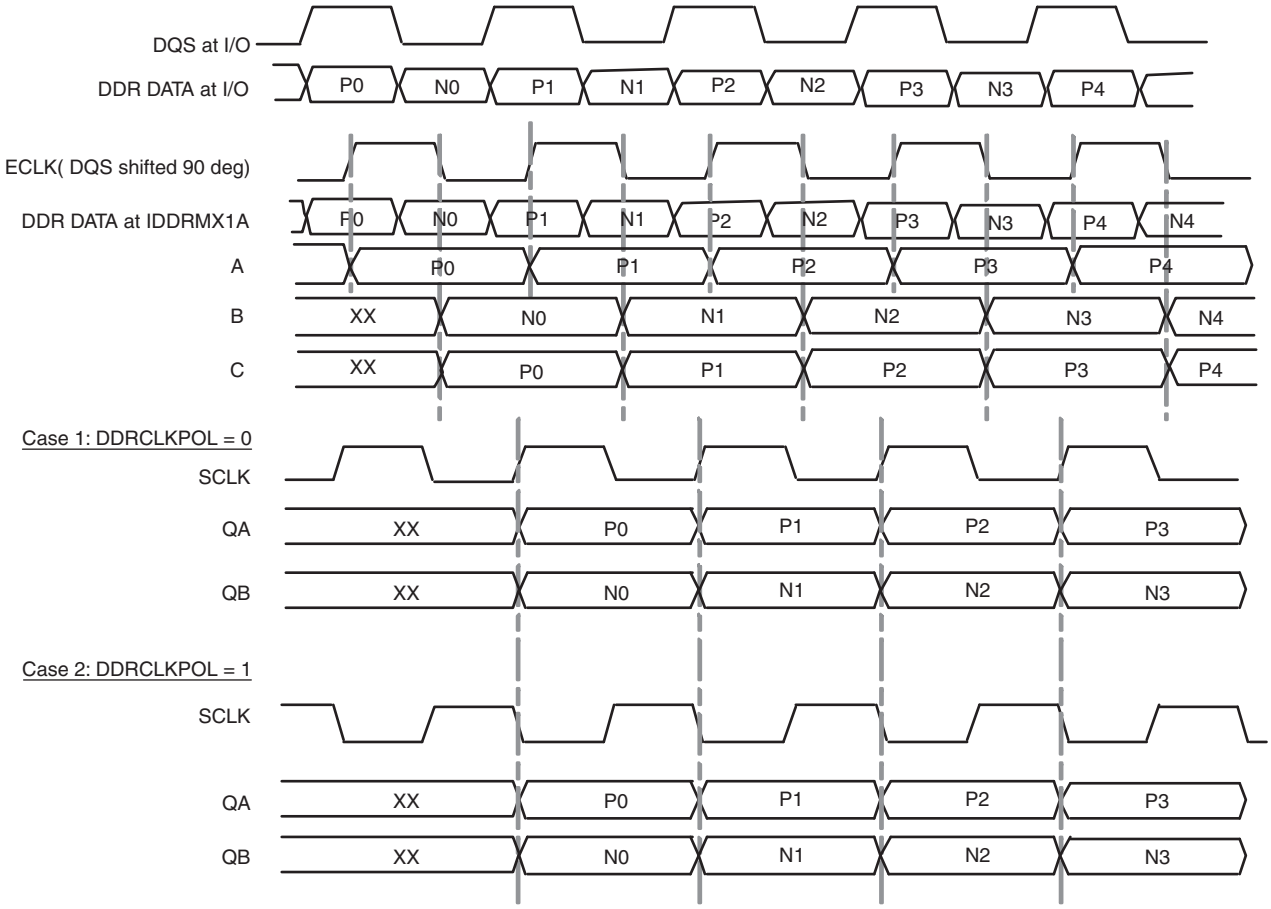


Figure 12-12 shows the IDDRMX1A timing waveform.

Figure 12-12. IDDRMX1A Waveform



IDDRMFX1A

With the IDDRMX1A, the data can enter the FPGA at either the positive or negative edge of the SCLK depending on the state of the DDRCLKPOL signal. The IDDRMFX1A module includes an additional clock transfer stage that ensures that the data is transferred at a known edge of the system clock.

Figure 12-13. IDDRMFX1A Symbol

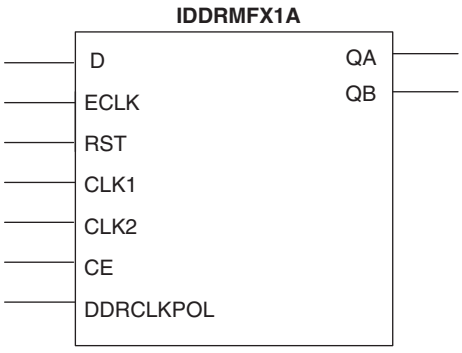


Table 12-5 provides a description of all I/O ports associated with the IDDRMFX1A primitive.

Table 12-5. IDDRMFX1A Ports

Port Name	I/O	Description
D	I	DDR Data
ECLK	I	The phase shifted DQS should be connected to this input
RST	I	Reset
CLK1	I	Slow FPGA CLK
CLK2	I	Slow FPGA CLK
CE	I	Clock enable
DDRCLKPOL	I	DDR clock polarity signal
QA	O	Data at the positive edge of the CLK
QB	O	Data at the negative edge of the CLK

Note: The DDRCLKPOL input to IDDRMFX1A should be connected to the DDRCLKPOL output of DQSBUFC.

Figure 12-14 shows the LatticeECP2 Input Register Block configured to function in the IDDRXMFX1A mode.

The DDR registers are designed to use Edge clock routing on the I/O side and the primary clock on the FPGA side. The ECLK input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFC primitive). The CLK1 and CLK2 inputs should be connected to the slow system (FPGA) clock. DDRCLKPOL is an input from the DQS Clock Polarity tree. This signal is generated by the DQS Transition detect circuit in the hardware. The additional clock transfer registers are shared with the output register block.

Figure 12-14. Input Register Block in IDDRMFX1A Mode

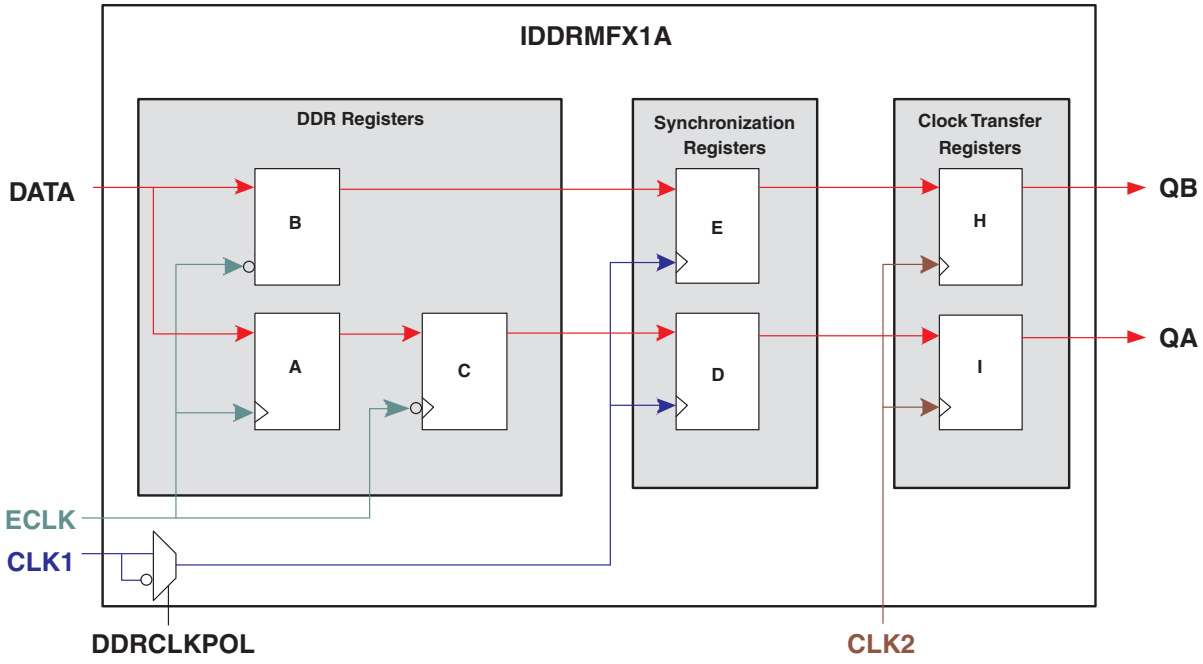
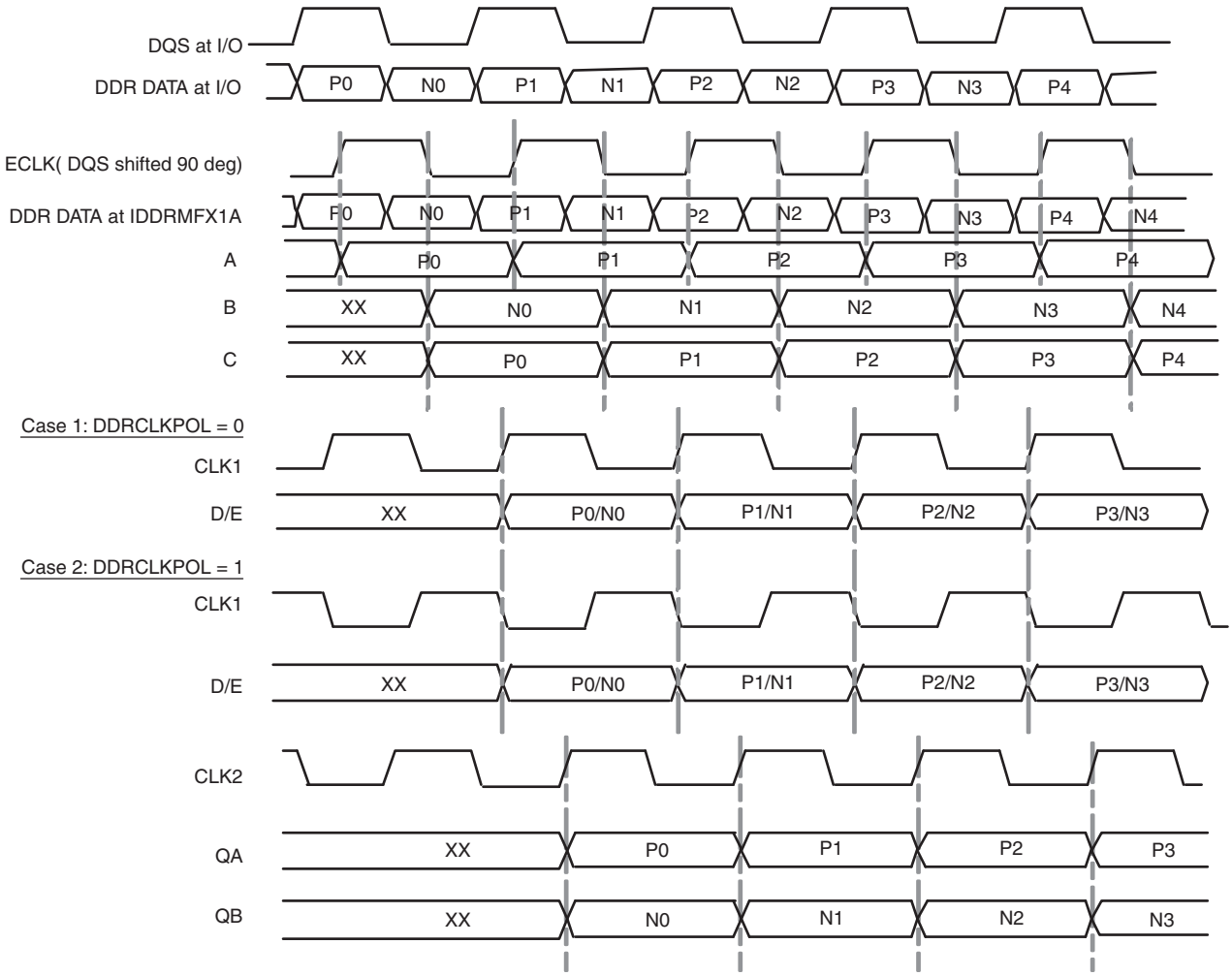


Figure 12-15 shows the IDDRMFX1A timing waveform.

Figure 12-15. IDDRMFX1A Waveform



ODDRMXA

The ODDRMXA primitive implements the output register for both the write and the tristate functions. This primitive is used to output DDR data and the DQS strobe to the memory. All the DDR output tristate functions are also implemented using this primitive.

Figure 12-16 shows the ODDRMXA primitive symbol and its I/O ports.

Figure 12-16. ODDRMXA Symbol

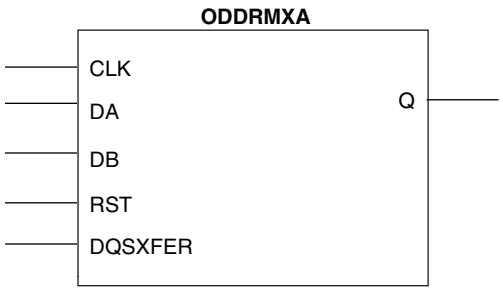


Table 12-6 provides a description of all I/O ports associated with the ODDRMXA primitive.

Table 12-6. ODDRMXA Ports

Port Name	I/O	Description
CLK	I	System CLK or ECLK
DA	I	Data at the negative edge of the clock
DB	I	Data at the positive edge of the clock
RST	I	Reset
DQSXFER	I	90-degree phase shifted clock coming from the DQSBUFC block
Q	I	DDR data to the memory

Notes:
1. RST should be held low during DDR Write operation.
2. DDR output and tristate registers do not have CE support. RST is available for the tristate DDRX mode (while reading). The LSR will default to set when used in the tristate mode.
3. When asserting reset during DDR writes, it is important to keep in mind that this only resets the flip-flops and not the latches.

Figure 12-17 shows the LatticeECP2 Output Register Block configured in the ODDRXMA mode.

Figure 12-17. Output Register Block in ODDRXMA Mode

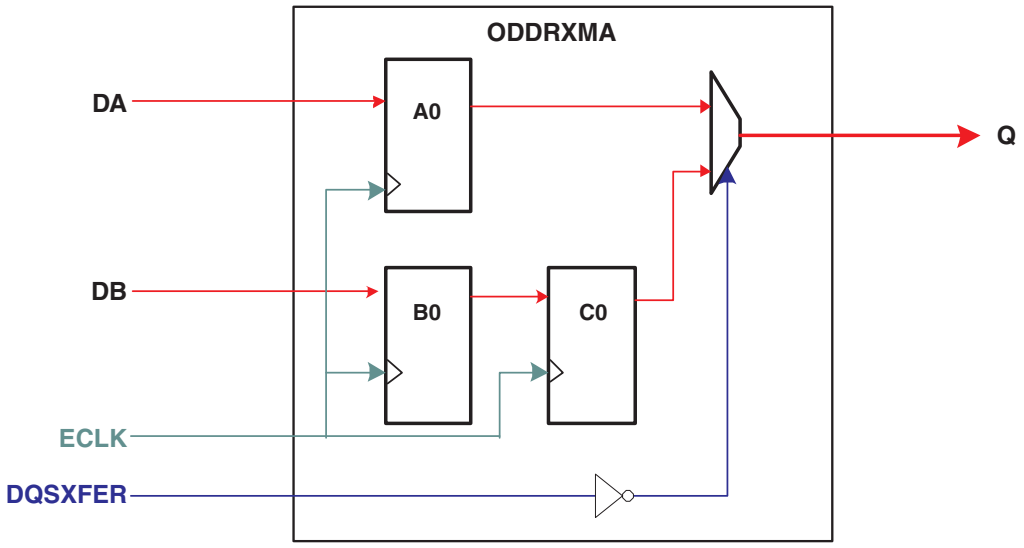
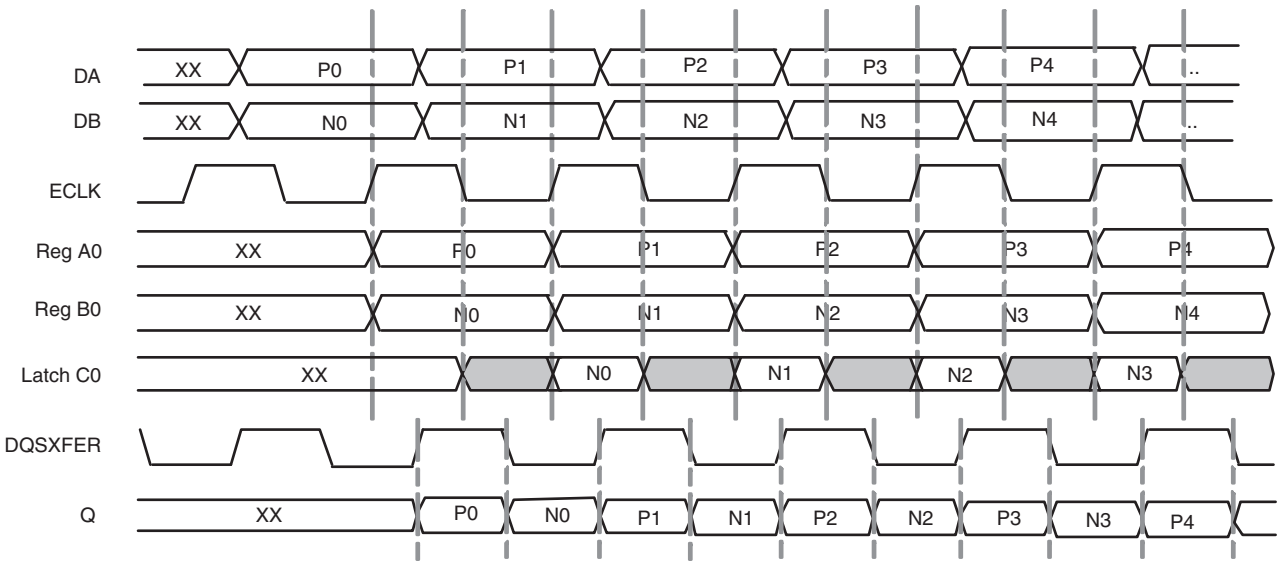


Figure 12-18 shows the ODDRMXA timing waveform.

Figure 12-18. ODDRMXA Waveform



Note that the DQSXFER is inverted inside the ODDRXMA. This will cause the data coming out of the ODDRXMA to be -90° in phase with the output of the ODDRXC module.

Memory Read Implementation

LatticeECP2/M devices contain a variety of features to simplify implementation of the read portion of a DDR interface:

- DLL compensated DQS delay elements
- DDR input registers
- Automatic DQS to system clock domain transfer circuitry
- Data Valid Module

DLL Compensated DQS Delay Elements

The DQS from the memory is connected to the DQS Delay element. The DQS Delay block receives a 6-bit delay control from the on-chip DQSDLL. This 6-bit delay is modeled as a single bit in the software. The LatticeECP2/M devices support two DQSDLL, one on the left and one on the right side of the device. The DQSDEL generated by the DQSDLL on the left side is routed to all the DQS blocks on the left and bottom half of the device. The delay generated by the DQSDLL on the right side is distributed to all the DQS Delay blocks on the right side and the other bottom half of the device. There are no DQS pins on the top banks of the device. These digital delay control signals are used to delay the DQS from the memory by 90 degrees.

The DQS received from the memory is delayed in each of the DQS Delay blocks and this delay DQS is used to clock the first set stage DDR input registers.

DQS Transition Detect or Automatic Clock Polarity Select

In a typical DDR memory interface design, the phase relation between the incoming delayed DQS strobe and the internal system clock (during the READ cycle) is unknown. Prior to the READ operation in DDR memories, DQS is in tristate (pulled by termination). Coming out of tristate, the DDR memory device drives DQS low in the Preamble State. The DQS Transition Detect block detects the first DQS rising edge after a preamble transition and generates a signal indicating the required polarity for the FPGA system clock (DDRCLKPOL). This signal is used to control the polarity of the clock to the synchronizing registers.

Data Valid Module

The data valid module generates a DATAVALID signal. This signal indicates to the FPGA that valid data is transmitted out the input DDR registers to the FPGA core.

DDR I/O Register Implementation

The first set of DDR registers is used to de-mux the DDR data at the positive and negative edge of the phase shifted DQS signal. The register that captures the positive-edge data is followed by a negative-edge triggered register. This register transfers the positive edge data from the first register to the negative edge of DQS so that both the positive and negative portions of the data are now aligned to the negative edge of DQS.

The second stage of registers is clocked by the FPGA clock, the polarity of this clock is selected by the DDR Clock Polarity signal generated by the DQS Transition Detect Block.

The I/O Logic registers can be implemented in two modes:

- Half Clock Transfer Mode
- Full Clock Transfer Mode

In Half Clock Transfer mode the data is transferred to the FPGA core after the second stage of the register. In Full Clock Transfer mode, an additional stage of I/O registers clocked by the FPGA clock is used to transfer the data to the FPGA core.

The LatticeECP2/M Family Data Sheet explains each of these circuit elements in more detail.

Memory Read Implementation in Software

Three primitives in the ispLEVER® design tools represent the capability of these three elements. The DQSDLL represents the DLL used for calibration. The IDDRMX1A/IDDRMFX1A primitive represents the DDR input registers and clock domain transfer registers with or without full clock transfer. Finally, the DQSBUFC represents the DQS delay block, the clock polarity control logic and the Data Valid module. Figures 12-19 and 12-20 show the READ interface block generated using the IPexpress™ tool in the ispLEVER software.

Figure 12-19. Software Primitive Implementation for Memory READ (Half Clock Transfer)

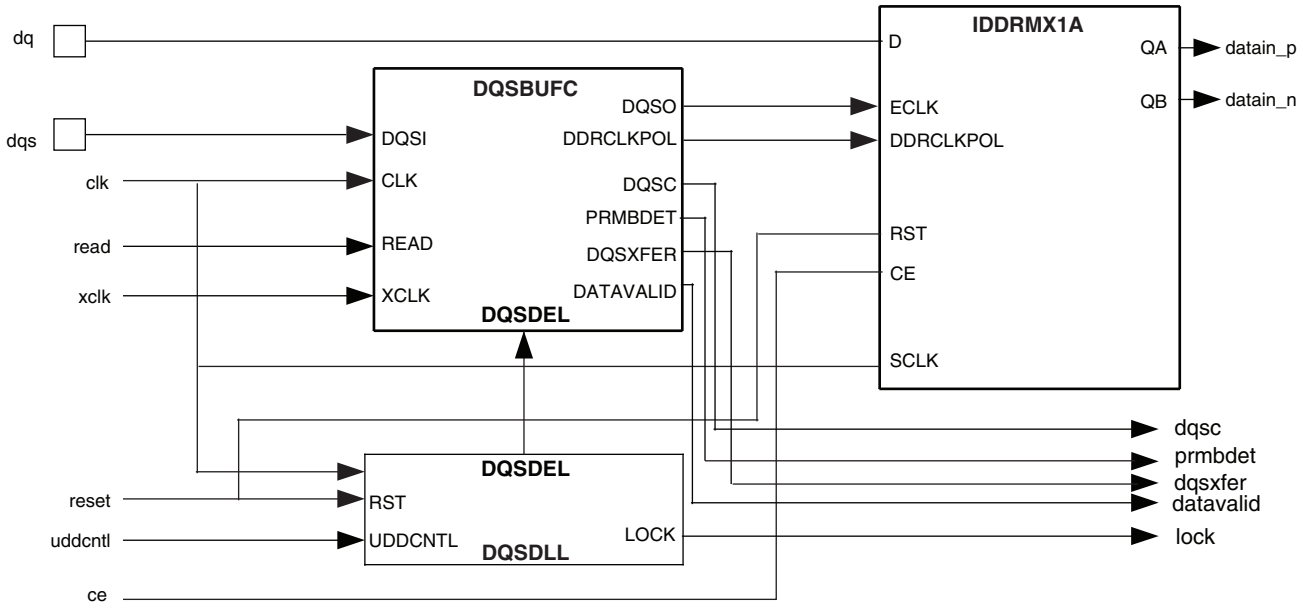
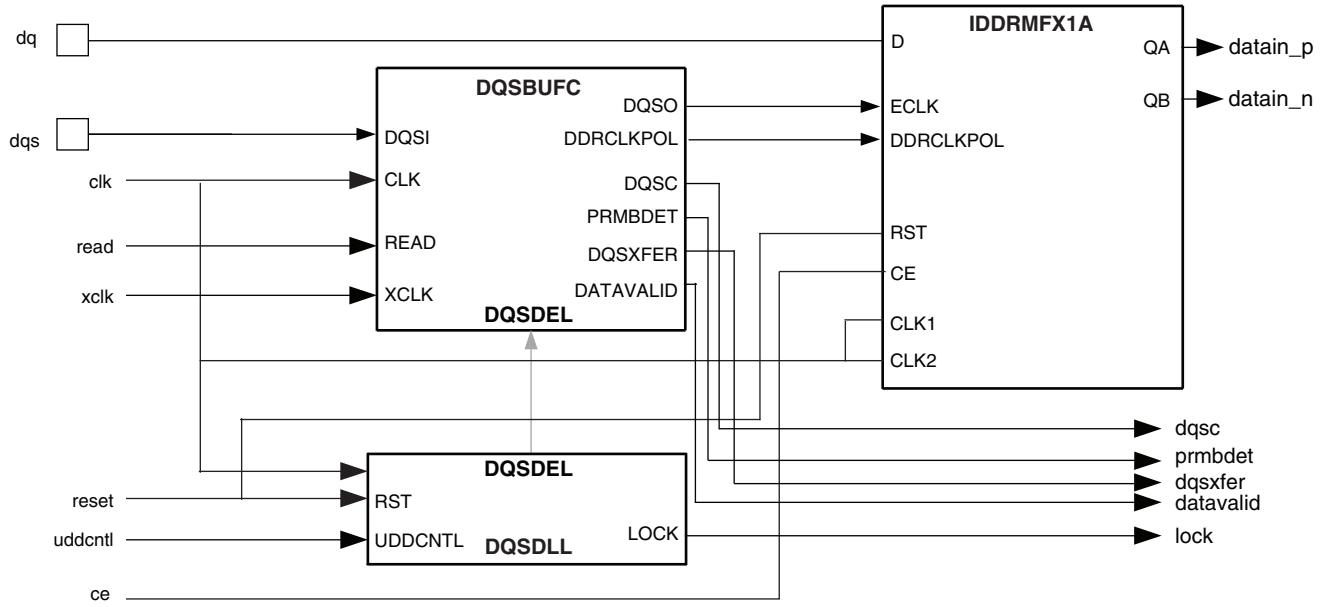


Figure 12-20. Software Primitive Implementation for Memory READ (Full Clock Transfer)



Read Timing Waveforms

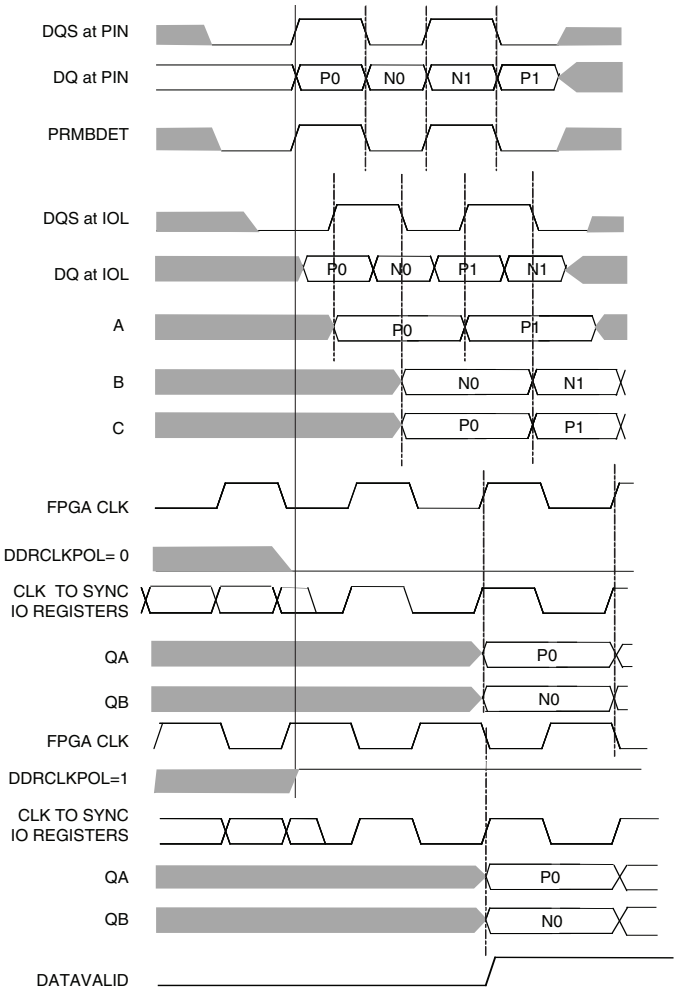
Figures 12-21 and 12-22 show READ data transfer for half and full clock cycle data transfer based on the results of the DQS Transition detector logic. This circuitry decides whether or not to invert the phase of FPGA system CLK to the synchronization registers based on the relative phases of PRMBDET and CLK.

- **Case 1:** If the FPGA clock is low on the first PRMBDET transition, then DDRCLKPOL is low and no inversion is required.
- **Case 2:** If the FPGA clock is high on the first PRMBDET, then DDRCLKPOL is high and the FPGA clock (CLK) needs to be inverted before it is used for synchronization.

Figure 12-21 illustrates the DDR data timing using half clock transfer mode at different stages of the IDDRMX1A registers. The first stage of the register captures data on the positive edge as shown by signal A and the negative edge as shown by signal B. Data stream A goes through an additional half clock cycle transfer shown by signal C. Phase-aligned data streams B and C are presented to the next stage registers clocked by the FPGA clock.

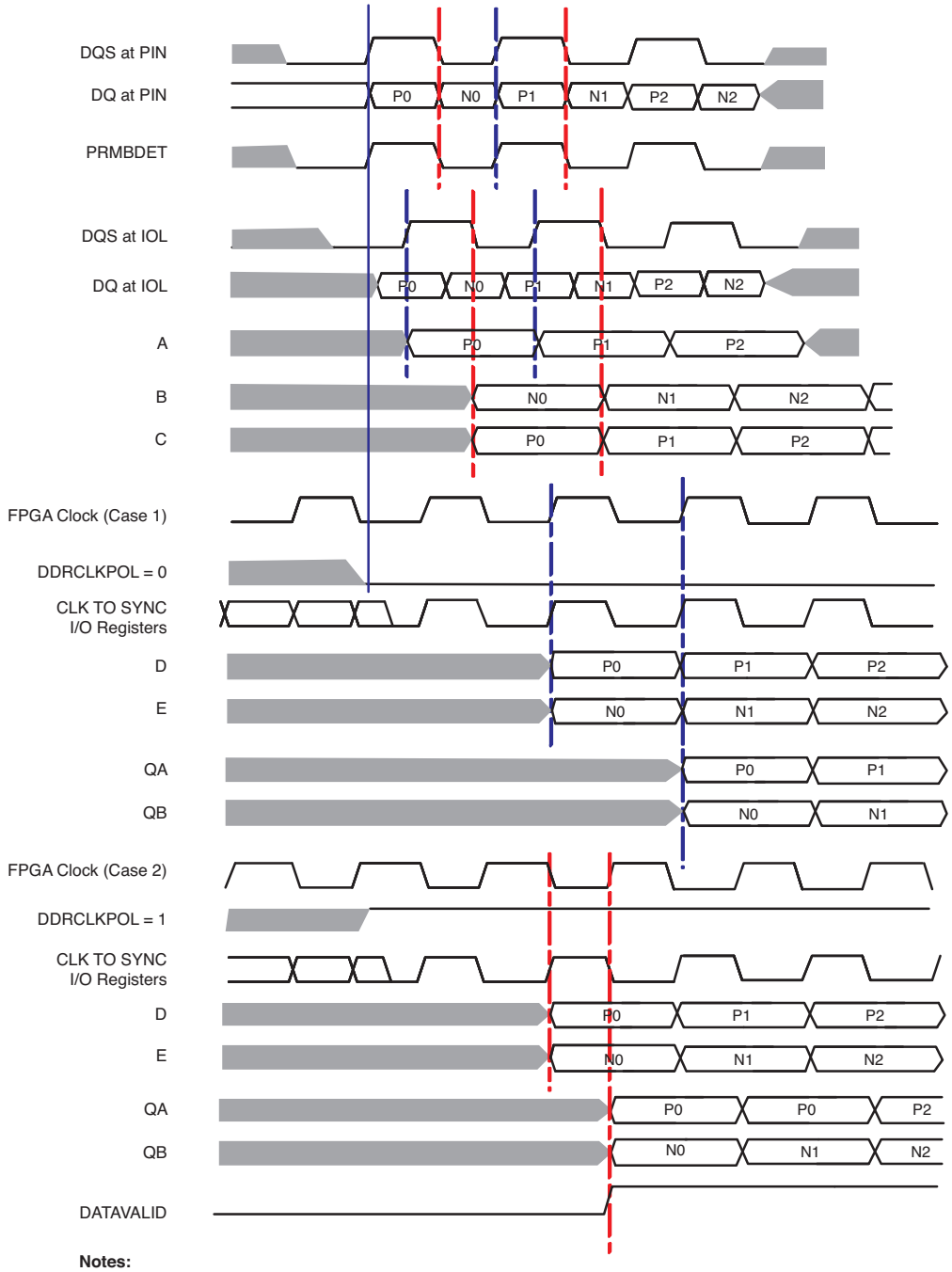
Figure 12-22 illustrates the DDR data timing using full clock transfer mode at different stages of IDDRMFX1A registers. In addition to the first two register stages in the half clock mode, the full clock transfer mode has an additional stage register clocked by the FPGA clock. In this case, D and E are the data streams after the second register stage presented to the final stage of registers clocked by the FPGA clock.

Figure 12-21. READ Data Transfer When Using IDDRMX1A



Notes:
1. DDR memory sends DQ aligned to DQS Strobe.
2. The DQS Strobe is delayed by 90 degrees using the dedicated DQS logic.
3. DQ is now center aligned to DQS Strobe.
4. PRMBDET is the Preamble detect signal generated using the DQSBUFB primitive. This is used to generate the DDRCLKPOL signal.
5. The first set of I/O registers, A and B, capture data on the positive and negative edges of DQS.
6. I/O Register C transfers data so that both data are now aligned to negative edge of DQS.
7. DDCLKPOL signal generated will determine if the FPGA CLK going into the synchronization registers need to be inverted. The DDRCLK-POL=0 when the FPGA CLK is LOW at the first rising edge of PRMBDET. The clock to the synchronization registers is not inverted. The DDRCLKPOL=1 when the FPGA CLK is HIGH at the first rising edge of PRMBDET. In this case the clock to the synchronization register is inverted.
8. The I/O synchronization registers capture data on either the rising or falling edge of the FPGA clock.
9. The DATAVALID signal goes HIGH when valid data enters the FPGA core. Once DATA VALID is asserted, it stays high until the next READ pulse.

Figure 12-22. Read Data Transfer When Using IDDRMFX1A



Notes:

1. DDR memory sends DQ aligned to DQS strobe.
2. The DQS strobe is delayed by 90 degress, using the dedicated DQS logic.
3. DQ is now center-aligned to the DQS strobe.
4. PRMBDET is the preamble detect signal generated usin the DQSBUFB primitive. This is used to generate the DDRCLKPOL signal.
5. The first set of I/O registers, A and B, capture data on the positive and negative edges of DQS.
6. I/O register C transfers data such that both data are aligned to the negative edge of DQS.
7. The DDCLKPOL signal generated will determine whether the FPGA clock going into the synchronization registers needs to be inverted. The DDRCLKPOL = 0 when the FPGA clock is LOW at the first rising edge of PRMBDET. So, the clock to the synchronization registers is not inverted. The DDRCLKPOL = 1 when the FPGA clock is HIGH at the first rising edge of PRMBDET. In this case, the clock to the synchronization register is inverted.
8. Registers D and E capture data at the FPGA clock.
9. The data again registers at the FPGA clock to ensure a full clock cycle transfer.
10. The DATAVALID signal goes HIGH when valid data enters the FPGA core. Once DATAVALID is asserted, it stays HIGH until the next READ pulse.

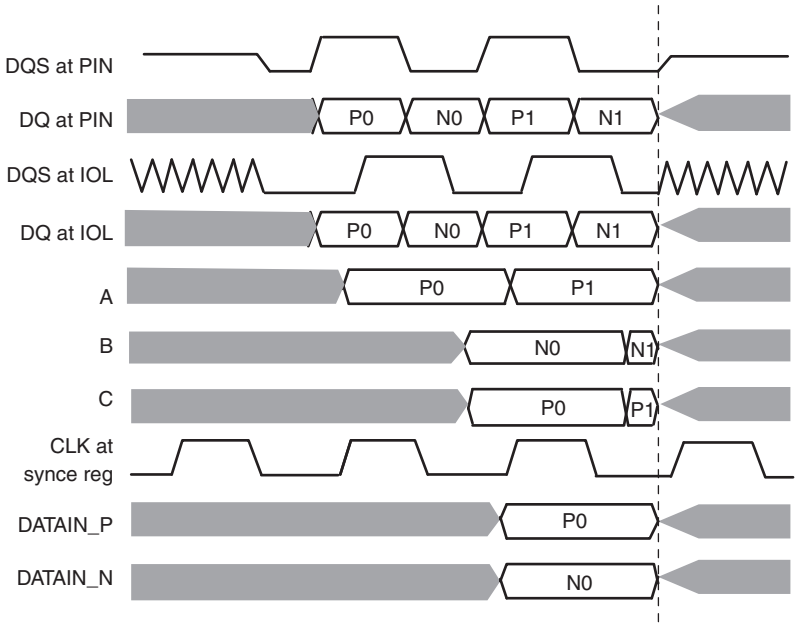
Data Read Critical Path

When using IDDRXM1A, the data in the second stage DDR registers can be registered either on the positive edge or on the falling edge of the FPGA clock depending on the DDRCLKPOL signal. In order to ensure that the data transferred to the FPGA core registers is aligned to the rising edge of the system clock, this path should be constrained with a half clock transfer. This half clock transfer can be forced in the software by assigning a multi-cycle constraint (multi-cycle of 0.5 X) on all the data paths to first PFU register. When using IDDRXMFX1A, there is an additional stage of registers inside the I/O block that transfers data to the positive edge of the FPGA clock. Hence no constraint is required for this case.

DQS Postamble

At the end of a READ cycle, the DDR SDRAM device executes the READ cycle postamble and then immediately tristates both the DQ and DQS output drivers. Since neither the memory controller (FPGA) nor the DDR SDRAM device are driving DQ or DQS at that time, these signals float to a level determined by the off-chip termination resistors. While these signals are floating, noise on the DQS strobe may be interpreted as a valid strobe signal by the FPGA input buffer. This can cause the last READ data captured in the IOL input DDR registers to be overwritten before the data has been transferred to the free running resynchronization registers inside the FPGA.

Figure 12-23. Postamble Effect on READ



LatticeECP2/M devices have extra dedicated logic in the in the DQS Delay Block that prevents this postamble problem. The DQS postamble logic is automatically implemented when the user instantiates the DQS Delay logic (DQSBUFC software primitive) in the design.

Memory Write Implementation

To implement the write portion of a DDR memory interface, two streams of single data rate data must be multiplexed together with data transitioning on both edges of the clock. In addition, during a write cycle, DQS must arrive at the memory pins center-aligned with the data, DQ. Along with the DQS strobe and data this portion of the interface must also provide the CLKP, CLKN Address/Command and Data Mask (DM) signals to the memory.

It is the responsibility of the FPGA output control to edge-align the DDR output signals (ADDR,CMD, DQS, but not DQ, DM) to the rising edge of the outgoing differential clock (CLKP/CLKN).

Challenges encountered by the during Memory WRITE:

1. DQS needs to be center-aligned with the outgoing DDR Data, DQ.

2. Differential CLK signals (CLKP and CLKN) need to be generated.
3. The controller must meet the DDR interface specification for t_{DSS} and t_{DSH} parameters, defined as DQS falling to CLKP rising setup and hold times.
4. The DDR output data must be muxed from two SDR streams into a single outgoing DDR data stream.

All DDR output signals (“ADDR, CMD”, DQS, DQ, DM) are initially aligned to the rising edge of the FPGA clock inside the FPGA core. The relative phase of the signals may be adjusted in the IOL logic before departing the FPGA. These adjustments are shown in Figure 12-24.

LatticeECP2/M devices contain DDR output and tri-state registers along with the DQSXFER signal generated by the DQSBUFC that allows easy implementation of the write portion of the DDR memory interfaces. The DDR output registers can be accessed in the design tools via the ODDRMXA and the ODDRXC primitives.

The DQS signal and the DDR clock outputs are generated using the ODDRXC primitive. As shown in the figure, the CLKP and DQS signals are generated so that they are 180 degrees in phase with the clock. This is done by connecting “1” to the DA input and “0” to the DB inputs of the ODDRXC primitive. Refer to the DDR Generic Software Primitive section of this document to see the ODDRXC timing waveforms.

The DDR clock output is then fed into a SSTL differential output buffer to generate CLKP and CLKN differential clocks. Generating the CLKN in this manner prevents any skew between the two signals. When interfacing to DDR1, SDRAM memory CLKP should be connected to the SSTL25D I/O standard. When interfacing to DDR2 memory, it should be connected to the SSTL18D I/O standard.

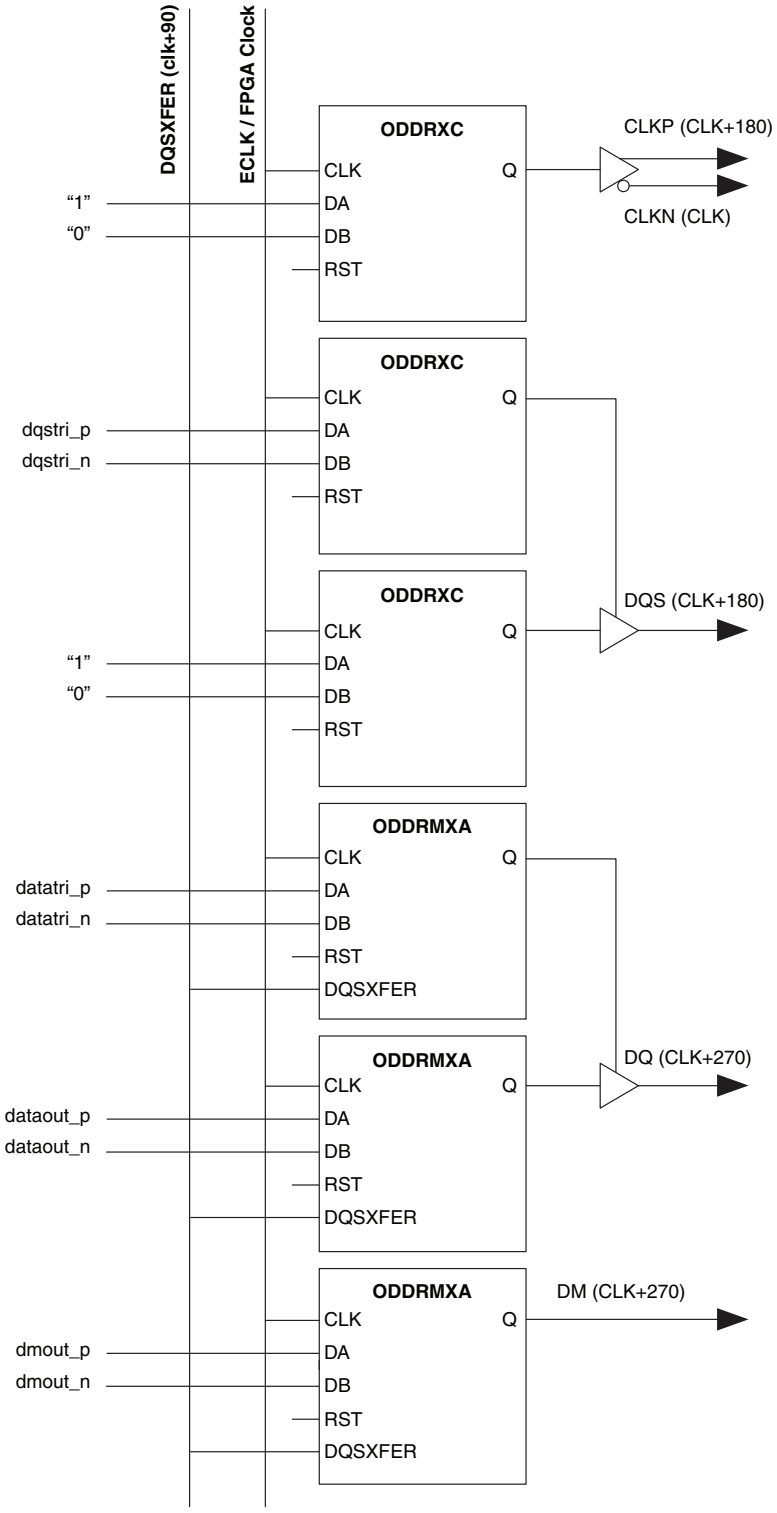
The DQSXFER output from the DQSBUFC block is the 90-degree phase shifted clock. This 90-degree phase shifted clock is used as an input to the ODDRMXA block. The ODDRMXA is used to generate the DQ and DM data outputs going to the memory. In the ODDRMXA module, the data is first registered using the ECLK or FPGA clock input and then shifted out using the DQSXFER signal. To ensure that the data going to the memory is center-aligned to the DQS, the DQSXFER is inverted inside the ODDRXMA primitive. This will generate data that is center-aligned to the DQS. Refer to the Software Primitives section of this document for the ODDRXMA timing waveforms.

The DDR interface specification for t_{DSS} and t_{DSH} parameters defined as DQS falling to CLKP rising setup and hold times must be met. This is accomplished by ensuring that the CLKP and DQS signals are identical in phase.

The tristate control for the DQS and DQ outputs can also be implemented using the ODDRXC primitive.

Figure 12-24 shows the DDR Write implementation using the DDR primitives.

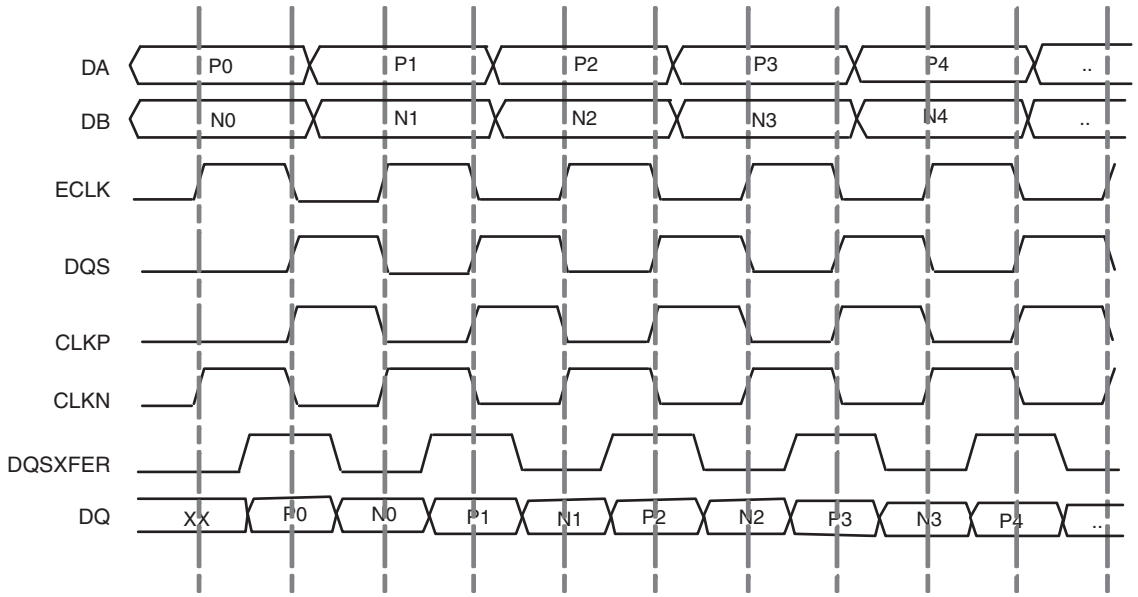
Figure 12-24. Software Implementation for Memory Write



Write Timing Waveforms

Figure 12-25 shows the DDR write side data transfer timing for the DQ Data pad and the DQS Strobe Pad. When writing to the DDR memory device, the DM (Data Mask) and the ADDR/ CMD (Address and Command) signals are also sent to the memory device along with the data and strobe signals.

Figure 12-25. DDR Write Data Transfer for DQ Data



Design Rules/Guidelines

Listed below are some rules and guidelines to keep in mind when implementing DDR memory interfaces in the LatticeECP2/M devices.

- The LatticeECP2/M devices have dedicated DQ-DQS banks. Please refer to the logical signal connections of the groups in the LatticeECP2/M Family Data Sheet before locking these pins.
- There are two DQSDLL on the device, one for the left half and one for the right half of the device. Therefore, only one DQSDLL primitive should be instantiated for each half of the device. Since there is only one DQSDLL on each half of the device, all the DDR memory interfaces on that half of the device should run at the same frequency. Each of the DQSDLL will generate 90-degree digital delay bits for all the DQS delay blocks on that half of the device based on the reference clock input to the DLL.
- When implementing a DDR SDRAM interface, all interface signals should be connected to the SSTL25 I/O standard. In the case of the DDR2 SDRAM interface, the interface signal should be connected to SSTL18 I/O standard.
- For DDR2, the differential DQS signals need to be connected to SSTL18 the Differential I/O standard.
- When implementing the DDR interface, the VREF1 of the bank is used to provide the reference voltage for the interface pins.

Generic High Speed DDR Implementation

In addition to the DDR memory interface, the I/O logic DDR registers can be used to implement high speed DDR interfaces. The Input DDR registers can operate in full clock transfer and half clock transfer modes. The DDR input and output register also support x1 and x2 gearing ratios. A gearing capability is provided to Mux/DeMux the I/O data rate (ECLK) to the FPGA clock rate (SCLK). For DDR interfaces, this ratio is slightly different than the SDR ratio. A basic 2x DDR element provides four FPGA side bits for two I/O side bits at half the clock rate on the FPGA side.

The data going to the DDR registers can be optionally delayed before going to the DDR register block.

Generic DDR Software Primitives

The IPexpress tool in the ispLEVER software can be used to generate the DDR modules. The various DDR modes described below can be configured in the IPexpress tool. The various modes are implemented using the following software primitives.

- IDDRXC – DDR Generic Input
- IDDRFXA – DDR Generic Input with full clock transfer (x1 gearbox)
- IDDRX2B – DDR Generic Input with 2x gearing ratio. DDRX2 inputs a double data rate signal as four data streams. Two stages of DDR registers are used to convert serial DDR data at input pad into four SDR data streams entering FPGA core logic.
- ODDRXC – DDR Generic Output
- ODDRX2B – DDR Generic Output with 2x gearing ratio. The DDRX2 inputs four separate data streams and outputs a single data stream to the I/O buffer.
- DELAYB – The DDR input can be optionally delayed before it is input to the DDR registers. The user can choose to implement a fixed delay value or use a dynamic delay.

IDDRXC

This primitive inputs DDR data at both edges of the CLK and generates two streams of data. The CLK to this module can be connected to either the edge clock or the primary FPGA clock.

Figure 12-26 shows the primitive symbol for IDDRXC mode.

Figure 12-26. IDDRXC Symbol

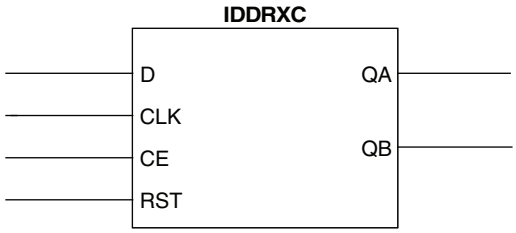


Table 12-7 lists the port names and descriptions for the IDDRXC primitive.

Table 12-7. IDDRXC Port Names

Port Name	I/O	Definition
D	I	DDR data
CLK	I	This clock can be connected to the ECLK or the FPGA clock
CE	I	Clock enable signal
RST	I	Reset to the DDR register
QA	O	Data at the positive edge of the clock
QB	O	Data at the negative edge of the clock

Figure 12-27 shows the LatticeECP2 Input Register Block configured in the IDDRXFC mode.

Figure 12-27. Input Register Block Configured as IDDRXC

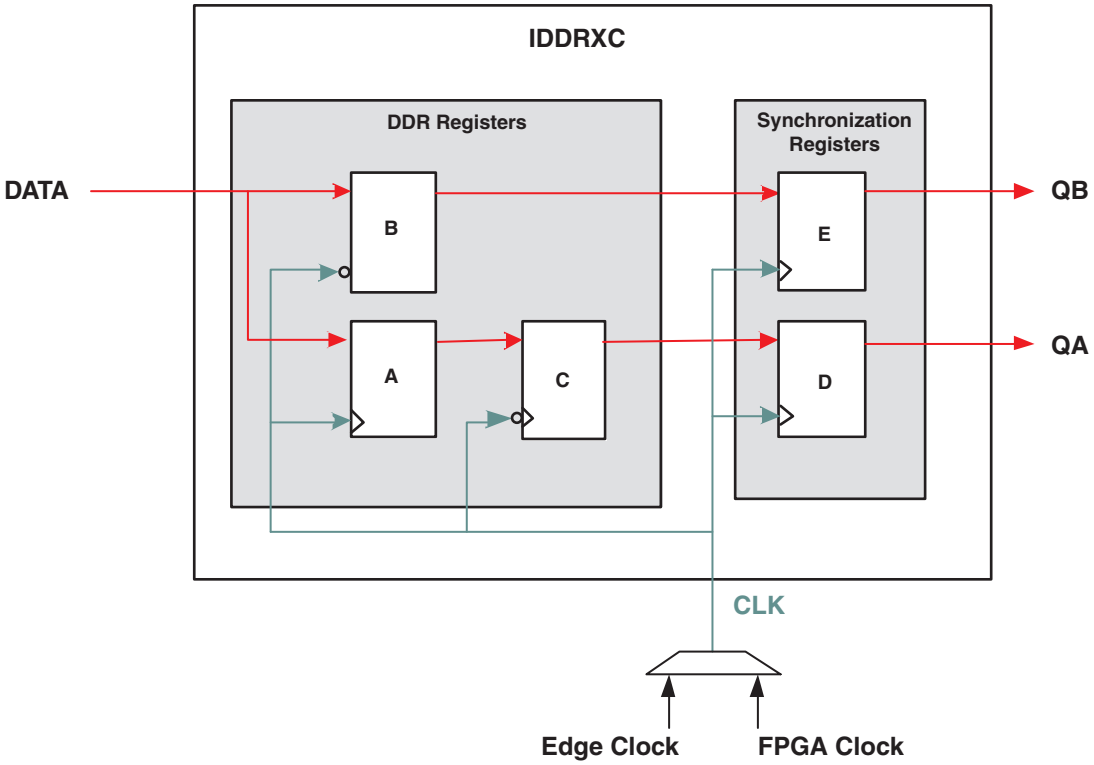
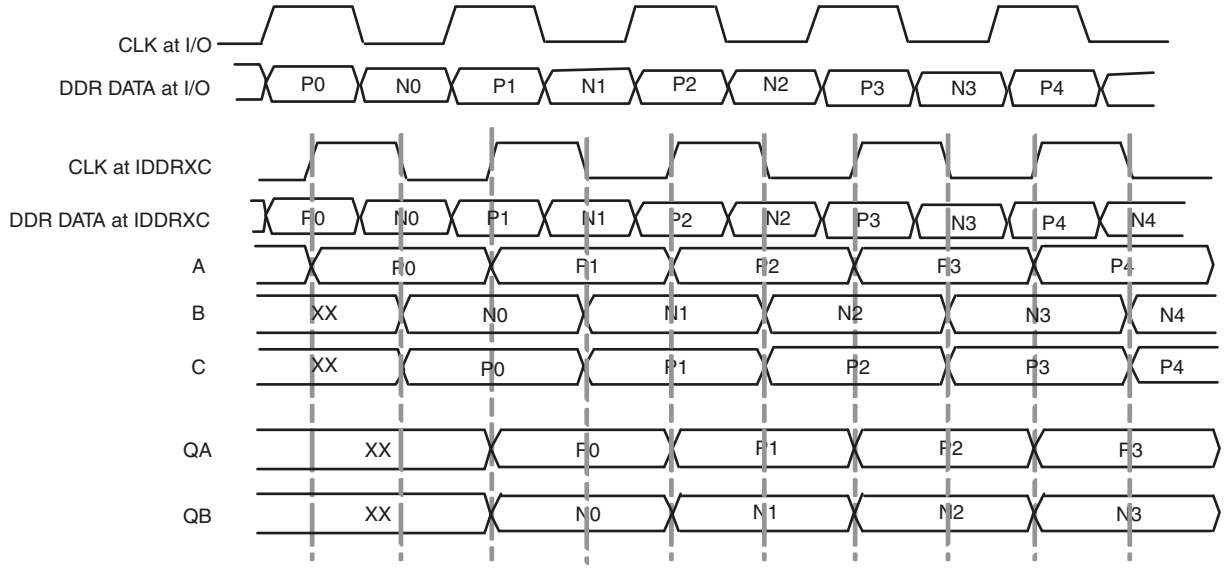


Figure 12-28 shows the timing waveform when using the IDDRXC module.

Figure 12-28. IDDRXC Waveform



IDDRFXA

This primitive inputs DDR data at both edges of clock CLK1 and generates two streams of data aligned to clock CLK2. CLK1 can be connected either to the edge clock or the internal FPGA clock. If the Edge clock input is used for CLK1 then CLK2 should be generated from the same clock going to CLK1.

Figure 12-29 shows the primitive symbol for the IDDRFXA mode.

Figure 12-29. IDDRFXA Symbol

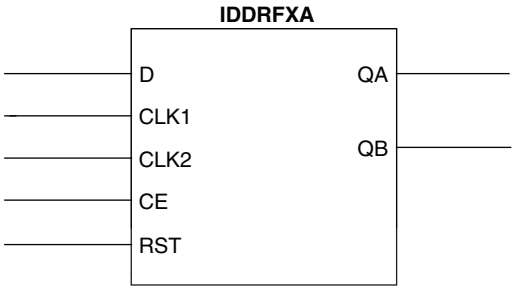


Table 12-8 lists the port names and descriptions for the IDDRFXA primitive.

Table 12-8. IDDRFXA Port Names

Port Name	I/O	Description
D	I	DDR data
CLK1	I	This clock can be connected to the ECLK or the FPGA clock
CLK2	I	This clock should be connected to the FPGA clock
CE	I	Clock Enable signal
RST	I	Reset to the DDR register
QA	O	Data at the positive edge of the clock
QB	O	Data at the negative edge of the clock

Figure 12-31 shows the LatticeECP2 Input Register Block configured in the IDDRXFXA mode. CLK1 used to register the DDR registers and the first set of synchronization registers. CLK2 is used by the third stage of registers and should be clocked by the FPGA clock. These clock transfer registers are shared with the output register block.

Figure 12-30. Input Register Block configured as IDDRFXA

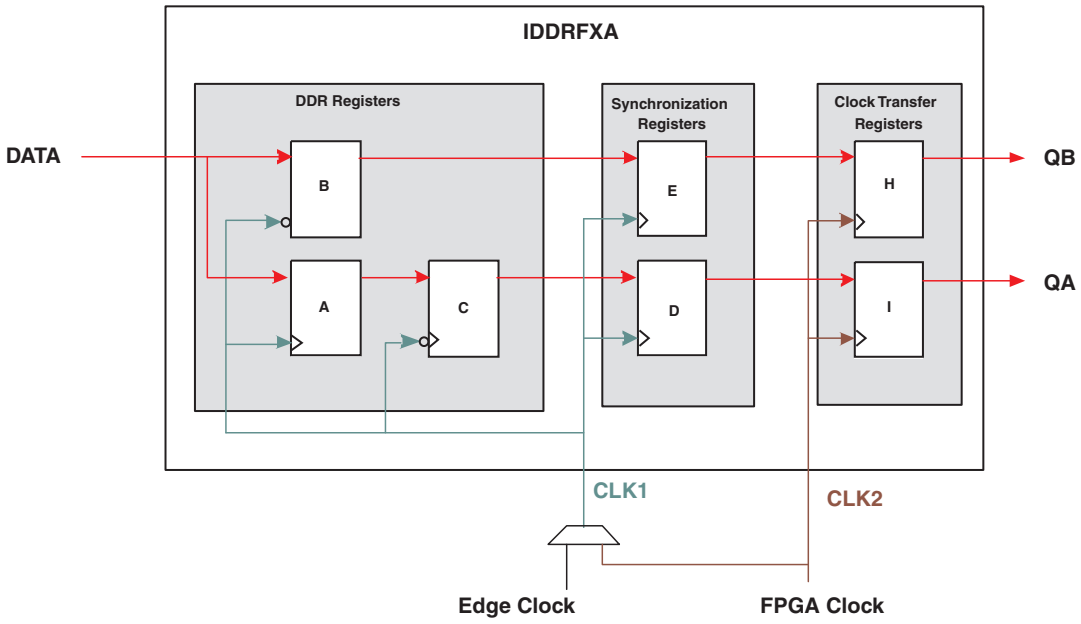
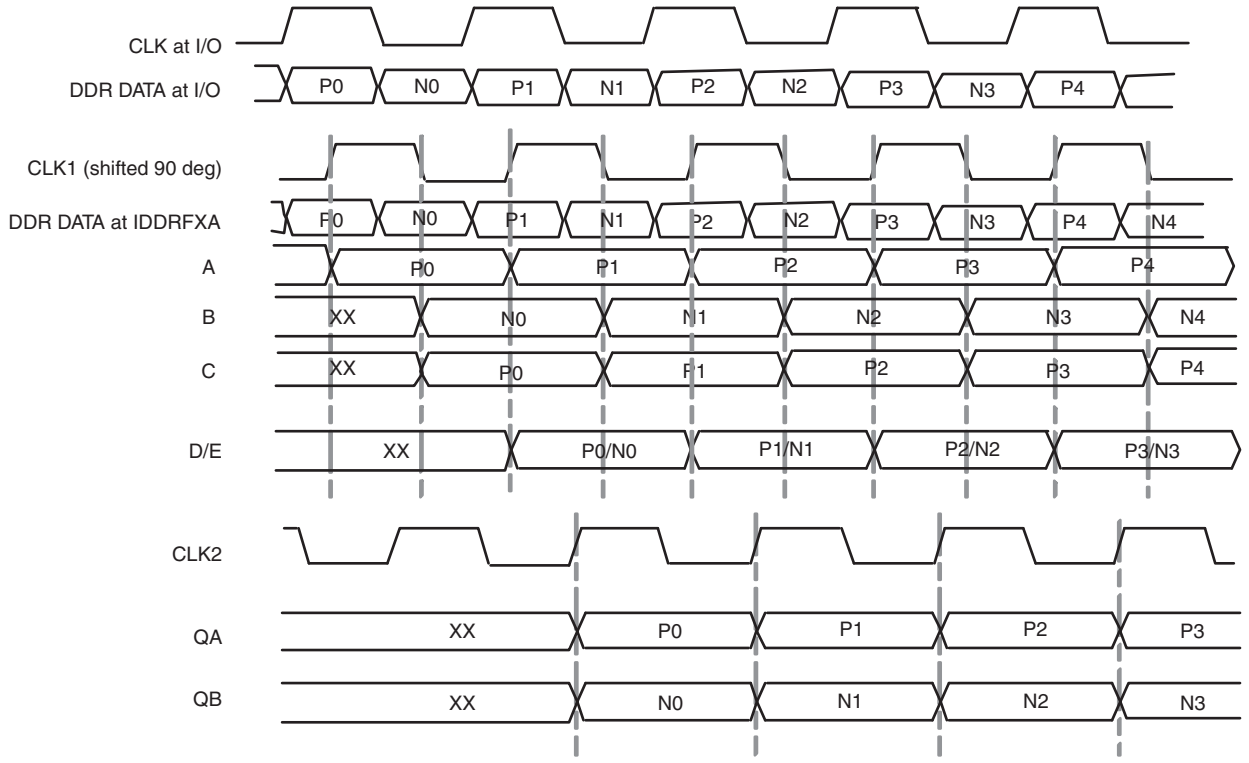


Figure 12-31 shows the timing waveform when using the IDDRFXA module.

Figure 12-31. IDDRFXA Waveform



IDDRX2B

This module is used when a gearing function is required. This primitive inputs the DDR data at both edges of the edge clock and generates four streams of data aligned to SCLK. SCLK is always half the frequency of ECLK. It is recommended that the CLKDIV module or PLL be used to generate the SCLK from the ECLK.

Figure 12-32 shows the primitive symbol for the IDDRX2B mode.

Figure 12-32. IDDRX2B Symbol

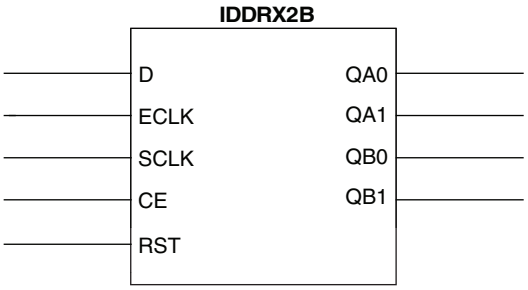


Table 12-9 lists the port names and descriptions for the IDDRX2B primitive.

Table 12-9. IDDRX2B Port Names

Port Name	I/O	Description
D	I	DDR data
ECLK	I	This clock can be connected to the fast edge clock
SCLK	I	This clock should be connected to the FPGA clock
CE	I	Clock enable signal
RST	I	Reset to the DDR register
QA0, QA1	O	Data at the positive edge of the clock
QB0, QB1	O	Data at the negative edge of the clock

Figure 12-33 shows the LatticeECP2 Input Register Block configured in the IDDRX2B mode. The DDR registers and the first set of synchronization registers are clocked by the ECLK input. The SCLK is used to clock the third stage of register. This primitive will output four streams of data. The 2x gearing function is implemented by using the synchronization registers of the complementary PIO. The clock transfer registers are shared with the output register block.

Figure 12-33. Input Register Block Configured as IDDRX2B

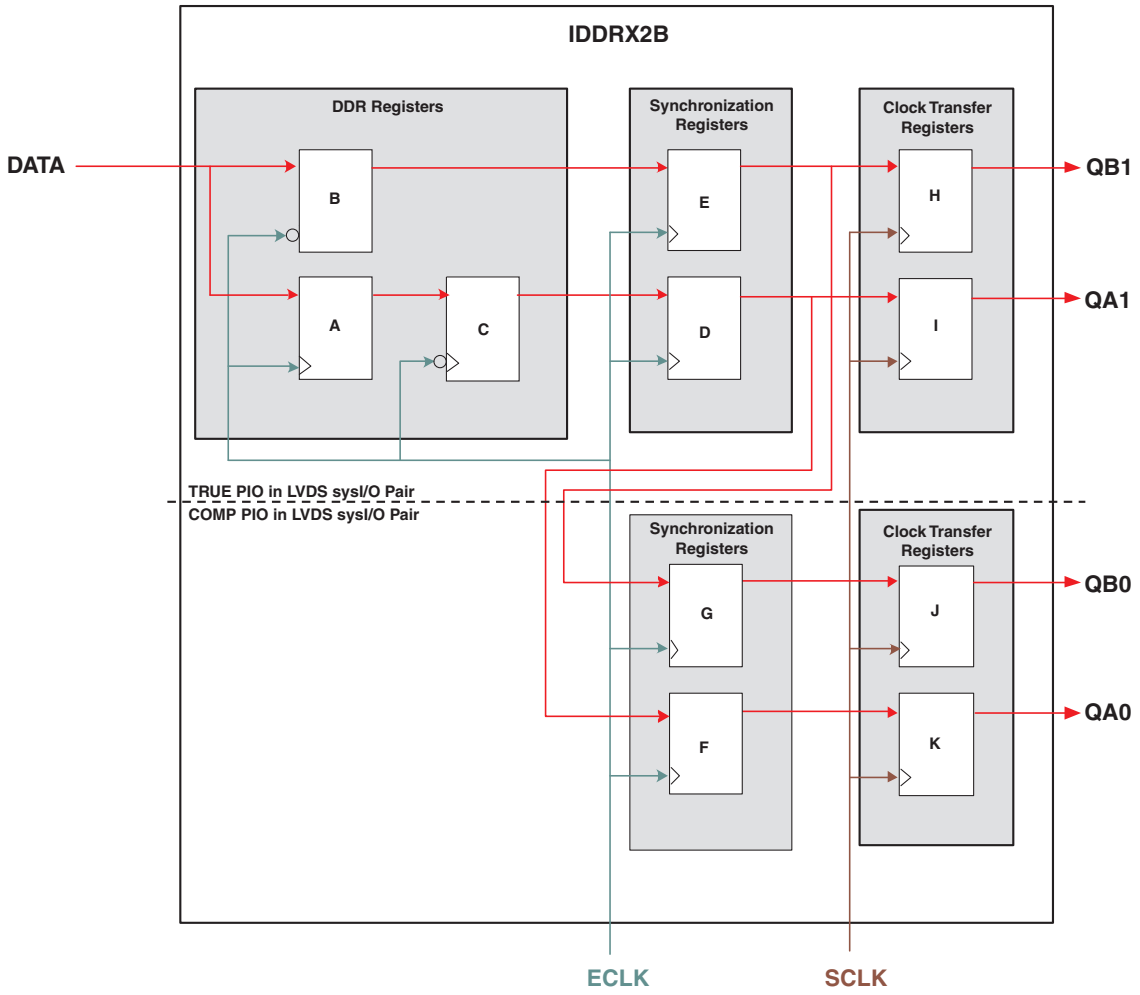
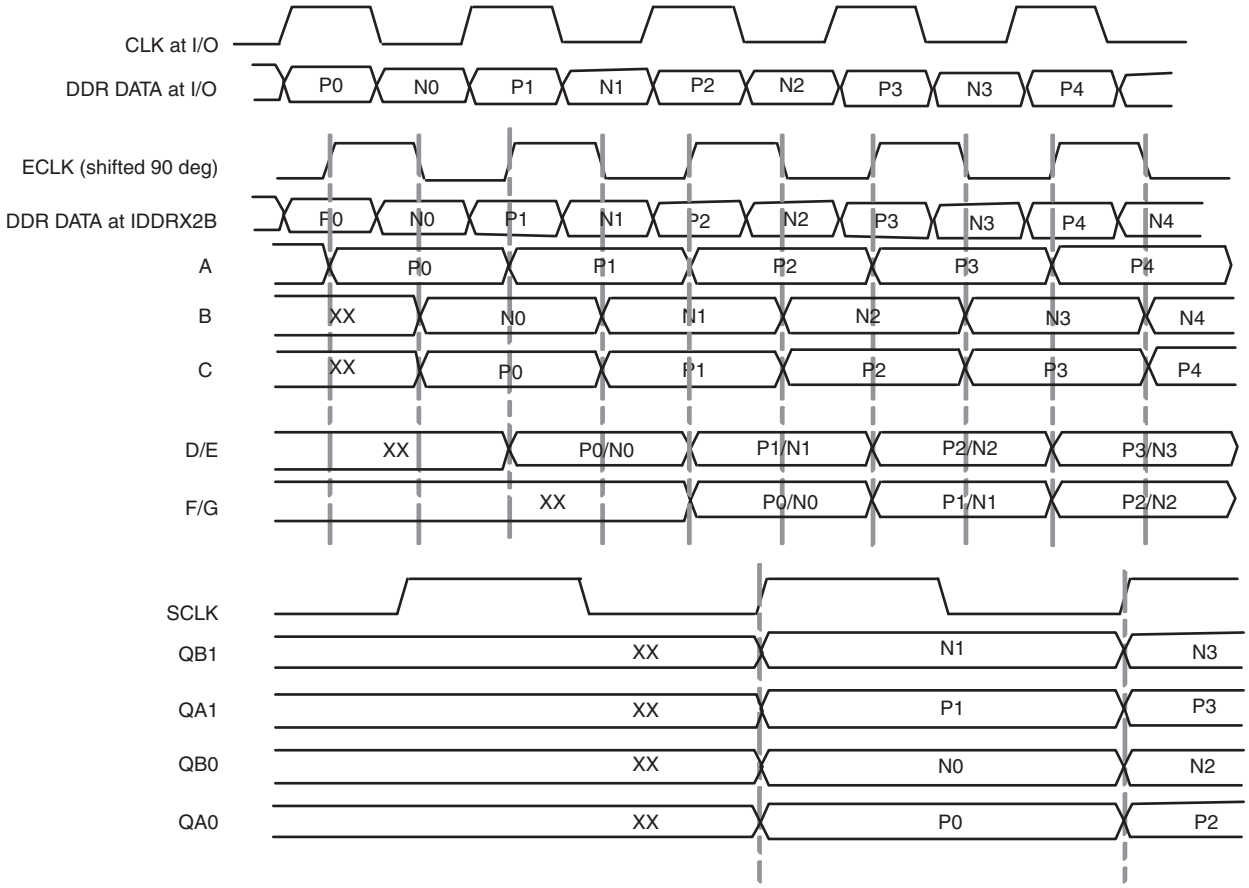


Figure 12-34 shows the timing waveform using the IDDRX2B module.

Figure 12-34. IDDRX2B Waveform



ODDRXC

This is the DDR output module. This primitive will input two data streams and mux them together to generate a single stream of data going to the sysIO™ buffer. The CLK to this module can be connected to the edge clock or to the FPGA clock. This primitive is also used for when DDR function is required for the tristate signal.

Figure 12-35 shows the primitive symbol for the ODDRXC mode.

Figure 12-35. ODDRXC Symbol

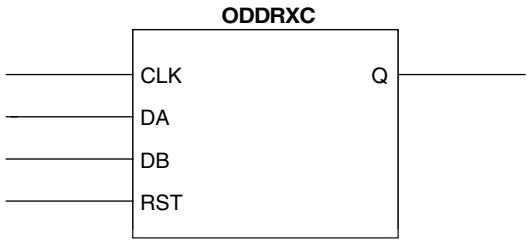


Table 12-10 lists the port names and descriptions for the ODDRXC primitive.

Table 12-10. ODDRXC Port Names

Port Name	I/O	Definition
DA	I	Data at the negative edge of the clock
DB	I	Data at the positive edge of the clock
CLK	I	This clock can be connected to the edge clock or to the FPGA clock
RST	I	Reset signal
Q	O	DDR data output

Figure 12-36 shows the Output Register Block of the LatticeECP2 device configured in ODDRXC mode.

Figure 12-36. Output Register Block in ODDRC Mode

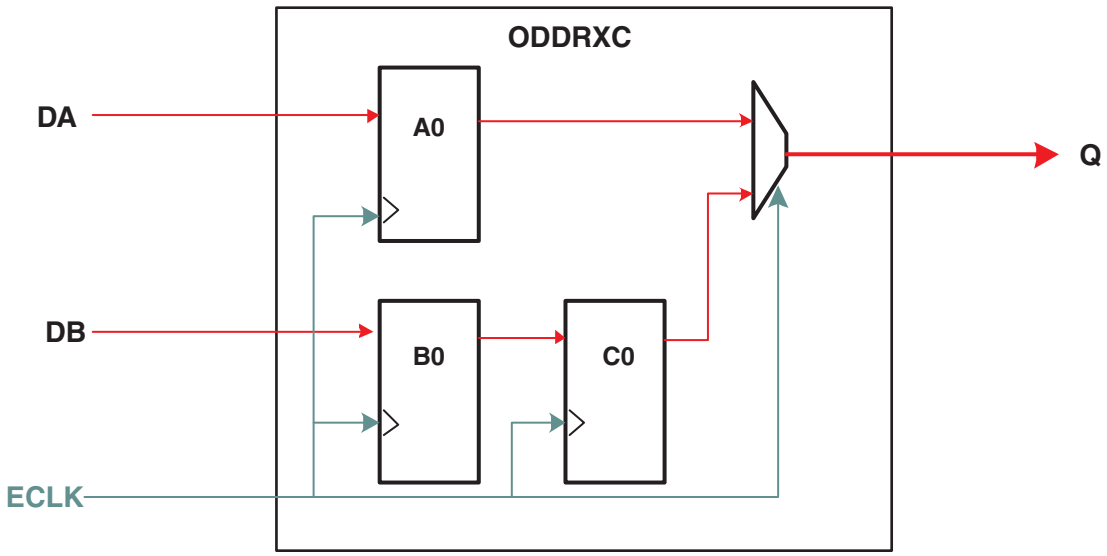
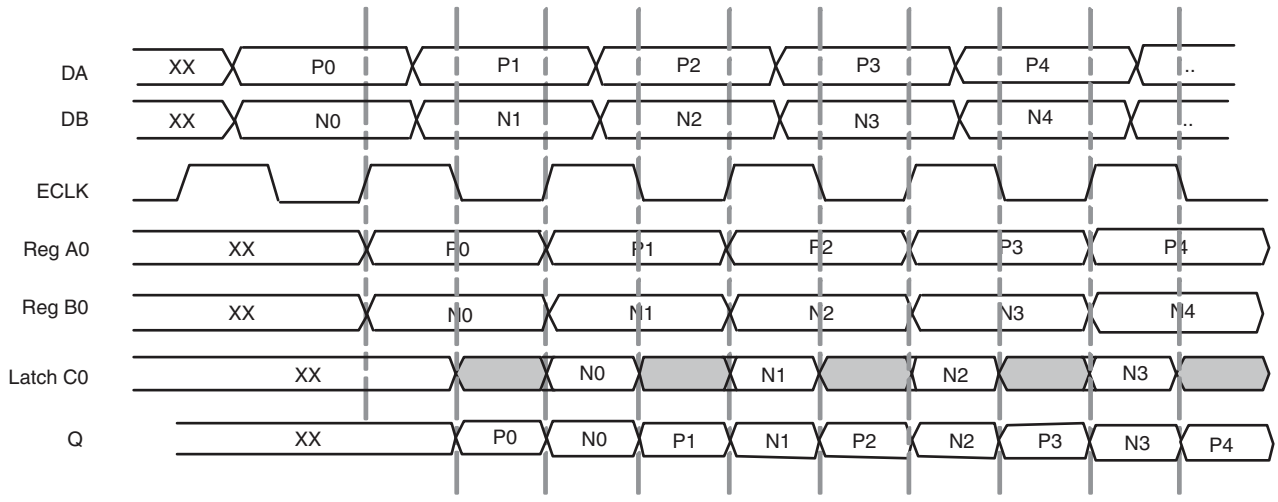


Figure 12-37 shows the timing waveform when using the ODDRXC module.

Figure 12-37. ODDRXC Waveform



ODDRX2B

This DDR output module can be used when a gearbox function is required. This primitive inputs four data streams and muxes them together to generate a single stream of data going to the sysIO buffer.

DDR registers of the complementary PIO are used in this mode. The complementary PIO register can no longer be used to perform the DDR function. There are two clocks going to this primitive. The ECLK is connected to the faster edge clock and the SCLK is connected to the slower FPGA clock. The DDR data output of this primitive is aligned to the faster edge clock.

Figure 12-38 shows the primitive symbol for the ODDRX2B mode.

Figure 12-38. ODDRX2B Symbol

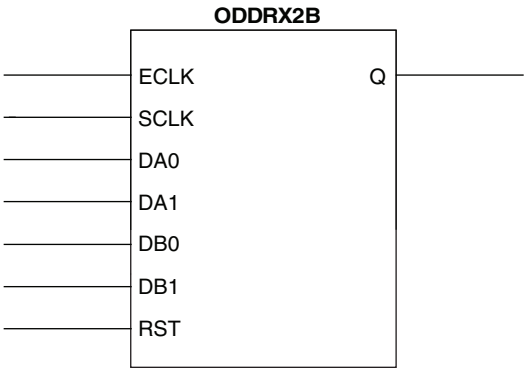


Table 12-11 lists the port names and descriptions for the ODDRX2B primitive.

Table 12-11. ODDRX2B Port Names

Port Name	I/O	Description
DA0, DB0	I	Data at the negative edge of the clock
DA1, DB1	I	Data at the positive edge of the clock
ECLK	I	This clock should be connected to the faster edge clock
SCLK	I	This clock should be connected to the slower FPGA clock
RST	I	Reset signal
Q	O	DDR data output

Figure 12-39 shows the LatticeECP2 Output Register Block in the ODDRX2B mode.

Figure 12-39. Output Register Block Configured in ODDRX2B Mode

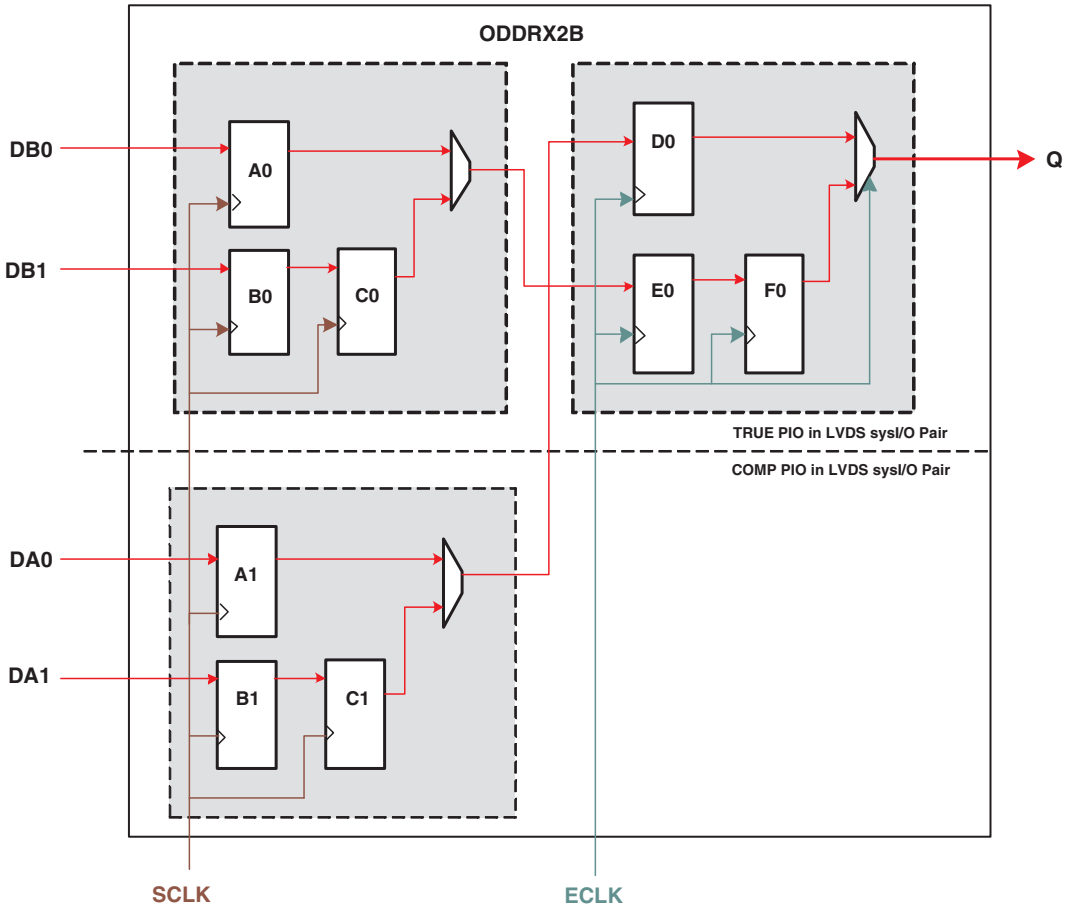
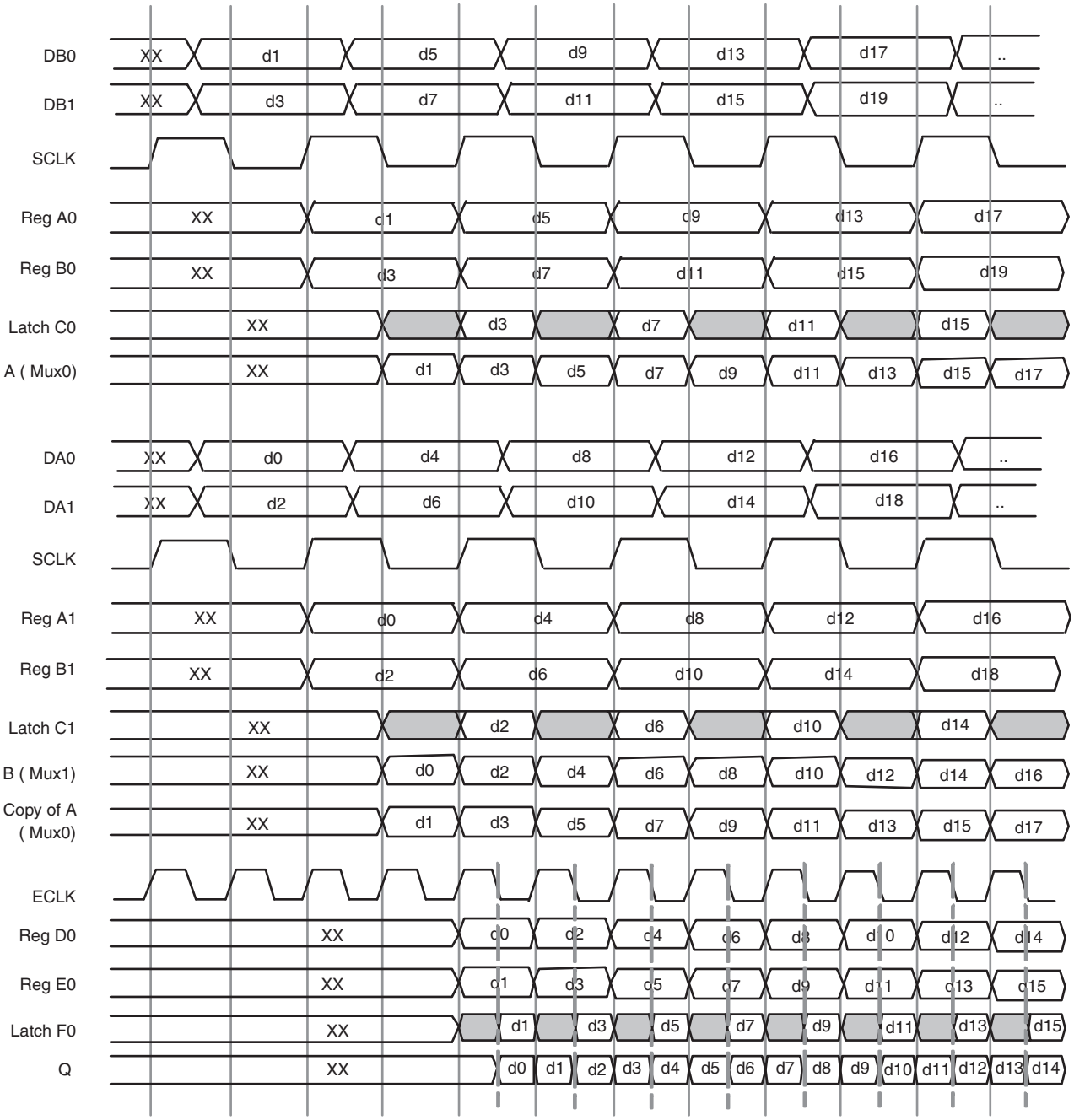


Figure 12-40 shows the timing waveform when using the ODDRXC module.

Figure 12-40. ODDRX2B Waveform



DELAYB

Data going to the DDR registers can be optionally delayed using the Delay block. The Delay block receives 4-bit delay control. The 4-bit delay can be set using fixed multiplier values or it can be controlled by the user. The DELAYB block is available for use with the input DDR registers.

The DELAYB block can be configured when generating the DDR input modules in the IPexpress tool of the software. The delay can be adjusted in 35ps steps. Users can choose from three types of delay values:

1. Dynamic – The delay value is controlled by the user logic using the DEL[3:0] input of the DELAYB block.

2. Fixed – When choosing the fixed value, the user will also need to choose from one of the 16 multiplier values. This will tie the inputs DEL[3:0] of the DELAYB block to a fixed value depending on the multiplier value chosen.
3. FIXED_XGMII – The DEL [3:0] will be configured with the delay value required when implementing a XGMII interface.

Figure 12-41 shows the primitive symbol for the DELAYB mode.

Figure 12-41. DELAYB Symbol

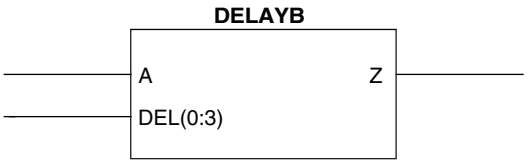


Table 12-12 lists the port names and descriptions for the DELAYB primitive.

Table 12-12. DELAYB Port Names

Port Name	I/O	Definition
A	I	DDR input from the sysIO buffer
DEL (0:3)	I	Delay inputs
Z	O	Delay DDR data

Design Rules/Guidelines

Listed below are some rules and guidelines for implementing generic DDR interfaces in LatticeECP2/M devices.

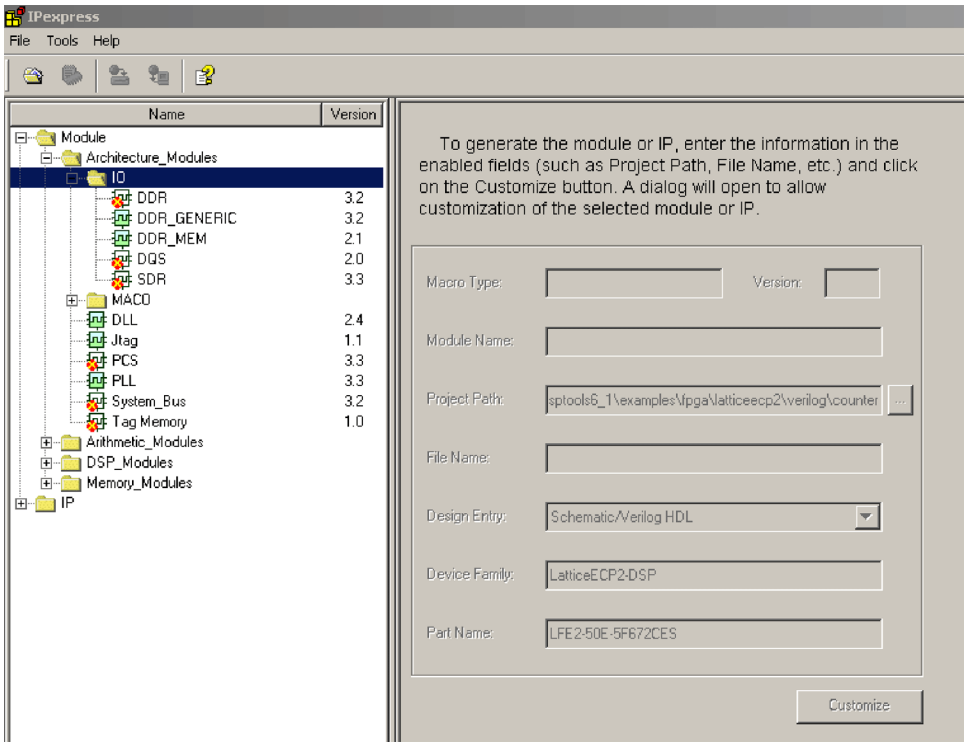
- When implementing a 2x gearing mode, the complement PIO registers are used. This complementary PIO register can no longer be used and should not be connected.
- DDR registers are available on the Left, Right and Bottom sides of the device. The top side does not support DDR registers.

DDR Usage in ispLEVER IPexpress

This section describes how IPexpress in ispLEVER is used to generate the DDR modules. If you are using Lattice Diamond™ design software, refer to Appendix A to see how DDR modules are generated in Diamond. IPexpress can be used to configure and generate the DDR Memory Interface and Generic DDR Module. The tool will generate an HDL module that will contain the DDR primitives. This module can be using in the top level design.

Figure 12-42 shows the main window of IPexpress. The DDR_Generic and DDR_MEM options under **Architecture->IO** are used to configure the DDR modules.

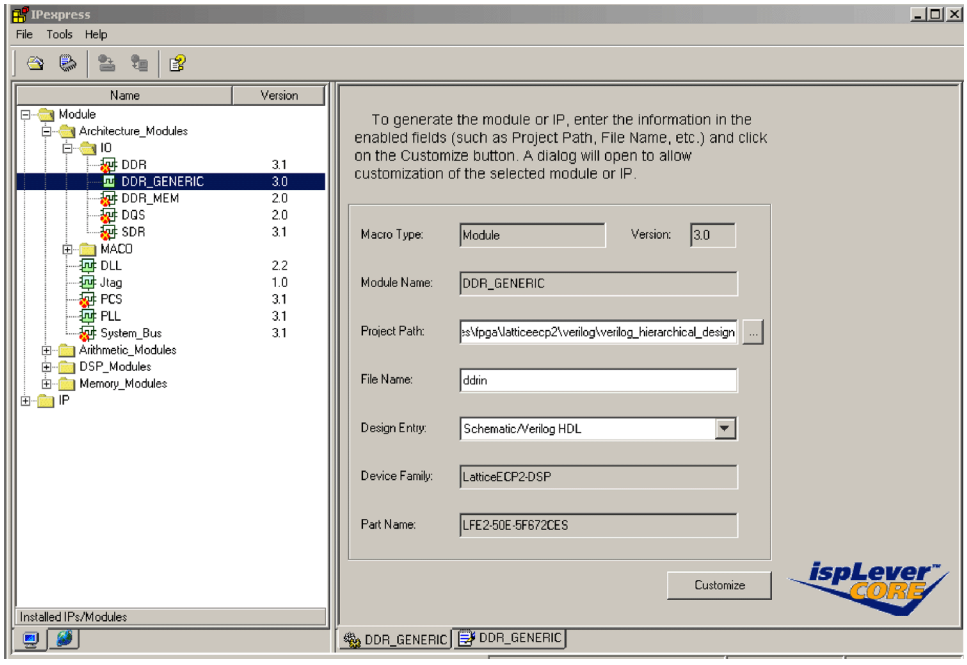
Figure 12-42. IPexpress Main Window



DDR Generic

Figure 12-43 shows the main window when DDR_Generic is selected. The only entry required in this window is the module name. Other entries are set to the project settings. The user may change these entries if desired. After entering the module name, click on **Customize** to open the **Configuration Tab** window as shown in Figure 12-44.

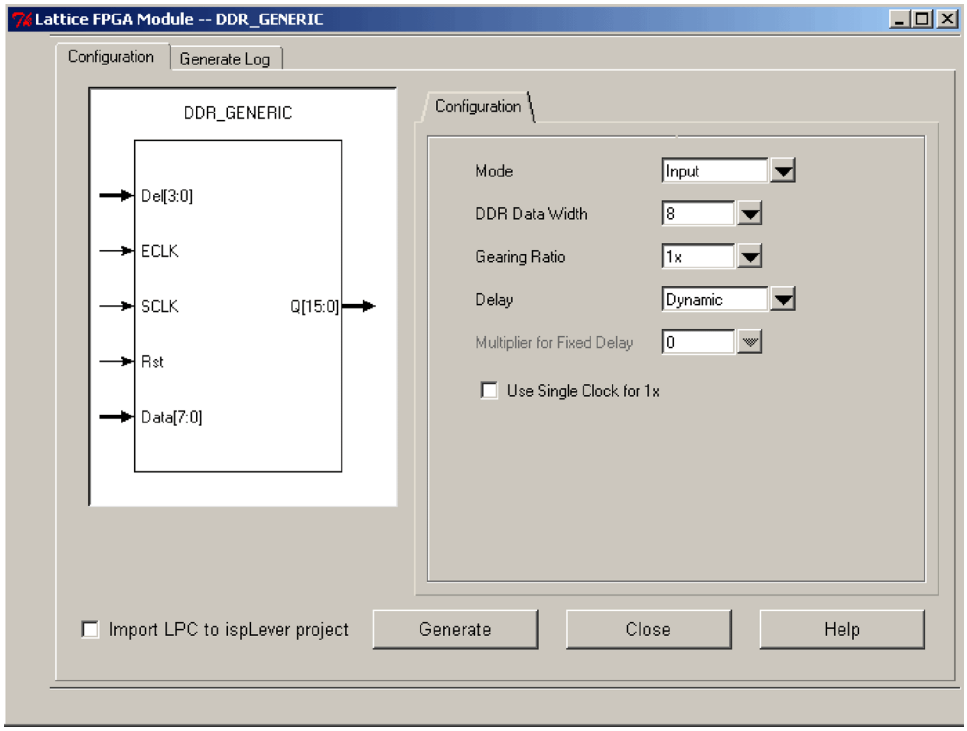
Figure 12-43. IPexpress Main Window for DDR_Generic



Configuration Tab

The Configuration Tab lists all user-accessible attributes with default values set. Upon completion, click **Generate** to generate source and constraint files. The user may choose to use the .lpc file to load parameters.

Figure 12-44. Configuration Tab for DDR_Generic



The user can change the Mode parameter to choose either Input, Output, Bidirection or Tristate DDR module. The other configuration parameters will change according to the mode selected. The Delay parameter is only available for Input and Bidirectional modes. Similarly the Multiplier for Fixed Delay parameter is only available when the Delay parameter is configured to Fixed.

Table 12-13. User Parameters in the IPexpress GUI

User Parameters	Description	Values/Range	Default
Mode	Mode selection for the DDR block.	Input, Output, Bidirectional, Tristate	Input
Data Width	Width of the data bus.	1-64	8
Gearing Ratio	Gearing ratio selection.	1x, 2x[1]	1x
Delay	Input delay configuration.	Dynamic, Fixed, Fixed XGMII	Dynamic
Multiplier for Fixed Delay	Fixed delay setting. Available only when delay is configured as Fixed.	0-15	0
Use Single Clk for 1x	Allows the selection of a single clock for the gearing logic.	On/Off	Off

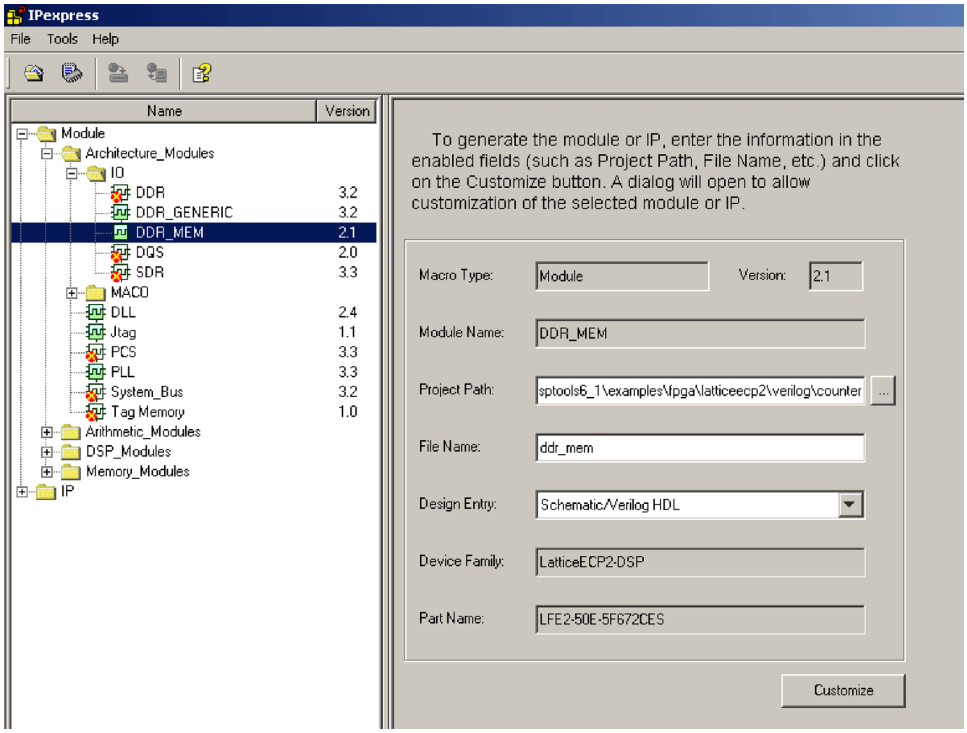
1. Only 1x available when Mode is Bidirection or Tristate.

DDR_MEM

Figure 12-45 shows the main window when DDR_MEM is selected. Similar to the DDR_Generic, the only entry required here is the module name. Other entries are set to the project settings. The user may change these entries if desired. After entering the module name, click on **Customize** to open the **Configuration Tab** window as shown

in Figure 12-46.

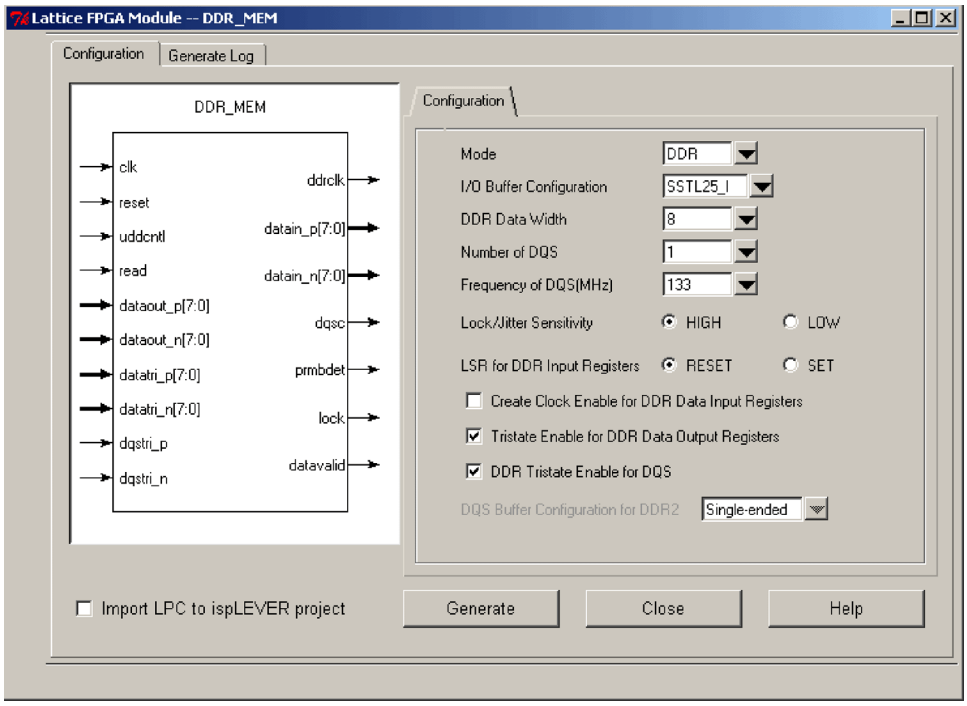
Figure 12-45. IPexpress Main Window for DDR_MEM



Configuration Tab

The Configuration Tab lists all user-accessible attributes with default values set. Upon completion, click **Generate** to generate source and constraint files. The user may choose to use the .lpc file to load parameters.

Figure 12-46. Configuration Tab for DDR_MEM



The user can change the Mode parameter to choose either the DDR or DDR2 interface. The other configuration parameters will change according to the Mode selected. The Number of DQS parameter determines the number of DDR interfaces. The software will assume there are eight data bits for every DQS. The user can also choose the frequency of operation and the DDR DLL will be configured to this frequency.

The user has an option to enable the clock enable and tristate enables for the DDR registers. It is recommend that the Lock/Jitter be enabled if the DDR interface is running at 150MHz or higher.

The parameters available depend on the mode selected. Tables 12-14 and 12-15 describe all user parameters in the IPexpress GUI and their usage for modes DDR and DDR2.

Table 12-14. User Parameters in the IPexpress GUI when in DDR Mode

User Parameters	Description	Values/Range	Default
I/O Buffer Configuration	I/O Standard used for the Interface. This will also depend on the Mode selected.	SSTL25_I, SSTL25_II	SSTL25_I
Data Width	Width of the Data bus	8-64	8
Number of DQS	Number of DQS will determine the number of DQS Groups	1, 2, 4, 8	1
Frequency of DQS	DDR Interface Frequency. This is also input to the DDR DLL. The values will depend on the mode selected.	100MHz, 133MHz, 166MHz, 200MHz	200MHz
Lock/Jitter Sensitivity	DLL Sensitivity to Jitter	High, Low	High
LSR for DDR Input Register	LSR Control	RESET, SET	RESET
Create Clock Enable for DDR Input Register	Create Clock enable inputs to the block	On/Off	Off
Tri-state Enable for DDR Output Registers	Creates Tri-state control for the DDR data output registers.	On/Off	On
DDR Tristate enable for the DQS output	Creates Tristate control for DQS output	On/Off	On

Table 12-15. User Parameters in the IPexpress GUI when in DDR2 Mode

User Parameters	Description	Values/Range	Default
I/O Buffer Configuration	I/O Standard used for the Interface. This will also depend on the Mode selected.	SSTL18_I, SSTL18_II	SSTL18_I
Data Width	Width of the Data bus	8-64	8
Number of DQS	Number of DQS will determine the number of DQS Groups	1, 2, 4, 8	1
Frequency of DQS	DDR Interface Frequency. This is also input to the DDR DLL. The values will depend on the mode selected.	166MHz, 200MHz, 266MHz	200MHz
Lock/Jitter Sensitivity	DLL Sensitivity to Jitter	High, Low	High
LSR for DDR Input Register	LSR Control	RESET, SET	RESET
Create Clock Enable for DDR Input Register	Create Clock enable inputs to the block	On/Off	Off
Tri-state Enable for DDR Output Registers	Creates Tri-state control for the DDR data output registers.	On/Off	On
DDR Tristate enable for the DQS output	Creates Tristate control for DQS output	On/Off	On
DQS Buffer Configuration for DDR2	DQS Buffer can be configured as Differential	On/Off	Off

FCRAM ("Fast Cycle Random Access Memory") Interface

FCRAM is a DDR-type DRAM, which performs data output at both the rising and falling edges of the clock. FCRAM devices operate at a core voltage of 2.5V with SSTL Class II I/O. It has enhanced both the core and peripheral logic of the SDRAM. In FCRAM the address and command signals are synchronized with the clock input, and the data pins are synchronized with the DQS signal. Data output takes place at both the rising and falling edges of the DQS. DQS is in phase with the clock input of the device. The DDR SDRAM and DDR FCRAM controller will have different pinouts.

LatticeECP2/M devices can implement the FCRAM interface using dedicated DQS logic, input DDR registers and output DDR registers, as described in the Implementing Memory Interfaces section of this document. Generation of address and control signals for FCRAM are different than in DDR SDRAM devices. Please refer to the FCRAM data sheets to see detailed specifications. Toshiba, Inc. and Fujitsu, Inc. offer FCRAM devices in 256Mb densities. They are available in x8 or x16 configurations.

Board Design Guidelines

The most common challenge associated with implementing DDR memory interfaces is the board design and layout. It is required that users strictly follow the guidelines recommended by memory device vendors.

Some of the common recommendations include matching trace lengths of interface signals to avoid skew, proper DQ-DQS signal grouping, proper termination of the SSTL2 or SSTL18 I/O Standard, proper VREF and VTT generation decoupling and proper PCB routing.

The following documents include board layout guidelines:

- www.idt.com, IDT, PCB Design for Double Data Rate Memory
- www.motorola.com, AN2582, Hardware and Layout Design Considerations for DDR Interfaces

References

- www.jedec.org, JEDEC Standard 79, Double Data Rate (DDR) SDRAM Specification
- www.micron.com, DDR SDRAM Data Sheets

- www.infinion.com, DDR SDRAM Data Sheets
- www.samsung.com, DDR SDRAM Data Sheets
- www.toshiba.com, DDR FCRAM Data Sheet
- www.fujitsu.com, DDR FCRAM Data Sheet
- RD1019, QDR Memory Controller Reference Design
- EB23, LatticeECP2 Advanced Evaluation Board User's Guide
- IPUG35, DDR1 & DDR2 SDRAM Controller (Pipelined Versions) User's Guide

Technical Support Assistance

e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

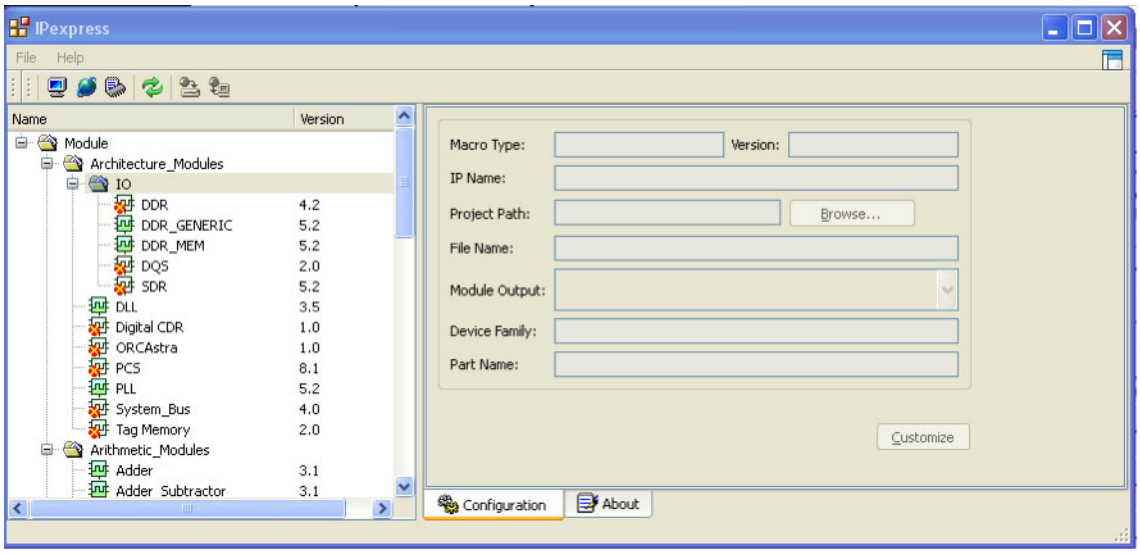
Date	Version	Change Summary
February 2006	01.0	Original version.
September 2006	01.1	Some figures updated. Added information on DELAYB block.
September 2006	01.2	Updated for LatticeECP2M. Added "DDR Generic Usage in IPexpress" section.
February 2007	01.3	Updated DDR and DDR2 SDRAM Interfaces Overview section.
June 2007	01.4	Updated the port names on the input DDR block diagrams.
		Updated text in the DQS Transition Detect under Memory Read Implementation section.
		Updated the DQSDEL from 6-bit bus to single bit in figures 12-19 and 12-20.
		Updated text in the DLL Compensated DQS Delay Elements section under Memory Read Implementation.
October 2007	01.5	Updated DDRX2B Waveform diagram.
June 2009	01.6	Updated DQSDLL Update Control text section.
January 2010	01.7	Updated Read Data Transfer When Using IDDRMFX1A figure.
		Updated Data Read Critical Path text section.
June 2010	01.8	Added Appendix A.
June 2013	01.9	Updated document with new corporate logo.
		Updated Technical Support Assistance information.

Appendix A. DDR Generation Using IPexpress with Lattice Diamond

The IPexpress tool in the Lattice Diamond design software can be used to configure and generate the DDR Memory Interface and Generic DDR Module. IPexpress will generate an HDL module that will contain the DDR primitives. This module can be using in the top level design.

Figure 12-47 shows the main window of IPexpress. The DDR_Generic and DDR_MEM options under **Architecture > IO** are used to configure the DDR modules.

Figure 12-47. IPexpress Main Window

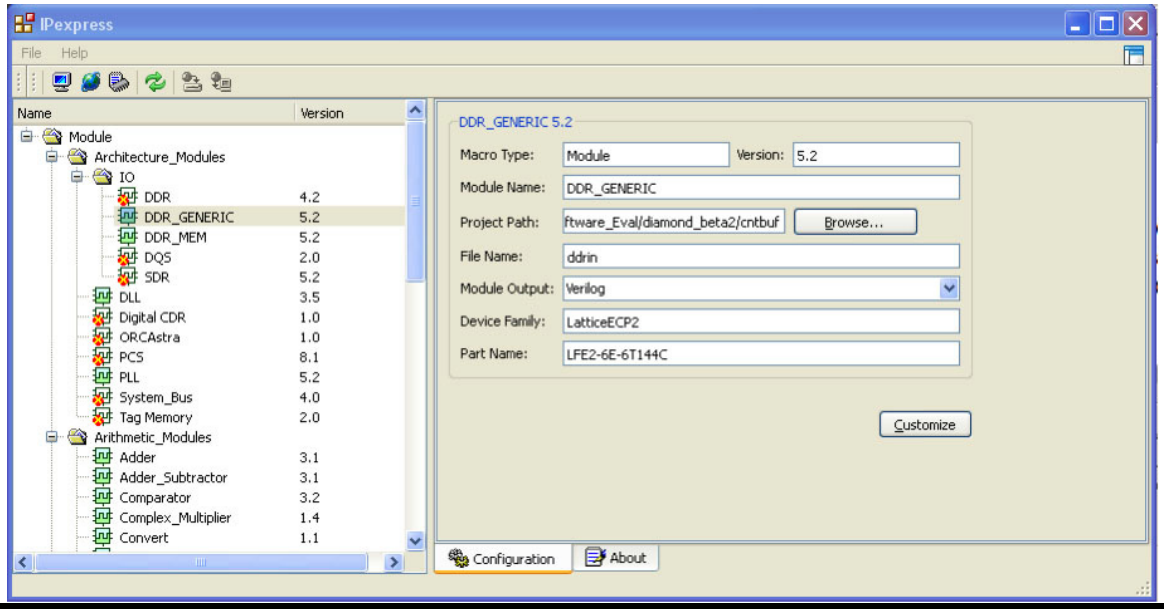


DDR Generic

Figure 12-48 shows the main window when DDR_Generic is selected. The only entry required in this window is the module name. Other entries are set to the project settings.

The user may change these entries if desired. After entering the module name, click on **Customize** to open the Configuration Tab window as shown in Figure 12-49.

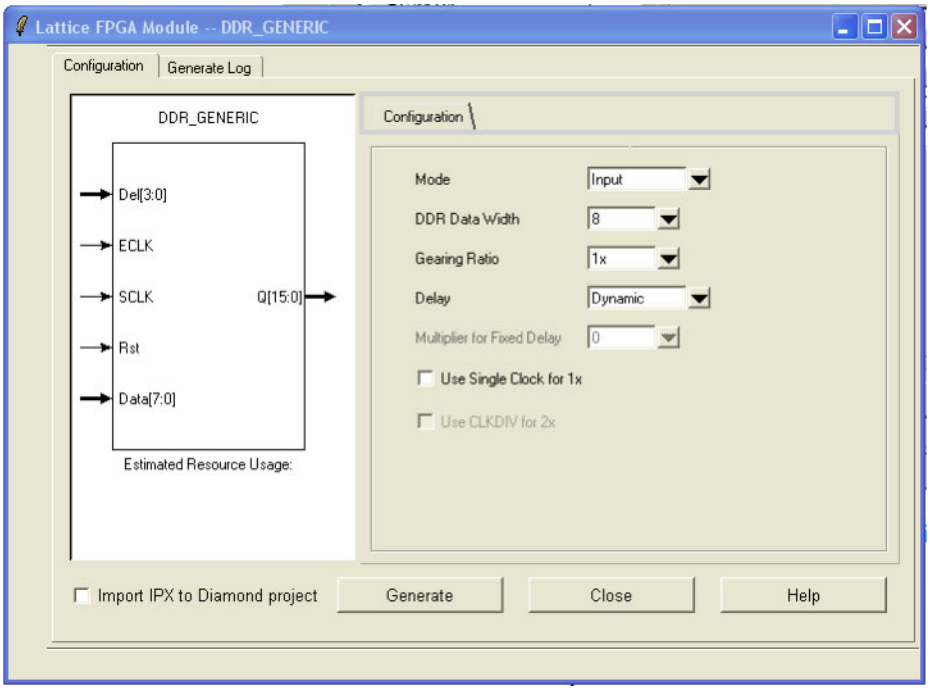
Figure 12-48. IPexpress Main Window for DDR_Generic



Configuration Tab

The Configuration Tab lists all user-accessible attributes with default values set. Upon completion, click **Generate** to generate source and constraint files. The user may choose to use the .lpc file to load parameters.

Figure 12-49. Configuration Tab for DDR_Generic



The user can change the Mode parameter to choose either the Input, Output, Bi-directional or Tristate DDR modules. The other configuration parameters will change according to the mode selected. The Delay parameter is only available for Input and Bi-directional modes. Similarly, the Multiplier for Fixed Delay parameter is only available when the Delay parameter is configured as Fixed.

Table 12-16. User Parameters in the IPexpress GUI

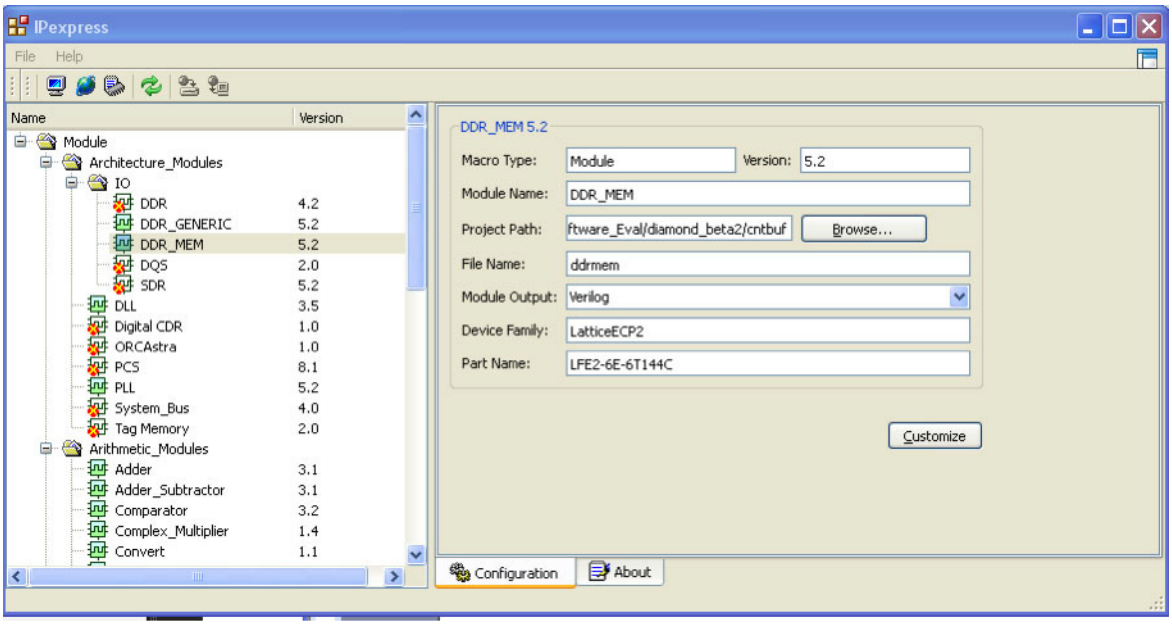
User Parameters	Description	Values/Range	Default
Mode	Mode selection for the DDR block.	Input, Output, Bidirectional, Tristate	Input
Data Width	Width of the data bus.	1-64	8
Gearing Ratio	Gearing ratio selection.	1x, 2x[1]	1x
Delay	Input delay configuration.	Dynamic, Fixed, Fixed XGMII	Dynamic
Multiplier for Fixed Delay	Fixed delay setting. Available only when delay is configured as Fixed.	0-15	0
Use Single Clk for 1x	Allows the selection of a single clock for the gearing logic.	On/Off	Off

1. Only 1x available when Mode is Bi-directional or Tristate.

DDR_MEM

Figure 12-50 shows the main window when DDR_MEM is selected. Similar to DDR_Generic, the only entry required here is the module name. Other entries are set to the project settings. The user may change these entries if desired. After entering the module name, click on **Customize** to open the Configuration Tab window as shown in Figure 12-51.

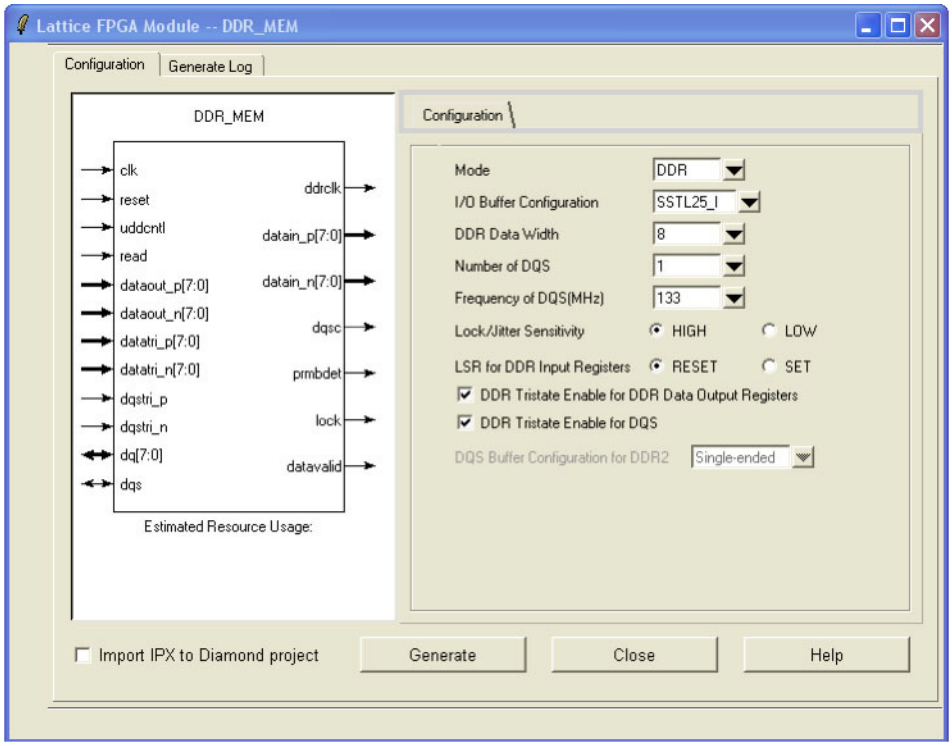
Figure 12-50. IPexpress Main Window for DDR_MEM



Configuration Tab

The Configuration Tab lists all user-accessible attributes with default values set. Upon completion, click **Generate** to generate source and constraint files. The user may choose to use the .lpc file to load parameters.

Figure 12-51. Configuration Tab for DDR_MEM



The user can change the Mode parameter to choose either the DDR or DDR2 interface. The other configuration parameters will change according to the Mode selected. The Number of DQS parameters determines the number

of DDR interfaces. The software will assume there are eight data bits for every DQS. The user can also choose the frequency of operation and the DDR DLL will be configured to this frequency.

The user has the option to enable the clock enable and tristate enables for the DDR registers. It is recommend that the Lock/Jitter be enabled if the DDR interface is running at 150 MHz or higher.

The parameters available depend on the mode selected. Tables 12-17 and 12-18 describe all user parameters in the IPexpress GUI and their usage for modes DDR and DDR2.

Table 12-17. User Parameters in the IPexpress GUI when in DDR Mode

User Parameters	Description	Values/Range	Default
I/O Buffer Configuration	I/O Standard used for the Interface. This will also depend on the Mode selected.	SSTL25_I, SSTL25_II	SSTL25_I
Data Width	Width of the Data bus	8-64	8
Number of DQS	Number of DQS will determine the number of DQS Groups	1, 2, 4, 8	1
Frequency of DQS	DDR Interface Frequency. This is also input to the DDR DLL. The values will depend on the mode selected.	100MHz, 133MHz, 166MHz, 200MHz	200MHz
Lock/Jitter Sensitivity	DLL Sensitivity to Jitter	High, Low	High
LSR for DDR Input Register	LSR Control	RESET, SET	RESET
Create Clock Enable for DDR Input Register	Create Clock enable inputs to the block	On/Off	Off
Tri-state Enable for DDR Output Registers	Creates Tri-state control for the DDR data output registers.	On/Off	On
DDR Tristate enable for the DQS output	Creates Tristate control for DQS output	On/Off	On

Table 12-18. User Parameters in the IPexpress GUI when in DDR2 Mode

User Parameters	Description	Values/Range	Default
I/O Buffer Configuration	I/O Standard used for the Interface. This will also depend on the Mode selected.	SSTL18_I, SSTL18_II	SSTL18_I
Data Width	Width of the Data bus	8-64	8
Number of DQS	Number of DQS will determine the number of DQS Groups	1, 2, 4, 8	1
Frequency of DQS	DDR Interface Frequency. This is also input to the DDR DLL. The values will depend on the mode selected.	166MHz, 200MHz, 266MHz	200MHz
Lock/Jitter Sensitivity	DLL Sensitivity to Jitter	High, Low	High
LSR for DDR Input Register	LSR Control	RESET, SET	RESET
Create Clock Enable for DDR Input Register	Create Clock enable inputs to the block	On/Off	Off
Tri-state Enable for DDR Output Registers	Creates Tri-state control for the DDR data output registers.	On/Off	On
DDR Tristate enable for the DQS output	Creates Tristate control for DQS output	On/Off	On
DQS Buffer Configuration for DDR2	DQS Buffer can be configured as Differential	On/Off	Off