

This application note describes how to implement the delay-locked loop (DLL) phase offset feature with Altera® Stratix® FPGAs and HardCopy® ASICs.

Introduction

A DLL provides a process, voltage, and temperature (PVT)-compensated delay that you can use to phase shift the read clock from an external memory to align it with the center of the data valid window. The DLL phase offset feature provides a method to make fine non-PVT-compensated phase adjustments to the read clock from an external memory. If the circuit board or memory timing specifications are different than expected, you can use the DLL phase offset feature to optimize the read capture timing.

This application note contains examples of changing the phase offset when you are using one of the following MegaCore® functions or megafunctions:

- DDR3/DDR2/DDR High Performance controller
- DDR2 Legacy Integrated Static Data Path and Controller for HardCopy II ASICs (referred to as the Legacy Static PHY)
- ALTMEMPHY megafunction
- ALTDQS megafunction

Included in this application note are example files that show how to implement a DLL phase offset with the ALTMEMPHY megafunction and Legacy Static PHY.

This application note contains the following sections:

- “DLL Overview” on page 1
- “DLL Phase Offset Control” on page 2
- “Using TimeQuest’s set_annotated_delay to Analyze the Effect of the DLL Phase Offset Feature” on page 5
- “Using DLL Phase Offset with the ALTMEMPHY Megafunction” on page 7
- “Instantiating a DLL with Altera IP” on page 8
- “Confirm Your DLL Phase Offset Control Settings” on page 29

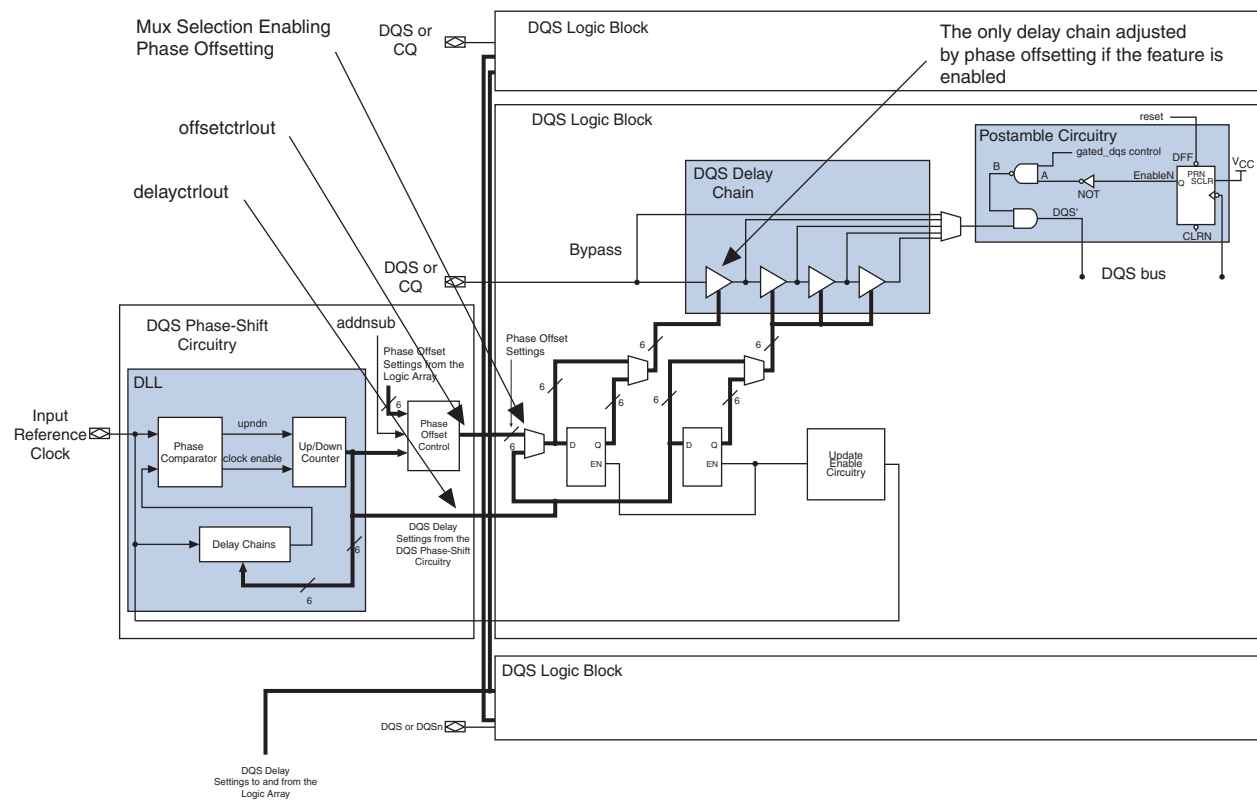
DLL Overview

Altera’s external memory interface read capture logic is composed of the DLL and its phase offset control block, the DQS logic block, and the DQ blocks. The DLL automatically adjusts its delay chains to have a delay equal to one cycle of its input clock.

The DLL outputs a value on the `delayctrlout` bus based on its phase shift setting. The `delayctrlout` bus goes to the phase offset control block and to the DQS blocks. In the DQS blocks, the value on `delayctrlout` selects the number of delay buffers used to phase shift the read clock from memory. In the phase offset control block, the value on the bus offset is added to or subtracted from `delayctrlout`, depending on the `addnsub` signal. This value, called `offsetctrlout`, goes to the DQS blocks and selects the number of delay buffers in only one of the four DQS delay chains.

Figure 1 shows an overview of the DLL/DQS logic block for all Stratix and HardCopy series devices.

Figure 1. DLL/DQS Logic Block for Stratix and HardCopy Devices



DLL Phase Offset Control

The DLL phase offset feature allows you to add or subtract a non-PVT-compensated offset to the DQS phase shift that is used to capture read data inside the device. Depending on the HardCopy device used, each DLL has one or two phase offset modules that can provide DQS delay settings with an independent offset, so you can fine-tune the DQS phase shift settings.

If you find bit errors in the HardCopy ASIC DDR interface as a result of a change to the board design (which changes board latency) or to the memory vendor's original specification, you can fine tune the DQS phase shift to accommodate the changes. For example, if a 90° DQS phase shifted DQ read capture is generating intermittent bit errors in corner-case conditions of operating temperature and/or voltage, a firmware change on the device could adjust the phase shift to a new value (for example, 85°) that would allow the device to properly capture the DDR read data. The main settings of the DLL do not allow a phase shift of 85°. Using the DLL phase offset feature allows for fine tuning the phase shift.

Figure 2 shows how the DQS inside a device changes with offset adjustment.


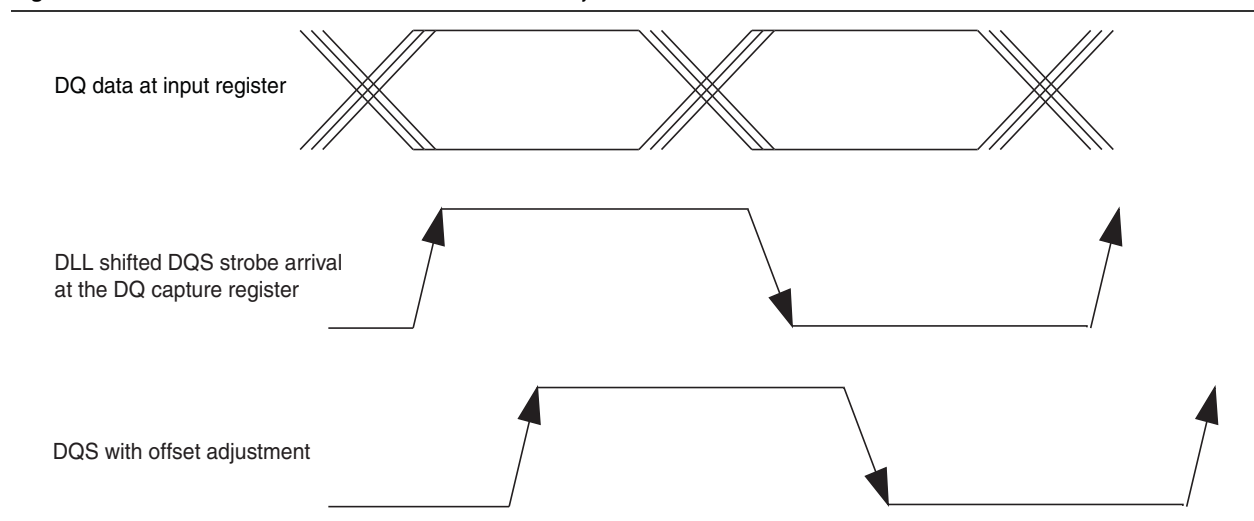
 The proximity of the edge without offset adjustment to the data changing is exaggerated for the purpose of illustration.

Figure 2. DQS Inside Device With and Without Phase Adjustment



The DLL phase offset is controlled by two signals, `addnsub` and `offset [5:0]`. The `addnsub` signal controls whether the value on `offset [5:0]` is added to or subtracted from the default delay value. You can control the values of `addnsub` and `offset [5:0]` in a variety of ways. For example, they can be controlled by an addressable port in a microprocessor's address space, by part of a scan chain that controls PLL reconfiguration, or by connecting directly to pins that are connected to a dual in-line package (DIP) switch. Another method of controlling phase offset is to use In-System Sources and Probes. You can drive the `addnsub` and `offset [5:0]` signals and you can monitor `delayctrlout [5:0]`.

 For more information about In-System Sources and Probes, refer to the *Design Debugging Using In-System Sources and Probes* chapter in volume 3 of the *Quartus II Handbook*.

When `addnsub` is high, the value specified in `offset [5:0]` multiplied by the delay in Table 2 on page 4 is included as additional uncompensated delay to the DQS phase shift. When `addnsub` is low, its 2's complement ($!\text{offset [5:0]} + 1$) multiplied by the delay in Table 2 on page 4 is subtracted from the DQS phase shift (for example, to subtract 1, set `offset` to 11111).

It might be easier to think of $\{ \text{!addnsub}, \text{offset}[5:0] \}$ as a 7-bit 2's complement value for the offset with the sign bit, `addnsub`, inverted. To add 11 (decimal), set `addnsub` to 1 and `offset[5:0]` to 001011. To subtract 9, set `addnsub` to zero. For `offset[5:0]`, take 9 (001001), its ones complement is 110110. Its 2's complement is $110110 + 1 = 110111$.

Table 1 lists the frequency modes, frequency ranges, and the number of delay chains used in Stratix II FPGAs and HardCopy II ASICs.

Table 1. Frequency Modes, Ranges, and Delay Chains for –4 Speed Grade of Stratix II FPGAs and HardCopy II ASICs

Frequency Mode	Frequency Range (MHz)	Available Phase Shift	Number of Delay Chains
0	100 to 175	30, 60, 90, 120	12
1	150 to 230	22.5, 45, 67.5, 90	16
2	200 to 310	30, 60, 90, 120	12
3	240 to 350	36, 72, 108, 144	10


 For frequency modes, ranges, and delay chains of Stratix III, Stratix IV, HardCopy III, and HardCopy IV devices, refer to the *DC and Switching Characteristics* chapter in their respective handbooks.


Table 2 lists the allowable DQS offset delay per stage for the various device speed grades for Stratix II FPGAs. The valid `offset[]` ranges are –64 to +63 when you configure the DLL and DQS delay buffers in low frequency mode (DLL frequency mode 0) and –32 to +31 when you configure the DLL and DQS delay buffers in high frequency mode (DLL frequency modes 1, 2, and 3).

Table 2. DQS Phase Offset Delay Per Stage for Stratix II FPGAs (Note 1), (2)

Speed Grade	Minimum	Maximum	Unit
–3	9	14	ps
–4	9	14	ps
–5	9	15	ps

Notes to Table 2:

- (1) The delay settings are linear.
- (2) The typical value equals the average of the minimum and maximum values.

 For the DQS phase offset delay per stage for Stratix III, Stratix IV, and HardCopy III devices, refer to the *DC and Switching Characteristics* chapter in their respective handbooks.

The following example is based on the DQS phase offset delay per stage for Stratix II FPGAs. You can use the same method for Stratix III and Stratix IV FPGAs and HardCopy III and HardCopy IV ASICs. If you are using frequency mode 1 for a 200-MHz DDR interface and the DQS DLL is set to 90°, and you want to shift the DQS to 85° to add an additional 5° of hold time margin, perform the following steps to determine the setting:

1. For a frequency of 200 MHz, the period is 5 ns, or 5,000 ps. The delay (in ps-per-degree) = $5000 \text{ ps} / 360 \text{ degrees} = 13.88 \text{ ps/degree}$. Because you want to shift the DQS by 5°, $5 \text{ degrees} \times 13.88 \text{ ps/degree} = 69.4 \text{ ps}$.

2. To achieve a shift of 69.4 ps, divide 69.4 by the average of the minimum and maximum delay per stage, which is 11.5. This gives you 6.03 (round to 6). The 2's complement of 6 is the ones complement plus one, so $6 = 000110$, its ones complement is 111001 and its 2's complement is $111001+1$, which is 111010 .
3. Set `addnsub low` and set `offset [5:0] = 111010`. Setting `offset [5:0]` to the 2's complement of 6 reduces the delay in the range of $9 \times 6 = 54$ ps to $14 \times 6 = 84$ ps in a -3 or -4 Stratix II FPGA. It is in the same range for a HardCopy II ASIC.

The DLL makes a determination of what the delay settings must be to achieve the requested phase shift at the capture register. For example, if you set a phase shift of 30° , the DLL might produce a delay setting of 28. This is the number of delay buffers to use in the delay chains. In this case, you cannot subtract more than 28 using the DLL phase offset feature. Doing so produces a delay setting that selects fewer than zero delay buffers, which is physically impossible.

The DQS delay buffer also has a maximum setting that depends on the DLL's `delayctrlout` output; the offset is added to the DLL's `delayctrlout`, so the offset cannot be greater than the maximum setting (+63 for DLL frequency mode 0 and +31 for DLL frequency modes 1, 2, and 3) minus the `delayctrlout` setting. The port `delayctrlout [5:0]` can be connected to your logic so that you can observe its value in order to determine the values that can be used for the offset. The value on `delayctrlout [5:0]` is not static; it can vary with changes in voltage and temperature. Using the value on `delayctrlout [5:0]` allows you to calculate the maximum value you can effectively use on `offset [5:0]`. Altera recommends using a mode for which the frequency is closer to the middle of the range rather than closer to the end of the range. This allows the most adjustment range to add or subtract delay. For example, if your frequency is 200 MHz, mode 1 is a better choice than mode 2.



For more information about frequency ranges, refer to the "DLL Frequency Range Specifications" table in the *DC and Switching Characteristics* chapter in volume 1 of the *Stratix II Device Handbook*.

Using TimeQuest's set_annotated_delay to Analyze the Effect of the DLL Phase Offset Feature

Use the TimeQuest Timing Analyzer to annotate the delay between two adjacent points on a path. To analyze timing, first calculate the offset value. Report the timing at the slow operating condition from the DQS pin to *. Here is the TimeQuest Timing Analyzer command:

```
report_path -from [get_ports [mem_dqs[0]]] -to * -npaths 10 \
-panel_name {Report Path}
```

Figure 3 shows an example of a situation that requires an earlier shift capture.

Figure 3. Example of a Situation that Requires the Shift Capture Point 5° Earlier

Path #1: Setup slack is 0.365							
Path Summary		Statistics	Data Path	Waveform			
Data Arrival Path							
Total	Incr	RF	Type	Fanout	Location	Element	
1	0.000					launch edge time	
2	0.000	0.000	R			clock network delay	
3	0.515	0.515	R	Ext	PIN_A18	ddr2_dq[6]	
4	2.555	2.040	RR	CELL	IOC_X17_Y37_N1	..._dqs_group\g_datapath:0:g_ddr_ioldq_captured_falling[6]	
Data Required Path							
Total	Incr	RF	Type	Fanout	Location	Element	
1	0.000	0.000				latch edge time	
2	0.000	0.000				source latency	
3	0.000	0.000		1	PIN_B15	ddr2_dqs[0]	
4	1.171	1.171	FF	CELL	IOC_X23_Y37_N0	...instlddr_intf_auk_ddr_datapath:ddr_iolddr_intf_auk_ddr_dqs_group\g_datapath:0:g_ddr_ioldqs_io~comb_dqs	
5	2.296	1.125	FF	CELL	IOC_X23_Y37_N0	ddr_intf_ddr_sdramlddr_intf_auk_ddr_sdram_instlddr_iol\g_datapath:0:g_ddr_ioldqs_ioldqsbusout	
6	2.596	0.300	FF	IC	IOC_X17_Y37_N1	ddr_intf_ddr_sdramlddr_intf_auk_ddr_sdram_instlddr_iol\g_datapath:0:g_ddr_iol\g_dq_io:6:dq_iolinc1k	
7	3.060	0.464	RR	CELL	IOC_X17_Y37_N1	...dr_intf_auk_ddr_datapath:ddr_iolddr_intf_auk_ddr_dqs_group\g_datapath:0:g_ddr_ioldq_captured_falling[6]	
8	2.920	-0.140		1	IOC_X17_Y37_N1	...dr_intf_auk_ddr_datapath:ddr_iolddr_intf_auk_ddr_dqs_group\g_datapath:0:g_ddr_ioldq_captured_falling[6]	

This is the arc affected by the DLL phase offset so the timing annotation is done here.

In the example shown in Figure 3, refer to the DQS phase offset delay values (Table 2) and shift the capture point 5° earlier. Use the following procedure to calculate the offset value (in ns):

1. Set the operating conditions to slow:

- a. Calculate the minimum delay:

$$1.125 - (5 \times 0.014) = 1.055$$

- b. Calculate the maximum delay:

$$1.125 - (5 \times 0.009) = 1.080$$

2. Set the new delay that includes the offset for slow conditions:

```
set_annotated_delay -max 1.080 -min 1.055 \
-from *g_datapath:*:g_ddr_io|dqs_io~comb_dqs \
-to *g_datapath:*:g_ddr_io|dqsbusout -rr -ff
```

3. Report timing at the slow conditions.

4. Set the operating conditions to fast. Assume the delay at the fast operating condition is 0.563, then:

- a. Calculate the maximum delay:

$$0.563 - (5 \times 0.009) = 0.518$$

- b. Calculate the minimum delay:

$$0.563 - (5 \times 0.014) = 0.493$$

5. Set the new delay that includes the offset for the fast conditions:

```
set_annotated_delay -max 0.518 -min 0.493 \
-from *g_datapath:*:g_ddr_io|dqs_io~comb_dqs \
-to *g_datapath:*:g_ddr_io|dqsbusout -rr -ff
```

6. Report timing at the fast conditions.

Using DLL Phase Offset with the ALTMEMPHY Megafunction

The system block diagram in Figure 4 shows how the DLL, ALTMEMPHY megafunction, and user logic interact.

Figure 4. DLL Phase Offset System View

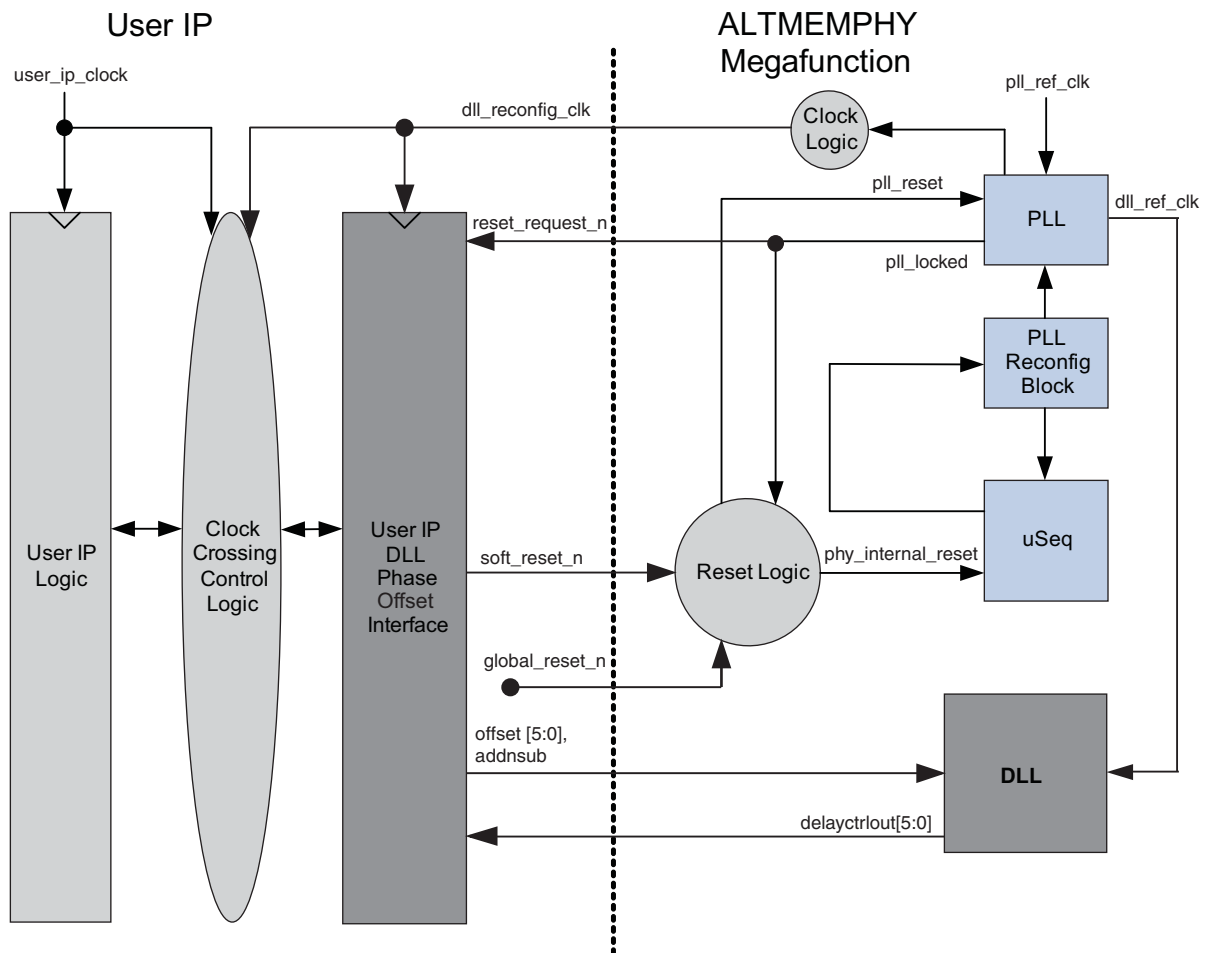
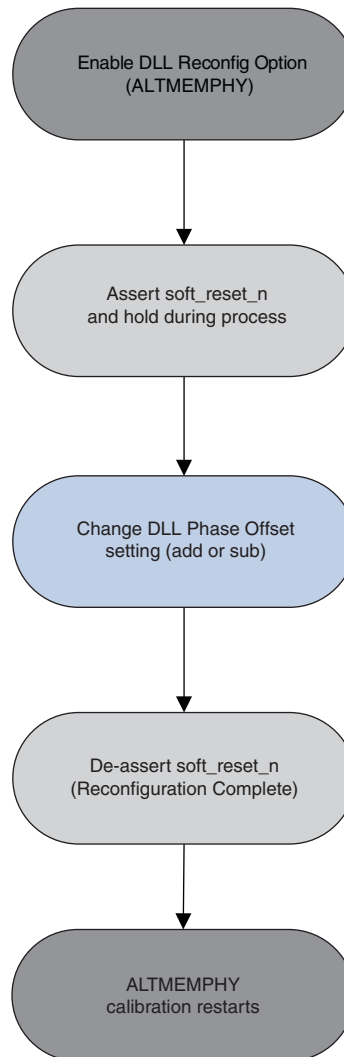


Figure 5 shows the sequence you must follow after changing the DLL phase offset.

Figure 5. DLL Phase Offset Flow



Instantiating a DLL with Altera IP

There are three ways to instantiate a DLL with Altera IP:

- The most common way to instantiate a DLL with Altera IP is to use the ALTMEMPHY megafunction. When using the ALTMEMPHY megafunction (Altera's auto-calibrating DDR PHY), the DLL phase offset must be set before calibration. If the DLL phase offset is changed after calibration, you must rerun calibration. To make the required connections in the ALTMEMPHY megafunction in the Quartus® II software version 8.1 and earlier, refer to ["Making the Connections to Use the DLL Phase Offset Feature with the ALTMEMPHY Megafunction"](#) on page 9. The DLL can be either an internal DLL or an external DLL.

- You can instantiate a DLL with custom IP by using the ALTDQS megafunction. Use this method if you are designing a custom DDR PHY interface. When you make the selections in the ALTDQS MegaWizard™ Plug-In Manager, it makes the connections necessary for the DLL phase offset feature automatically. To make the required connections using the ALTDQS megafunction, refer to “Instantiating the DLL Phase Offset Using the ALTDQS Megafunction” on page 22.
- For HardCopy II ASICs, you can also instantiate a DLL with Altera IP by using the Legacy Static PHY. To make the connections between the DLL offset block and the DQS blocks in the Legacy Static PHY, refer to “Making the Connections to Use the DLL Phase Offset Feature with the Legacy Static PHY” on page 23.

The following sections contain step-by-step instructions for each of these methods. These instructions are valid for all Stratix FPGA device families.

Examples of the files that you must change to instantiate a DLL with Altera IP are available in the accompanying zip files. The lines that you must change are marked with the comment `//DLL_Phase_Offset`.

You can find these example files on the Altera web site (www.altera.com), along with this application note:

- `altmemory_int_dll.zip`—The zip file for the ALTMEMPHY megafunction with internal DLL.
- `altmemory_ext_dll.zip`—The zip file for the ALTMEMPHY megafunction with external DLL.
- `static_dll.zip`—The zip file for the Legacy Static PHY.

Making the Connections to Use the DLL Phase Offset Feature with the ALTMEMPHY Megafunction

The following examples describe the connections you must make in your design to implement the DLL phase offset feature with the ALTMEMPHY megafunction for the Quartus II software version 8.1 and earlier. The first example gives instructions for an ALTMEMPHY megafunction with an internal DLL. The second example gives instructions for an external DLL, which you can use when the DLL is shared between ALTMEMPHY megafunctions.

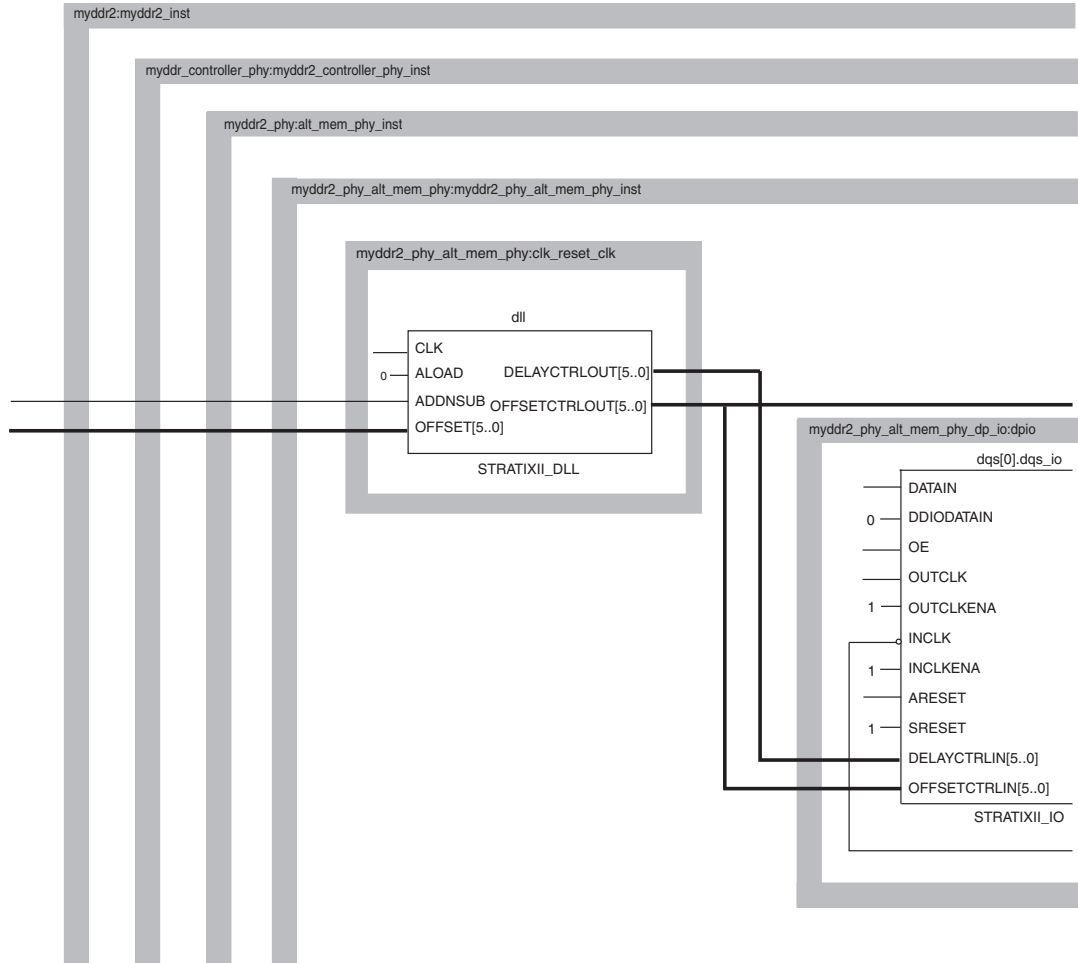
This demonstration design uses:

- The filename `myddr2` in the MegaWizard Plug-In Manager
- The DDR2 SDRAM High Performance Controller version 8.0
- The project name is `myddr2_example_top`
- The top-level module is `myddr2_example_top` and is contained in the `myddr2_example_top.v` file

Internal DLL

Figure 6 shows a schematic block diagram for an ALTMEMPHY megafunction with an internal DLL.

Figure 6. ALTMEMPHY Megafunction with Internal DLL



1. In the `myddr2_example_top.v` file:

- a. In the wire list before the first wire declaration, add these two lines:

```
wire addnsub;
wire [5:0] offset;
```

to make the connections from the DLL to your logic that controls the offset feature.

- b. Connect those wires to pins or registers in your design to control the DLL phase offset feature.

- c. In the `myddr2 myddr2_inst` instantiation, in the port connection list, add these two lines:

```
.addnsub (addnsub) ,  
.offset (offset) ,
```

to create the top-level port connections and wires to these ports.

2. Because the MegaWizard Plug-In Manager controller filename you are using in your example is `myddr2`, the Verilog file that you must modify is `myddr2.v`.

- a. In the port list, add these two lines:

```
addnsub ,  
offset ,
```

to create ports to connect `addnsub` and `offset` to the controller.

- b. In the input declarations, add these two lines:

```
input addnsub ;  
input [5:0] offset ;
```

to declare the ports `addnsub` and `offset`.

- c. In the `myddr2_controller_phy myddr2_controller_phy_inst` instantiation, in the port connection list, add these two lines:

```
.addnsub (addnsub) ,  
.offset (offset) ,
```

to create the top-level port connections and wires to these ports on the controller.

3. In the `myddr2_controller_phy.v` file:

- a. In the port list, add these two lines:

```
addnsub ,  
offset ,
```

to create ports to connect `addnsub` and `offset` to `alt_mem_phy`.

- b. In the input declarations, add these two lines:

```
input addnsub ;  
input [5:0] offset ;
```

to declare the ports `addnsub` and `offset`.

- c. In the `myddr2_phy alt_mem_phy_inst` instantiation, in the port connection list, add these two lines:

```
.addnsub (addnsub) ,  
.offset (offset) ,
```

to create the top-level port connections and wires to these ports on `alt_mem_phy`.

4. In the `myddr2_phy.v` file:
 - a. In the port list, add these two lines:


```
addnsub,
offset,
```

to create ports to connect `addnsub` and `offset`.
 - b. In the input declarations, add these two lines:


```
input addnsub;
input [5:0] offset;
```

to declare the ports `addnsub` and `offset`.
 - c. In the `myddr2_phy_alt_mem_phy_sii` `myddr2_phy_alt_mem_phy_sii_inst` instantiation, in the port connection list, add these two lines:


```
.addnsub(addnsub) ,
.offset(offset) ,
```

to create the top-level port connections and wires to these ports.
5. In the `myddr2_phy_alt_mem_phy_sii.v` file:
 - a. In the port list in the section:


```
//DLL import/export ports :
```

add these three lines:

```
addnsub,
offset,
dqs_offset_ctrl_export,
```

to create ports to connect `addnsub`, `offset`, and `dqs_offset_ctrl_export`.
 - b. In the input and output declarations, add these three lines:


```
input wire addnsub;
input wire [5:0] offset;
output wire [DQS_DELAY_CTL_WIDTH - 1 : 0] \
dqs_offset_ctrl_export;
```

to declare the ports `addnsub`, `offset`, and `dqs_offset_ctrl_export`.
 - c. After the line:


```
wire [DQS_DELAY_CTL_WIDTH - 1 : 0] \
dqs_delay_ctrl_internal;
```

add these two lines:

```
wire [DQS_DELAY_CTL_WIDTH - 1 : 0] \
dqs_offset_ctrl_internal;
```

```
wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] dqs_offset_ctrl;
//Output from clk_reset block
```

to create wires to connect `dqs_offset_ctrl_internal` and `dqs_offset_ctrl`.

d. After the line:

```
assign dqs_delay_ctrl_internal = dqs_delay_ctrl;
```

add this line:

```
assign dqs_offset_ctrl_internal = dqs_offset_ctrl;
```

to connect `dqs_offset_ctrl` to `dqs_offset_ctrl_internal`.

e. After the line:

```
assign dqs_delay_ctrl_export = dqs_delay_ctrl;
```

add this line:

```
assign dqs_offset_ctrl_export = dqs_offset_ctrl;
```

to connect `dqs_offset_ctrl` to `dqs_offset_ctrl_export`.

f. In the `myddr2_phy_alt_mem_phy_dp_io_sii` module instantiation, after the line:

```
.dqs_delay_ctrl (dqs_delay_ctrl_internal),
```

add this line:

```
.dqs_offset_ctrl (dqs_offset_ctrl_internal),
```

to create the port and make the connection between `dqs_offset_ctrl_internal` and `dqs_offset_ctrl`.

g. In the module instantiation:

`myddr2_phy_alt_mem_phy_clk_reset_sii`, after the line:

```
.dqs_delay_ctrl (dqs_delay_ctrl),
```

add these three lines:

```
.dqs_offset_ctrl (dqs_offset_ctrl),
```

```
.addnsub(addnsub),
```

```
.offset(offset),
```

to create the top-level port connections and wires to these ports.

h. In the module declaration:

```
module myddr2_phy_alt_mem_phy_dp_io_sii
```

after the line:

```
dqs_delay_ctrl,
```

add this line:

```
dqs_offset_ctrl,
```

to add the port `dqs_offset_ctrl`.

- i. In the input port declaration list, after the line:

```
input wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] \
dqs_delay_ctrl;
```

add this line:

```
input wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] \
dqs_offset_ctrl;
```

to declare the port `dqs_offset_ctrl`.

- j. In the generate block that instantiates:

```
begin : dqs stratixii_io ... dqs_io
```

being careful not to change the block that has `dq_io`, change the line:

```
.dqs_offsetctrl_enable ("false"),
```

to:

```
.dqs_offsetctrl_enable ("true"),
```

after the line:

```
.delayctrlin (dqs_delay_ctrl),
```

change the line:

```
.offsetctrlin (),
```

to:

```
.offsetctrlin (dqs_offset_ctrl),
```

to make the connection of `offsetctrlin` to the DQS blocks.

- k. In the module declaration:

```
module: myddr2_phy_alt_mem_phy_clk_reset_sii
```

in the port list, add these three lines:

```
dqs_offset_ctrl,
```

```
addnsub,
```

```
offset,
```

to add the ports `dqs_offset_ctrl`, `addnsub`, and `offset`.

- l. In the instantiation:

```
myddr2_phy_alt_mem_phy_dp_io_sii ... ) dpio (
```

after the line:

```
.dqs_delay_ctrl (dqs_delay_ctrl_internal),
```

add this line:

```
.dqs_offset_ctrl (dqs_offset_ctrl_internal),
```

to create the port `dqs_offset_ctrl` and connect `dqs_offset_ctrl_internal` to it.

- m. In the input declarations in the "Misc I/O" section, add these three lines:

```
input addnsub;  
input [5:0] offset;  
output wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] \  
dqs_offset_ctrl;
```

to declare the ports `addnsub`, `offset`, and `dqs_offset_ctrl`.

- n. In the generate block that instances `stratixii_dll`,
change the line:

```
.offsetctrlout_mode ("static"),
```

to:

```
.offsetctrlout_mode ("dynamic_addnsub"),
```

to set the mode of the offset block of the DLL to `addnsub`.

- o. In the same generate block that instances `stratixii_dll`,
change the line:

```
.offsetctrlout (),
```

to:

```
.offsetctrlout (dqs_offset_ctrl),
```

to connect `dqs_offset_ctrl` to the `offsetctrlout` port of the DLL.

- p. In the same generate block that instances `stratixii_dll`,
change the line:

```
.addnsub (),
```

to:

```
.addnsub (addnsub),
```

to connect `addnsub` to the `addnsub` port of the DLL.

- q. In the same generate block that instances `stratixii_dll`,
change the line:

```
.offset (),
```

to:

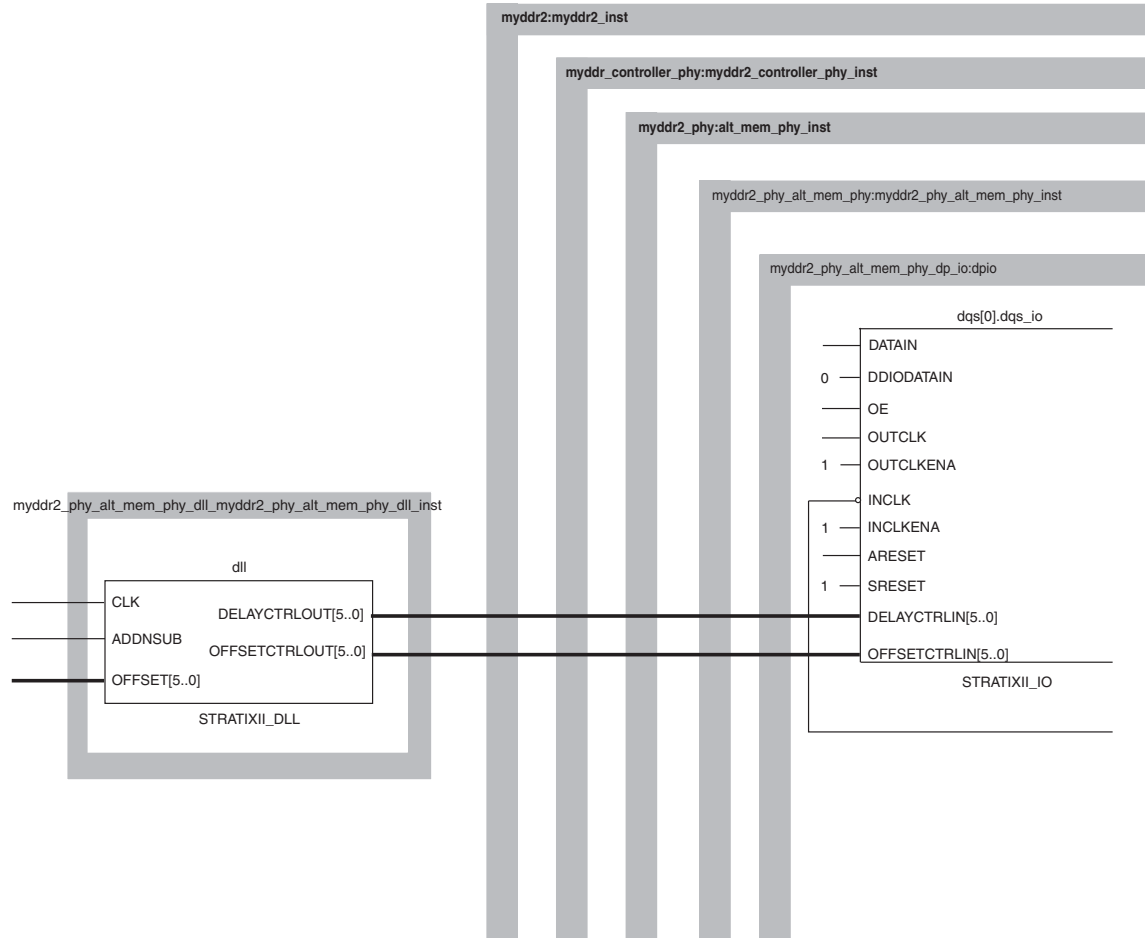
```
.offset (offset),
```

to connect `offset` to the `offset` port of the DLL.

External DLL

Figure 7 shows a schematic block diagram for an ALTMEMPHY megafunction with an external DLL.

Figure 7. ALTMEMPHY Megafunction with External DLL



1. In the file with the level that instantiates the controller (this example uses the `mddr2_example_top.v` file):

- a. In the wire declaration list, after the line:

```
wire [5:0] dqs_delay_ctrl_import_sig;
```

add this line:

```
wire [5:0] dqs_offset_ctrl_import_sig;
```

to create the wire `dqs_offset_ctrl_inport_sig`.

b. In the DLL instantiation:

```
myddr2_phy_alt_mem_phy_dll_sii \  
myddr2_phy_alt_mem_phy_dll_sii_inst
```

after the line:

```
.delayctrlout (dqs_delay_ctrl_import_sig)
```

add these three ports:

```
.offsetctrlout (dqs_offset_ctrl_import_sig),  
.offset (your_offset_signal),  
.addnsub (your_addnsub_signal)
```

to create the top-level port connections and wires to these ports.



For each line you insert, be sure to add a comma to the end of the previous line.

c. Add wire declarations for:

```
wire [5:0] your_offset_signal  
wire your_addnsub_signal
```

to create the wires that connect your control logic to the DLL.



Connect these signals either to pins or registers to control the DLL phase offset feature.

d. In the instantiation:

```
myddr2 myddr2_inst
```

after the line:

```
.dqs_delay_ctrl_import (dqs_delay_ctrl_import_sig),
```

add this line:

```
.dqs_offset_ctrl_import (dqs_offset_ctrl_import_sig),
```

to add the top-level port and make the connection for `dqs_offset_ctrl_import_sig`.

2. In the `myddr2_phy_alt_mem_phy_dll_sii.v` file:

a. In the port list of the module declaration:

```
myddr2_phy_alt_mem_phy_dll_sii
```

add these three ports:

```
offsetctrlout  
offset  
addnsub
```

to create ports to connect `offsetctrlout`, `offset`, and `addnsub` to the DLL.

- b. In the input and output list, add these three lines:

```
output [5:0] offsetctrlout;  
input [5:0] offset;  
input addnsub;
```

to declare the ports `offsetctrlout`, `offset`, and `addnsub`.

- c. In the wire list, add these three lines:

```
wire [5:0] offsetctrlout;  
wire [5:0] offset;  
wire      addnsub;
```

to create wires to connect `offsetctrlout` and `offset`.

- d. In the `stratixii_dll` instantiation, change these three lines:

change:

```
.addnsub ( ),
```

to:

```
.addnsub(addnsub) ,
```

change:

```
.offset ( ),
```

to:

```
.offset(offset) ,
```

and change:

```
.offsetctrlout ( ),
```

to:

```
.offsetctrlout(offsetctrlout) ,
```

to connect `addnsub`, `offset`, and `offsetctrlout` to the DLL.

3. In the filename that you used for your controller (this example uses `myddr2`), the Verilog file is `myddr2.v`:

- a. In the port list, after:

```
dqs_delay_ctrl_import ,
```

add this line:

```
dqs_offset_ctrl_import ,
```

to create a port to connect `dqs_offset_ctrl_import`.

- b. In the input declarations, after:

```
input [5:0] dqs_delay_ctrl_import;
```

add this line:

```
input [5:0] dqs_offset_ctrl_import;
```

to declare the port `dqs_offset_ctrl_import`.

- c. In the instantiation:

```
myddr2_controller_phy myddr2_controller_phy_inst(
```

after the line:

```
.dqs_delay_ctrl_import(dqs_delay_ctrl_import),
```

add this line:

```
.dqs_offset_ctrl_import(dqs_offset_ctrl_import),
```

to create the top-level port and connect the wire to `dqs_offset_ctrl_import`.

4. In the `myddr2_controller_phy.v` file:

- a. In the port list, after:

```
dqs_delay_ctrl_import,
```

add this line:

```
dqs_offset_ctrl_import,
```

to create port `dqs_offset_ctrl_import`.

- b. In the input declarations, after:

```
input [5:0] dqs_delay_ctrl_import;
```

add this line:

```
input [5:0] dqs_offset_ctrl_import;
```

to declare port `dqs_offset_ctrl_import`.

- c. In the instantiation:

```
mddr2_phy alt_mem_phy_inst
```

after the line:

```
.dqs_delay_ctrl_import(dqs_delay_ctrl_import),
```

add this line:

```
.dqs_offset_ctrl_import(dqs_offset_ctrl_import),
```

to create the port and make the connection to `dqs_offset_ctrl_import`.

5. In the `myddr2_phy.v` file:

a. In the port list, after:

```
dqs_delay_ctrl_import,
```

add this line:

```
dqs_offset_ctrl_import,
```

to create the port `dqs_offset_ctrl_import`.

b. In the input declarations, after:

```
input [5:0] dqs_delay_ctrl_import;
```

add this line:

```
input [5:0] dqs_offset_ctrl_import;
```

to declare the port `dqs_offset_ctrl_import`.

c. In the instantiation:

```
mddr2_phy_alt_mem_phy_sii
myddr2_phy_alt_mem_phy_sii_inst
```

after the line:

```
.dqs_delay_ctrl_import (dqs_delay_ctrl_import),
```

add this line:

```
.dqs_offset_ctrl_import (dqs_offset_ctrl_import),
```

to add the port and make the connection to `dqs_offset_ctrl_import`.

6. In the `myddr2_phy_alt_mem_phy_sii.v` file:

a. In the port list of the module:

```
altmemphy_phy_alt_mem_phy_sii
```

after the line:

```
dqs_delay_ctrl_import,
```

add this line:

```
dqs_offset_ctrl_import,
```

to create the port `dqs_offset_ctrl_import`.

b. In the port declaration list, after the line:

```
input wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] \
dqs_delay_ctrl_import;
```

add this line:

```
input wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] \
dqs_offset_ctrl_import;
```

to declare the port `dqs_offset_ctrl_import`.

c. After the wire declaration:

```
wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] \  
dqs_delay_ctrl_internal;
```

add this line:

```
wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] \  
dqs_offset_ctrl_internal;
```

to declare the wire `dqs_offset_ctrl_internal`.

d. Before the line:

```
if (DLL_EXPORT_IMPORT == "IMPORT")
```

add this line:

```
assign dqs_offset_ctrl_internal = \  
dqs_offset_ctrl_import;
```

to connect `dqs_offset_ctrl_import` to `dqs_offset_ctrl_internal`.

e. In the instantiation:

```
myddr2_phy_alt_mem_phy_dp_io_sii ... dpio(  
after the line:
```

```
.dqs_delay_ctrl (dqs_delay_ctrl_internal),
```

add this line:

```
.dqs_offset_ctrl (dqs_offset_ctrl_internal),
```

to create the port and make the connection to `dqs_offset_ctrl_internal`.

f. In the module declaration:

```
module myddr2_phy_alt_mem_phy_dp_io_sii
```

in the port list, after the line:

```
dqs_delay_ctrl,
```

add this line:

```
dqs_offset_ctrl,
```

to create the port `dqs_offset_ctrl`.

g. In the input declaration list, after the line:

```
input wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] \  
dqs_delay_ctrl;
```

add this line:

```
input wire [DQS_DELAY_CTL_WIDTH - 1 : 0 ] \  
dqs_offset_ctrl;
```

to declare port `dqs_offset_ctrl`.

h. After the line:

```
begin : dqs
```

in the generate block that instantiates `dqs_stratixii_io`, being careful not to change the block that has `dq_io`, change the line:

```
.dqs_offsetctrl_enable ("false")
```

to:

```
.dqs_offsetctrl_enable ("true")
```

to enable the DLL phase offset feature in the DQS blocks.

i. In the `dqs_io` port connection list, change the line:

```
.offsetctrlin (),
```

to:

```
.offsetctrlin (dqs_offset_ctrl),
```

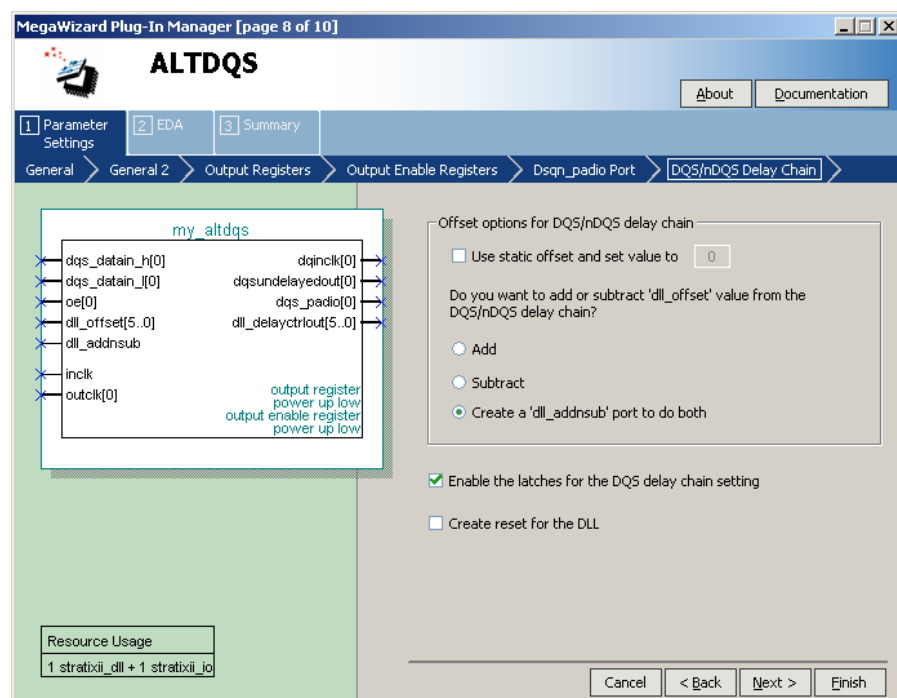
to connect `dqs_offset_ctrl` to the `offsetctrlin` port on the DQS block.

Instantiating the DLL Phase Offset Using the ALTDQS Megafunction

This section describes how you can instantiate a DLL with Altera IP by using the ALTDQS megafunction. You can use this method if you are designing a custom DDR PHY interface. When you make the selections in the ALTDQS MegaWizard Plug-In Manager, it makes the connections necessary for the DLL phase offset feature automatically.

Figure 8 shows page 8 of the ALTDQS MegaWizard Plug-In Manager.

Figure 8. ALTDQS MegaWizard Plug-In Manager (Page 8)



On page 8 of the ALTDQS MegaWizard Plug-In Manager, deselect **Use static offset...** and select **Create a 'dll_addnsub' port to do both**. This creates the `addnsub` and `offset [5:0]` ports. These ports must be brought up to a register (initialized to zero) that is controlled from firmware in your end system.

Making the Connections to Use the DLL Phase Offset Feature with the Legacy Static PHY

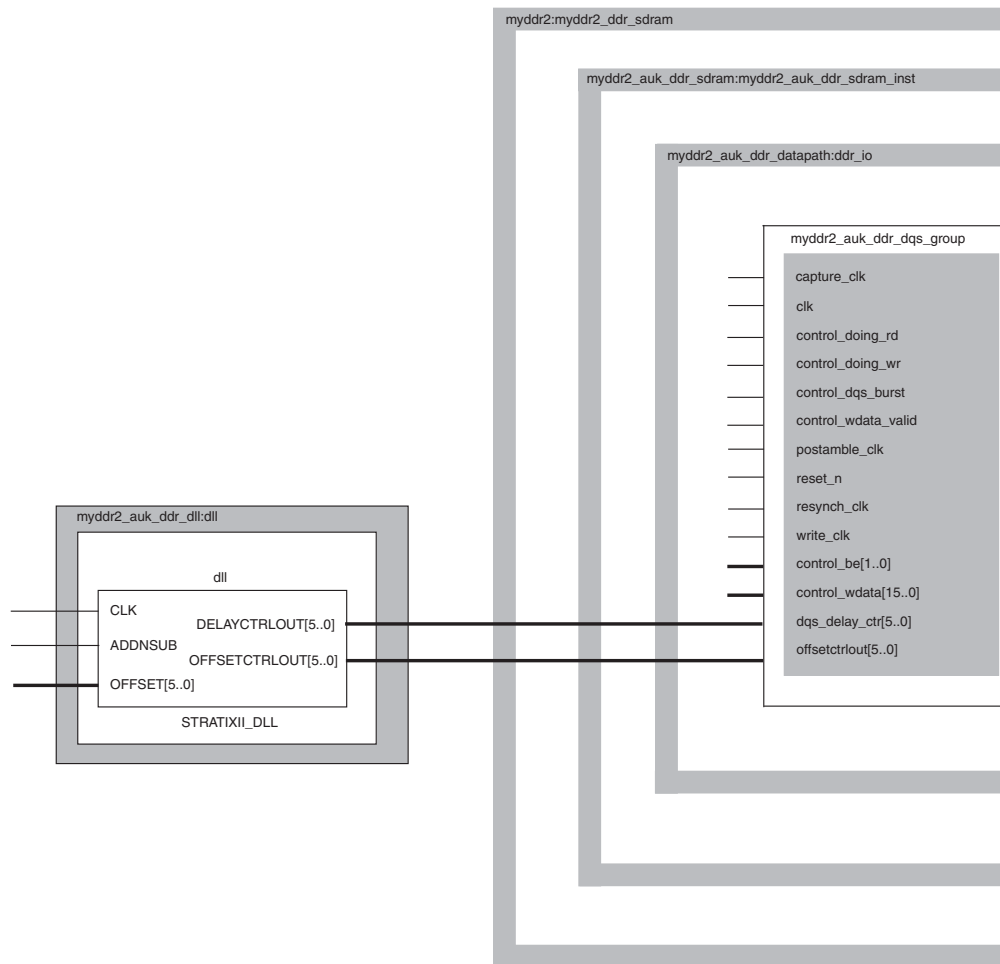
To use the DLL phase offset feature with the Legacy Static PHY, you must connect the `offsetctrlout [5:0]` output from the DLL to the DQS blocks. The following is a demonstration design to show how to make the connections. In the MegaWizard Plug-In Manager for the Legacy Static PHY, on the **controller** tab at the bottom, turn on the **Insert logic to allow the DLL to update only during the memory refresh period** option (when turned on, it inserts a register in the `delayctrlout` path that causes it to be out-of-sync with `offsetctrlout`).

The demonstration design uses:

- The filename `myddr2` in the MegaWizard Plug-In Manager for the name of the Legacy Static PHY.
- The project name is `myddr2_example_top`.
- The top-level module is `myddr2_example_top` and is contained in the file `top.v`.

Figure 9 shows a schematic block diagram for a Legacy Static PHY.

Figure 9. Legacy Static PHY



To make the connections for using the DLL phase offset feature with the Legacy Static PHY, perform the following steps:

1. In the top-level file (this example uses `myddr2_example_top.v`), locate the DLL instance. In this example, the DLL instance contains the following 10 lines:

```
myddr2_auk_ddr_dll dll
(
  .addnsub (1'b0),
  .clk (dqs_ref_clk),
  .delayctrlout (dqs_delay_ctrl),
  .dqsupdate (dqsupdate),
  .offset (6'b000000),
  .reset_n (soft_reset_reg2_n),
  .stratix_dll_control (stratix_dll_control)
);
```

Modify the DLL instance to replace the hard coded values `1'b0` and `6'b000000` with the signals `addnsub` and `offset`, respectively, and add the output port `offsetctrlout`.

Your new DLL instance has 11 lines and looks like the following:

```
myddr2_auk_ddr_dll dll
(
  .addnsub (addnsub), //Changed 1'b0 to addnsub
  .clk (dqs_ref_clk),
  .delayctrlout (dqs_delay_ctrl),
  .offsetctrlout (offsetctrlout), //Added the offsetctrlout port
  .dqsupdate (dqsupdate),
  .offset (offset), //Changed 6'b000000 to offset
  .reset_n (soft_reset_reg2_n),
  .stratix_dll_control (stratix_dll_control)
);
```

2. In the `myddr2_example_top.v` file, after:

```
wire [5:0] dqs_delay_ctrl;
```

add this line:

```
wire [5:0] offsetctrlout; //Added offsetctrlout wire.
```

to declare the wire `offsetctrlout`.

3. In the `myddr2_example_top.v` file in the instantiation:

```
myddr2 myddr2_ddr_sdram
```

after the line:

```
.dqs_delay_ctrl (dqs_delay_ctrl),
```

add this line:

```
.offsetctrlout (offsetctrlout), //Added offsetctrlout port
```

to add the port and make the top-level connection for `offsetctrlout`.

4. In the `myddr2_auk_ddr_dll.v` file, add the port `offsetctrlout` to the module:

```
myddr2_auk_ddr_dll
```

Also add a wire `[5:0] offsetctrlout` and connect it to the port `offsetctrlout` on the `stratixii_dll` instantiation.

The Verilog file should look like the following:

```
module myddr2_auk_ddr_dll (
  //inputs:
  addnsub,
  clk,
  offset,
  reset_n,
  stratix_dll_control,

  //outputs:
  delayctrlout,
  offsetctrlout, //Add offsetctrlout port
  dqsupdate
)

/* synthesis ALTERA_ATTRIBUTE = \
"MESSAGE_DISABLE=14130;MESSAGE_DISABLE=14110" */ ;
```

```

output [5:0] delayctrlout;
output [5:0] offsetctrlout; //Add offsetctrlout port
output      dqsupdate;
input      addnsub;
input      clk;
input [5:0] offset;
input      reset_n;
input      stratix_dll_control;
wire [5:0] delayctrlout;
wire [5:0] offsetctrlout; //Add offsetctrlout wire
wire      dqsupdate;

//-----
// Instantiate Stratix II DLL
//-----

stratixii_dll dll
(
    .addnsub (addnsub),
    .aload (),
    .clk (clk),
    .delayctrlout (delayctrlout),
    .devclrn (),
    .devpor (),
    .dqsupdate (dqsupdate),
    .offset (offset),
    .offsetctrlout (offsetctrlout), //Add the offsetctrlout
    // connection
    .upndnin (),
    .upndninclkena (),
    .upndnout ()
);

defparam dll.delay_buffer_mode = "low",
          dll.delay_chain_length = 12,
          dll.delayctrlout_mode = "normal",
          dll.input_frequency = "7500ps",
          dll.jitter_reduction = "false",
          dll.lpm_type = "stratixii_dll",
          dll.offsetctrlout_mode = "dynamic_addnsub",
          //Make sure this parameter is set to dynamic_addnsub
          dll.sim_loop_delay_increment = 144,
          dll.sim_loop_intrinsic_delay = 3600,
          dll.sim_valid_lock = 1,
          dll.sim_valid_lockcount = 27,
          dll.static_delay_ctrl = 0,
          dll.static_offset = "0",
          dll.use_upndnin = "false",
          dll.use_upndninclkena = "false";

```

The MegaWizard Plug-In Manager sets the above parameters appropriately for the frequency of your design. The only parameter you typically must change is `dll.offsetctrlout_mode` to `dynamic_addnsub`.

Route the `offset` and `addnsub` signals up through your design hierarchy to either register configuration pins or directly to jumper-controlled I/Os. Ensure that your system initializes these pins to 0. If you are using I/O control, ensure each of the seven pins (six for `offset` and one for `addnsub`) can be set either to 1 or 0.

1. In the `myddr2_auk_ddr_sdram.v` file:

a. In the module declaration:

```
module myddr2_auk_ddr_sdram
```

after the port: `dqs_delay_ctrl`,

add the port:

```
offsetctrlout,
```

in the input declaration section, add:

```
input [5:0] offsetctrlout; //Add offsetctrlout port
```

b. In the module instantiation:

```
myddr2_auk_ddr_datapath ddr_io
```

after the port connection: `.dqs_delay_ctrl (dqs_delay_ctrl)`,

add this line:

```
.offsetctrlout(offsetctrlout), //Add offsetctrlout port
```

to add the port and make the connection for `offsetctrlout`.

2. In the `myddr2.v` file:

a. In the module declaration:

```
module myddr2
```

after the port: `dqs_delay_ctrl`,

add the port:

```
offsetctrlout,
```

in the input declaration section, add:

```
input [5:0] offsetctrlout; //Add offsetctrlout port
```

b. In the module instantiation:

```
myddr2_auk_ddr_sdram myddr2_auk_ddr_sdram_inst
```

after the port connection: `.dqs_delay_ctrl (dqs_delay_ctrl)`,

add this line:

```
.offsetctrlout(offsetctrlout), //Add offsetctrlout port
```

to add the port and make the connection for `offsetctrlout`.

3. In the `myddr2_auk_ddr_datapath.v` file:

a. In the module declaration:

```
module myddr2_auk_ddr_datapath
```

after the port: `dqs_delay_ctrl`,

add the port:

```
offsetctrlout,
```

in the input declaration section, add:

```
input [5:0] offsetctrlout; //Added offsetctrlout port
```

b. In the module instantiation:

```
myddr2_auk_dds_dqs_group \g_datapath:N:g_dds_io
```

where the “N” goes from 0 to the width of the DDR DQS bus minus 1, after the port connection: `.dqs_delay_ctrl (dqs_delay_ctrl),`

add this line:

```
.offsetctrlout(offsetctrlout), //Add offsetctrlout port
```

4. In the `myddr2_auk_dds_dqs_group.v` file:

a. In the module declaration:

```
module myddr2_auk_dds_dqs_group
```

after the port: `dqs_delay_ctrl,`

add the port:

```
offsetctrlout,
```

in the input declaration section, add:

```
input [5:0] offsetctrlout; //Added offsetctrlout port
```

b. In the module instantiation:

```
stratixii_io dqs_io
```

after the port: `.delayctrlin(dqs_delay_ctrl),`

add the port:

```
.offsetctrlin(offsetctrlout),
```

Do this in both the `SIMULATION-ONLY CONTENTS` and the `//synthesis translate_on` sections.

In both the “simulation” and “synthesis defparam” sections of the instantiation, change the parameter:

```
dqs_io.dqs_offsetstrl_enable = "false"
```

to:

```
dqs_io.dqs_offsetstrl_enable = "true"
```



Take care to change ONLY DQS instantiations; do NOT change the DQI/O instantiations `g_dq_io:?:dq_io .offsetctrl_enable`, as these are for the DQ pins, not the DQS pins.

Confirm Your DLL Phase Offset Control Settings

Regardless of which method you use, check that you have implemented the RTL changes correctly. Perform these steps to check your design:

1. Compile your design.
2. Open the resource section of the Fitter report.
 - a. In the DLL Summary section, the column labeled **Offset Control Out Mode** must read `dynamic_addnsb`, as shown in Figure 10.
 - b. In the DQS Summary section, in the column labeled **Offset Control**, all rows must read **On**, as shown in Figure 11.

If these columns read `dynamic_addnsb` and **On** respectively, you have made the DLL phase control offset settings correctly.

Figure 10 shows the **Offset Control Out Mode** column reading `dynamic_addnsb`.

Figure 10. DLL Summary Reads `dynamic_addnsb` in the Offset Control Out Mode Column

Location	Input Frequency	Low Jitter/Fast Lock	Cycles Required to Lock	Delay Control Out Mode	Delay Buffers in Loop	Delay Buffer Mode	Offset Control Out Mode	Static Offset
1 DLL_X59_V0_N0	133.33 MHz	Fast Lock	256	normal	12	low	dynamic_addnsb	N/A

Figure 11 shows all rows in the **Offset Control** column reading **On**.

Figure 11. DQS Summary Reads **On** in the Offset Control Column

Location	DQS Bus	Input Frequency	Associated DLL	DLL-Controlled Phase Shift	Delay Buffers	Delay Buffer Mode	Offset Control	Control Latches	Edge Detect	Gated DQS
1	DQS-9 I/O bus	133.33 MHz	myddr2_auk_ddr_dll:dlljdl	60.00 degrees	2	low	On	Off	Off	On
2	DQS-9 I/O bus	133.33 MHz	myddr2_auk_ddr_dll:dlljdl	60.00 degrees	2	low	On	Off	Off	On
3	DQS-9 I/O bus	133.33 MHz	myddr2_auk_ddr_dll:dlljdl	60.00 degrees	2	low	On	Off	Off	On
4	DQS-9 I/O bus	133.33 MHz	myddr2_auk_ddr_dll:dlljdl	60.00 degrees	2	low	On	Off	Off	On

Document Revision History

Table 3 shows the revision history for this application note.

Table 3. Document Revision History

Date	Version	Changes Made
March 2010	2.0	<ul style="list-style-type: none"> ■ Changed “HardCopy II” and “HardCopy devices” to “HardCopy ASICs” when referring to the HardCopy device family. ■ Changed “Stratix II” and “Stratix devices” to “Stratix FPGAs” when referring to the Stratix device family. ■ Updated Table 1 and Table 2. ■ Updated Figure 4 and Figure 5 with color. ■ Minor text edits.
November 2008	1.0	Initial Release.



101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support

Copyright © 2010 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



I.S. EN ISO 9001