Stratix II DLL WYSISYG Description

Version 1.71
May 27, 2005

by
Altera Corporation

Author:	Altera Corporation
System:	
Subsystem:	
Keywords:	DLL, DQS, DDR

Table of Contents:

1 Overview

This describes the Stratix II DLL (Delay Locked Loop) WYSIWYG and relevant input ports on the Stratix II I/O WYSIWYG. These components can be used to correctly register data inputs when reading from DDR-I/II SDRAM, DDR-I/II FCRAM, QDR-II SDRAM, or RLDRAM-II.

Typically when reading from an external RAM, the external RAM sends a strobe or clock, referred to here as DQS, edge-aligned with the data input, referred to here as DQ. To register the data at the optimum time using the strobe, the strobe must be phase shifted to center-align with the data (usually a shift of about 90°). The DLL circuit uses a reference clock at the same frequency as the input DQS strobes to compute the amount of delay required at each DQS pin to perform the phase shift. The DLL can then feed the required setting to the delay chains of DQS I/Os.

2 Delay Locked Loop Representation

The DLL (delay locked loop) represents the specialized phase shift reference circuit for determining the delay in the DQS read strobes required to register the DQ inputs. Currently, each device in the family has two DLLs – one on the top edge and one on the bottom edge. Note that the WYSIWYG should be backwards compatible with the Stratix DLL WYSIWYG.

2.1 DLL Primitive

stratixii_dll <*dll_name*>
(
.clk(<*input port*>**),**
.aload(<*input port*>**),**
.offset[5..0](<*input bus*>**),**
.upndnin(<*input port*>**),**
.upndninclkena(<*input port*>**),**
.addnsub(<*input port*>**),**

.delayctrlout[5..0](<*output bus*>**),**
.offsetctrlout[5..0](<*output bus*>**),**
.dqsupdate(<*output port*>**)**
.upndnout(<*output port*>**),**
);

```
defparam <dll_name>.input_frequency = <input_frequency of clk>;
defparam <dll_name>.delay_buffer_mode = <delay_buffer_mode>;
defparam <dll_name>.delay_chain_length = <delay_chain_length>;
defparam <dll_name>.delayctrlout_mode = <delayctrlout_mode>;
defparam <dll_name>.offsetctrlout_mode = <offsetctrlout_mode>;
defparam <dll_name>.static_offset = <static_offset>;
defparam <dll_name>.jitter_reduction = <jitter_reduction>;
defparam <dll_name>.use_upndnin = <use_upndnin>;
defparam <dll_name>.use_upndninclkena = <use_upndninclkena>;
```

```
Simulation parameter:
defparam <dll_name>.sim_valid_lock = <sim_valid_lock>;

Hidden Simulation parameters:
defparam <dll_name>.sim_valid_lockcount = <sim_valid_lockcount>;
defparam <dll_name>.sim_loop_intrinsic_delay =
      <sim_loop_intrinsic_delay>;
defparam <dll_name>.sim_loop_delay_increment =
      <sim_loop_delay_increment>;


Test parameter:
defparam <dll_name>.static_delay_ctrl = <static_delay_ctrl>;
```

2.2 DLL Input Signals

<*dll_name*>**:** is the unique identifier for the delay locked loop. This is any identifier name which is legal for the given description language (e.g. Verilog, VHDL, AHDL, etc.). *This field is required.*

.clk(<*input port*>**)** is the reference clock matching in frequency to the DQS clock used to determine the delay required to perform the phase shift. This input is required.

.aload(<*input port*>**)** is the asynchronous load signal for the DLL's up/down counter. When aload is high, the register is asynchronously loaded with the initial_delay_setting. This input is optional and defaults to GND .

.offset[5..0](<*input bus*>**)** is the offset added or subtracted (depending on the setting of the *offsetctrlout_mode* parameter and/or the state of the **addnsub** input) from the DLL's **delayctrlout** output to get the **offsetctrlout** result. This input is ignored if *offsetctrlout_mode* is set to *static* and the *delayctrlout_mode* isn't set to *normal_offset* or *offset_only*. This input is optional and defaults to GND. This input can be fed by the core and is a 2's complement number. The resulting offset is limited to the min and max possible values for the **delayctrlout[5..0]**, so any overflow will result in a **offsetctrlout** output of 0x3F; similarly, any underflow will result in an **offsetctrlout** output of 0x00. Here's an example of how to use this input in *dynamic_addnsub* mode:

addnsub input	offset[5..0] input	offsetctrlout[5..0] output
1	0x02	**delayctrlout[5..0]** + 2
1	0x01	**delayctrlout[5..0]** + 1
1	0x00	**delayctrlout[5..0]**
0	0x3F	**delayctrlout[5..0]** - 1
0	0x3E	**delayctrlout[5..0]** - 2

.upndnin(<*input port*>**)** is the up/down input port for the DLL's up/down counter. This input is required if *use_upndnin* is set to *true*. This input is optional and defaults to GND. This input can only be fed by the core.

.upndninclkena(<*input port*>**)** is the clock enable input port for the DLL's up/down counter. This input is required if *use_upndninclkena* is set to *true*. This input is optional and defaults to VCC. This input can only be fed by the core.

.addnsub(<*input port***>)** is the add/sub input port to control whether the delay offset setting is added or subtracted. If set to VCC, the offset is added; if set to GND, the offset is subtracted. This input is required if *offsetctrlout_mode* is set to *dynamic_addnsub*, otherwise this input is optional and defaults to VCC. This input can only be fed by the core.

2.3 DLL Output Signals

.delayctrlout[5..0](<*output bus*>**)** is typically the value of the DLL's current delay chain setting (targeted toward delaying the input clock by the amount specified in the phase_shift parameter). The value can be adjusted with an offset depending on the value of the *delayctrlout_mode* parameter (see Section 2.4 below).

.offsetctrlout[5..0](<*output bus*>**)** is typically the value of the DLL's current delay offset setting (which is **delayctrlout** +/- offset). The output can be adjusted depending on the value of the *offsetctrlout_mode* parameter (see Section 2.4 below). This signal can only feed dedicated routing to the corresponding offsetctrlin ports of DQS pins. To use this signal, the DQS pins must have their *dqs_offsetctrl_enable* parameter set to *true*.

.dqsupdate(<*output port*>**)** is the update enable signal for the delay setting latches in the DQS pins. This signal can only feed DQS pins and only the **dqsupdateen** input port of those pins. The DQS pins must have their *dqs_ctrl_latches_enable* parameter set to *true*.

.upndnout(<*output port***>)** is the raw output of the phase comparator (before any jitter reduction). This signal feeds the core.

2.4 DLL Modes

<*input_frequency*> is the frequency of the clock connected to the clk input port. This parameter should be checked to ensure it falls within a legal range. If the frequency of the parameter is 100MHz to less than 167MHz, the DLL operates in low-frequency mode; if the frequency is in the range 167MHz to less than 250MHz, the DLL operates in medium-frequency mode; if the frequency is in the range of 250MHz to less than 267MHz, the DLL operates in high-frequency mode; if the frequency is in the range of 267MHz to 300MHz, the DLL operates in very high-frequency mode . This value should be legality checked to make sure it falls in a legal range for the chosen device and speed grade. *This field is required.*

<*delay_chain_length*> is one of *{8, 10, 12, 16}*. Determines the number of variable delay buffers in the delay loop. This is a required field. It is recommended that you set the length to *12* under low- or high-frequency operation, *16* under medium-frequency operation, or *10* for very high-frequency operation.

<*delay_buffer_mode*> is one of *{low, high}*. Determines whether the variable delay buffers are working in low-frequency mode (< 167MHz) or high-frequency mode. This field is optional and defaults to *low*.

<*delayctrlout_mode*> is one of *{normal, normal_offset, offset_only, static, test}*. Determines the output of the **delayctrlout** output bus. *static* and *test* are modes for TEST only. If set to *normal*, the DLL feedback loop counter determines the **delayctrlout** output**.** If set to *normal_offset*, the value of the **offsetctrlout** output bus is added to the value of DLL feedback loop counter to get the **delayctrlout** output. If *delayctrlout*_mode is set to *offset_only* and *offsetctrlout_mode* is set to *dynamic_add* or *dynamic_sub*, then the **offset**

input directly feeds the **delayctrlout** output. If *delayctrlout_*mode is set to *offset_only* and *offsetctrlout_mode* is set to *static*, then the value of the *static_offset* parameter is output. If set to *static* (a TEST mode), the value of the *static_delay_ctrl* parameter is output. If set to *test* (another TEST mode), a combination of values from the aload input and clk input is output. This field is optional and defaults to *normal*.

<*static_delay_ctrl*> is an integer with a range from *0* to *63*. This parameter is only useful for TEST. If *delayctrlout_mode* is set to *static*, the value is output on the delayctrlout output bus. The value should be checked to make sure it's an integer within the legal range. This field is optional and defaults to *0*. Note that if *delayctrlout_mode* is not set to *static*, this value is ignored.

<*offsetctrlout_mode*> is one of *{ dynamic_addnsub, static}*. It determines the output of the **offsetctrlout** output bus. If set to *dynamic_addnsubsub*, then depending on whether the **addnsub** input is asserted or not, the phase offset specified on the **offset** input bus is added or subtracted from the DLL feedback counter output to get the **offsetctrlout** output. If set to *static*, the phase offset specified by the *static_offset* parameter is added to DLL feedback counter output to get the **offsetctrlout** output. This field is optional and defaults to *static*.

<*static_offset*> is an integer with a range from *-63* to *63*. If *offsetctrlout_mode* is set to *static*, the value is added to the DLL feedback counter value and output on the offsetctrlout output bus. The value should be checked to make sure it's an integer within the legal range. This field is optional and defaults to *0*. Note that if *offsetctrlout_mode* is not set to *static*, this value is ignored.

<*jitter_reduction*> is one of {true, false}. Determines whether or not the jitter reduction circuit is enabled on the delayctrlout and offsetctrlout outputs. This field is optional and defaults to *false*.

<*sim_valid_lock*> is the number of half-cycles needed from the clk input before the DLL locks onto the signal. This parameter should be a multiple of 16, if not, it will be rounded down. This parameter is included so that the user doesn't have to add simulation vectors for the actual number of half-cycles needed for a lock (512). This field is optional and defaults to *16*. This parameter is only used for simulation.

<*sim_valid_lockcount*> is the value loaded into the DLL feedback counter when the number of cycles dictated by *sim_valid_lock* has passed. This parameter should only use by the netlist writers to pass the delay loop parameter to the simulation model. This field is optional and defaults to 0. This parameter is only used for simulation.

<*sim_loop_intrinsic_delay*> is the intrinsic delay of the DLL loop in ps. This parameter should only use by the netlist writers to pass the delay loop parameter to the simulation model. This field is optional and defaults to 0. This parameter is only used for simulation.

<*sim_loop_delay_increment*> is the delay increment of the DLL loop in ps. This parameter should only use by the netlist writers to pass the delay loop parameter to the simulation model. This field is optional and defaults to 0. This parameter is only used for simulation.

<*use_upndnin*> is one of *{true, false}*. This field is optional and defaults to *false*. Determines if the **upndnin** port is used to update the DLL counter. If the set to *true*, the DLL will not automatically update the counter, otherwise, the DLL will. Thus if set to *true*, *jitter_reduction* must be set to *false*.

<*use_upndninclkena*> is one of *{true, false}*. This field is optional and defaults to *false*. Determines if the **upndninclkena** port is used or not. Determines if the **upndninclkena** port is used to clock enable the DLL counter. If the set to *true*, the DLL will not automatically set the clock enable for the counter, otherwise, the DLL will.

3 DQS I/O Representation

The delay control input port is user-controllable bus routed from the core. The dqs_bus_out port feeds the DQS clock network and the comb_out/dqs_core_out ports feeds normal routing. Note that the WYSIWYG should be backwards compatible with the Stratix I/O WYSIWYG.

3.1 Updated I/O Primitive

```
stratixii_io <I/O name>
(
        {normal ports},
        .ddioinclk(<input port>),
        .delayctrlin[5..0](<input bus>),
        .offsetctrlin[5..0](<input bus>),
        .dqsupdateen(<input bus>),

        .dqsbusout(<output port>)
);
{same original parameters}…
defparam <I/O name>.dqs_input_frequency = <DQS input frequency>;
defparam <I/O name>.dqs_out_mode = <DQS out mode>;
defparam <I/O name>.dqs_delay_buffer_mode = <DQS delay buffer
   mode>;
defparam <I/O name>.dqs_phase_shift = <phase offset of the delayed
   DQS signal>;
defparam <I/O name>.inclk_input = <inclk input>;
defparam <I/O name>.ddioinclk_input = <ddioinclk input>;
defparam <I/O name>.dqs_offsetctrl_enable = <DQS offsetctrl
   enable>;
defparam <I/O name>.dqs_ctrl_latches_enable = <DQS ctrl latches
   enable>;
defparam <I/O name>.dqs_edge_detect_enable = <DQS edge detect
   enable>;
defparam <I/O name>.gated_dqs = <gated DQS>;

Hidden Simulation parameters:
defparam <I/O name>.sim_dqs_intrinsic_delay =
      <sim_dqs_intrinsic_delay>;
defparam < I/O name>.sim_dqs_delay_increment =
      <sim_dqs_delay_increment>;
defparam < I/O name>.sim_dqs_offset_increment =
      <sim_dqs_delay_increment>;
```

3.2 New I/O Input Signal

.ddioinclk(<*input port*>**)** is a clock input for the negative-edge DDIO input register. Note that this clock is active high. This port is optional and defaults to GND. This port is only valid if *ddioinclk_input* is set to *dqsb_bus*.

.delayctrlin[5..0](<*input bus*>**)** is the delay chain setting for the DQS read path. This port is optional and defaults to GND. This port is only valid if the *dqs_out_mode* is set to *delay_chain*. If the ctrl latches are enabled (*dqs_ctrl_latches_enable* set to *true*), then this bus can only be fed from the **delayctrlout** output of a DLL. If *dqs_ctrl_latches_enable* is set to *false*, then this bus can be fed by the core.

.offsetctrlin[5..0](<*input bus*>**)** is the fine-tune delay chain setting for the DQS output path. This port is optional and defaults to GND. This bus can only be fed from the **offsetctrlout** output of a DLL.

.dqsupdateen(<*input port*>**)** is the enable signal for the **delayctrlin** and **offsetctrlin** latches. This port is only valid if the *dqs_ctrl_latches_enable* parameter is set to *true*. This port can only be fed by the **dqsupdate** output of a DLL. In addition, the DLL's delayctrlout_mode must be set to *normal* or *normal_offset.*

3.3 New I/O Output Signal

.dqsbusout(<**DQS** *bus output*>**)** is the delayed DQS signal from DQS pin that drives onto the dedicated DQS clock network to DQ pins. It can also drive out to the core by stealing outputs from unused DQS pins in the DQS group. This port is optional. It is only valid if the dqs_out_mode is set to *bypass* or *delay_chain*.

3.4 New I/O Modes

Note that all DQS-related parameters are optional since they're not applicable for normal I/Os.

<*dqs input_frequency*> is either set to the frequency of the DQS strobe/clock input or set to *unused*. This parameter should be checked to ensure it falls within a legal range. It should not be specified if *dqs_out_mode* is set to *none*. This field is optional and defaults to *unused*.

<*DQS out mode*> is one of *{none, bypass, delay_chain1, delay_chain2, delay_chain3, delay_chain4}*. This sets the number of DQS delay buffers that should feed the **dqsbusout** port. When it is set to *none*, the **dqsbusout** port should not be connected. When it is set to *bypass*, the **dqsbusout** signal will bypass the DQS delay chain. When it is set to *delay_chain1, delay_chain2, delay_chain3*, or *delay_chain4*, the **dqsbusout** signal will go through 1, 2, 3, or 4 delay buffers, respectively, that are controlled by **delayctrlin[5..0]**. This field is optional and defaults to *none.* Here is a table with the settings needed to achieve particular phase shifts:

DLL *delay_chain_length*	*dqs_phase_shift* (hundreds of degrees)	*dqs_out_mode*
16	2250	delay_chain1
16	4500	delay_chain2
16	6750	delay_chain3
16	9000	delay_chain4
12	3000	delay_chain1
12	6000	delay_chain2
12	9000	delay_chain3
12	12000	delay_chain4
10	3600	delay_chain1
10	7200	delay_chain2
10	10800	delay_chain3

10	14400	delay_chain4

<*DQS delay buffer mode*> is one of *{low, high}*. Determines whether the variable delay buffers are working in low-frequency mode or high-frequency mode. This field is optional and defaults to *low*.

<*phase offset of the delayed DQS signal*> is a number from *{0..36000}*. This parameter indicates the phase shift between the delayed DQS signal and the input DQS signal in units of hundreds of degrees (so a 90 degree phase shift would be represented as 9000). This parameter is mostly used for static timing analysis only since timing analysis cannot figure out the phase shift through the delayctrlin[5:0] and offsetctrlin[5:0] ports like simulation can. This field is optional and defaults to *0*.

<*inclk input*> is one of *{dqs_bus, normal}* and describes whether the **inclk** input ports should be fed by the DQS_BUS where possible or not. If set to *dqs_bus*, the fitter must use the dedicated DQS bus to connect to the **inclk** port, or give an error if it cannot. If set to *normal*, the normal **inclk** input feeds the positive-edge triggered register. This parameter is optional and defaults to *normal*.

<*ddioinclk input*> is one of *{dqsb_bus, negated_inclk}* and describes whether the **ddioinclk** input ports should be fed by the DQS#_BUS where possible or not. If set to *dqsb_bus*, the fitter must use the dedicated DQS# bus to connect to the **ddioinclk** port, or give an error. If set to *negated_inclk*, the **ddioinclk** port is ignored and the negative-edge triggered DDIO register will be fed by a negated signal from the **inclk** port. This parameter is optional and defaults to *negated_inclk*.

<*dqs offsetctrl enable*> is one of *{true, false}* and describes whether the **offsetctrlin** input is used. This parameter is optional and defaults to *false*.

<*dqs ctrl latches enable*> is one of *{true, false}* and describes whether the **delayctrlin** and **offsetctrlin** inputs are latched or not. This parameter is optional and defaults to *false*. If set to *true*, then only a DLL can feed the **delayctrlin** input bus.

<*dqs edge detect enable*> is one of *{true, false}* and describes whether the edge detection circuit prevent updates to the delayctrlin and offsetctrlin latches during a DQS transition. This parameter is optional and defaults to *false*. This parameter cannot be true if *dqs_ctrl_latches_enable* is set to *false*.

<*gated dqs*> is one of *{true, false}* and describes whether the DQS input signal should be gated by an AND gate fed by an I/O input register before outputting on the **dqsbusout** port. The AND is used to hold the DQS input low after the read post-amble phase to ensure that the 0 to Z transition doesn't trigger the DQ registers. To use this feature, the *input_register_mode* parameter must be set to *register*, the *input_sync_reset* parameter must be set to *clear*, and the *input_async_reset* parameter must be set to *preset*. Also, the **sreset** input port must be tied to VCC, the **dqsbusout** output must be inverted and fed back to the **inclk** input port, and the **areset** input port should be used to control the input register feeding the AND gate. The **areset** input should normally be asserted, and since it is a preset, the input register is kept high and the gate is transparent. To switch the gate off, the **areset** input should be released to allow the register to enter the low state on the last falling DQS transition. This would keep the DQS low after the post-amble for a full cycle. This parameter is optional and defaults to *false*.

<*sim_dqs_intrinsic_delay*> is the intrinsic delay of the DQS delay chain in ps. This parameter should only use by the netlist writers to pass the delay loop parameter to the simulation model. This field is optional and defaults to 0. This parameter is only used for simulation.

<*sim_dqs_delay_increment*> is the delay increment of the DQS delay chain in ps. This parameter should only use by the netlist writers to pass the delay loop parameter to the simulation model. This field is optional and defaults to 0. This parameter is only used for simulation.

<sim_dqs_offset_increment> is the offset increment of the DQS delay chain in ps. This parameter should only use by the netlist writers to pass the delay loop parameter to the simulation model. This field is optional and defaults to 0. This parameter is only used for simulation.

4 Backward Compatibility with Stratix

The mapping of Stratix DLL ports and parameters to Stratix II DLL ports and parameters is straightforward. The ports new to the Stratix II DLL are all optional and can be left unconnected. The Stratix DLL **delayctrlout**->Stratix I/O **delayctrlin** connection has now been expanded from a single bit to a bus; the WYSIWYG conversion should take care to expand the bus. The parameters new to the Stratix II DLL can be left at default settings, except for the DQS parameter dqs_ctrl_latches_enable, which should be set to true. Because of the ctrl latches, the conversion functions will also need to connect the dqsupdate output from the DLL to the dqsupdateen input on all its associated DQS. The *sim_invalid_lock* parameter is obsolete in Stratix II (simulation of the Stratix II DLL will be dynamic instead of static). The only special case is if the user has specified a *phase_shift* setting of 72 degrees which is not natively handled by the Stratix II DLL. In that case, the Stratix II DLL will be set to 67.5 degrees, a constant offset of +4.5 degrees will be enabled on the **offsetctrlout** output, and the **offsetctrlout** output bus will be connected to the DQS pins' **offsetctrlin** inputs.

The mapping of the DQS-specific Stratix I/O ports and parameters to Stratix II I/O ports and parameters is also relatively simple. The Stratix II I/O *dqs_out_mode* is set according to the Stratix I/O *dll_phase_shift* parameter as follows*: {unused: none, 0: bypass, 72: delay_chain<n>, 90: delay_chain<n>*}, where <n> is the number given in the following formula rounded down:

$$n = dll\_delay\_chains * phase\_shift/360$$

The Stratix II I/O **dqsbusout** port should have the same fan-out as the comb_out port on the Stratix I/O if *dll_phase_shift* is not set to *unused*. Otherwise, the **dqsbusout** port should be left unconnected. Also, the Stratix II I/O **combout** port should have the same fan-out as the dqsundelayedout port on the Stratix I/O.