CycloneIII/O WYSIWYG Description

Version 1.3

May 27, 2005

By

Altera Corporation

Table of Contents:

1. Overview

This document describes the Cyclone II I/O WYSIWYG primitive. The Cyclone II I/O element is almost identical to the Cyclone I/O element in terms of functionality except the new addition of sneak LVDS paths. There is no DDIO mode like the Stratix® and Stratix II I/O element. Each I/O element only contains 3 registers (input register, output register and output-enable register).

2. Cyclone II I/O Cell

Since the Cyclone II I/O cell is very similar to the Cyclone I/O element, the WYSIWYG and atom models are mostly the same as Cyclone device with additional ports and parameters to model the new LVDS sneak paths.

The following is the definition of the Cyclone II I/O primitive.

2.1 Cyclone II I/O Primitive

```
cycloneii_io <I/O name>
(
        // Inputs
        .datain(<output source>),
        .oe(<oe source>),
        .outclk(<output clock source>),
        .outclkena(<output clock-enable source>),
        .inclk(<input register clock source>),
        .inclkena(<input register clock-enable source>),
        .areset(<asynchronous set/reset source>),
        .sreset(<synchronous set/reset source>),

        // Dedicated inputs for LVDS sneak paths
        .differentialin(<differential input>),

        // Dedicated input for differential DQS
        .linkin(<linkin source>),

        // Bidirectional Port
        .padio(<pad I/O source/output>),

        // Outputs
        .combout(<combinational output>),
        .regout(<input register output>),

        // Dedicated outputs for LVDS sneak paths
        .differentialout(<differential output>),

        // Dedicated output for differential DQS
        .linkout(<linkout output>),
```

```
        // Buried ports: placeholder to keep buried register names
        .dffdataout(<buried output register>),
        .dffoe(<buried output-enable register>)
    );
    defparam <I/O name>.operation_mode = <operation mode>;
    defparam <I/O name>.open_drain_output = <open drain mode>;
    defparam <I/O name>.bus_hold = <bus hold mode>;

    defparam <I/O name>.output_register_mode = <output register
        mode>;
    defparam <I/O name>.output_async_reset = <output register
        asynchronous reset mode>;
    defparam <I/O name>.output_power_up = <output register power up
        mode>;
    defparam <I/O name>.output_sync_reset = <output register
        synchronous reset mode>;
    defparam <I/O name>.tie_off_output_clock_enable = <tie off output
        clock enable>;

    defparam <I/O name>.oe_register_mode = <oe register mode>;
    defparam <I/O name>.oe_async_reset = <oe register asynchronous
        reset mode>;
    defparam <I/O name>.oe_power_up = <oe register power up mode>;
    defparam <I/O name>.oe_sync_reset = <oe register synchronous
        reset mode>;
    defparam <I/O name>.tie_off_oe_clock_enable = <tie off oe clock
        enable>;

    defparam <I/O name>.input_register_mode = <input register mode>;
    defparam <I/O name>.input_async_reset = <input register
        asynchronous reset mode>;
    defparam <I/O name>.input_power_up = <input register power up
        mode>;
    defparam <I/O name>.input_sync_reset = <input register
        synchronous reset mode>;
    defparam <I/O name>.use_differential_input = <use the
        differential input>;
```

2.2 Cyclone II I/O Input Ports

<*I/O name*> is the unique identifier for the I/O element. This is any identifier name which is legal for the given description language (e.g. Verilog, VHDL, AHDL, etc.). *This field is required.*

.**datain (**<*output source*>**)** is the data input to the I/O element.

.oe(<*oe source*>**)** is the output enable signal for the I/O element. This should not be specified when the I/O element is in “input” mode. It defaults to VCC (permanently enabled) in output mode, and it should not be specifically set to VCC.

.outclk(<*output clock source*>**)** designates the output-clock input to the I/O element, which drives the clock input on the output and oe registers. If this signal is used, then at least one of the output and oe registers must be in “register” mode.

.outclkena(<*output clock-enable source*>**)** designates the clock-enable signal for the output and oe registers in the I/O element. Only allowed when the .outclk port is specified and the output or oe is in “register” mode.

.inclk(<*input clock source*>**)** designates the clock input to the input register in the I/O element. Only allowed when .regout is connected.

.inclkena(<*input clock-enable source*>**)** designates the clock-enable signal for the input register in the I/O element. Only allowed when the .inclk port on the logic element is specified and **.regout** is connected.

.areset(<*asynchronous set/reset source*>**)** designates the asynchronous set/reset signal for the I/O element. Each set of registers (input, output and OE) in the I/O element can independently choose to use the .areset signal as either a clear or a preset or neither. Only allowed when at least one of the .outclk or .inclk ports on the logic element is specified, and when at least one of the input, output, or oe register’s asynchronous reset mode is “preset” or “clear.”

.sreset(<*synchronous set/reset source*>**)** designates the synchronous set/reset signal for the I/O element. Each set of registers (input, output and OE) in the I/O element can independently choose to use the .sreset signal as either a clear, a preset or neither. Only allowed when at least one of the .outclk or .inclk ports on the logic element is specified, and when at least one of the input, output or oe register’s synchronous reset mode is “preset” or “clear.”

.differentialin(<*differential input*>**)** designates the differential input port from its adjacent positive IO partner. This input is then mux with the normal pad input to provide the input to the IO register or to the core. This port is meant for implementing the soft DDIO input registers in two adjacent IO registers.

.linkin(<*linkin source*>**)**, this port is mainly provided to allow association of a differential DQS pin pair. A negative differential DQS pin can have the linkin port connect to the linkout port of its positive counterpart.

2.3 Cyclone II I/O Output Ports

.combout(<*combinational output*>**)** is the combinatorial output of the I/O element. It always feeds directly from the .padio.

.regout(<*registered output*>**)** is the registered output of the I/O element. Only allowed when the .inclk port on the logic element is specified. It is always sourced from the input register.

.differentialout(<*differential output*>**)** designates the differential output port to its adjacent negative IO partner. This output port must only feed the *differentialin* port of its adjacent IO.

.linkout(<*linkout output*>**)**, this port is mainly provided to allow association of a differential DQS pin pair. A positive differential DQS pin can have the linkout port connect to the linkin port of its negative counterpart.

2.4 Cyclone II I/O Bidirectional Ports

.padio(<*pad I/O source/output*>**)** represents the physical pad of the I/O element. A bi-directional port, this signal should be connected directly to module inputs and outputs. Logic feeding out between these ports and the top-level module outputs is illegal.

The following Example demonstrates how the .padio signal is used to connect up to input, output and bidir signals.

Example 1 – Cyclone II .padio usage in different modes

```
module foo ( input_signal, output_signal, bidir_signal )
      input input_signal;
      output output_signal;
      inout bidir_signal;
      cycloneii_io input_io (.padio(input_signal), ...);
            defparam input_io.operation_mode = “input”;
            ...
      cycloneii_io bidir_io (.padio(bidir_signal), ...)
            defparam bidir_io.operation_mode = “bidir”;
            ...
      cycloneii_io output_io (.padio(output_signal), ...)
            defparam output_io.operation_mode = “output”;
            ...
end module
```

2.5 Cyclone II I/O Buried Ports

The following ports are placeholder to keep names for buried registers. They should not have real fan-outs.

.dffdataout(<*buried output register*>**)** is the placeholder output to keep the name of buried output register.

.dffoe(<*buried output-enable register*>**)** is the placeholder output to keep the name of buried output-enable register.

2.6 I/O Modes

<*operation_mode*> is one of *{input, output, bidir}*. Determines the directionality of the I/O element. *This field is required.*

<*open_drain_mode*> is one of *{true, false}*. This field is optional and defaults to *false*. This field can only be set to *true* when the I/O is in output or bidir mode but not input mode.

<*bus_hold_mode*> is one of *{true, false}*. This option is used to always enable the bus hold circuitry in user mode. This field is optional and defaults to *false*.

<*output_register_mode*> is one of *{register or none}*. This field is optional, and defaults to none. This determines if the .**datain** signal should be registered or not. The .**outclk** port is required if this mode is *register*.

<*output_async_reset*> is one of *{clear, preset or none}*. This field is optional, and defaults to none. This determines if the .**areset** port clears, presets, or has no effect on the output register(s). The .**areset** port is required if this mode is *clear* or *preset*.

<*output_power_up*> is one of *{high, low}* and describes the power-up condition of the output register(s). This field is optional and defaults to low.

<*output_sync_reset*> is one of *{clear, preset or none}*. This field is optional, and defaults to none. This determines if the .**sreset** port clears, presets, or has no effect on the output register(s). The .**sreset** port is required if this mode is *clear* or *preset*.

<*tie off output clock enable*> is one of *{true, false}* and determines if the clock enable for the output register, controlled by **.outclkena**, should be tied off (has no effect on the register). This field is optional and defaults to false.

<*oe_register_mode*> is one of *{register or none}*. This field is optional, and defaults to none. This determines if the .**oe** signal (output enable source) is registered or not. The .**outclk** port is required if this mode is *register*.

<*oe_async_reset*> is one of *{clear, preset or none}*. This field is optional, and defaults to none. This determines if the .**areset** port clears, presets, or has no effect on the oe register. The .**areset** port is required if this mode is *clear* or *preset*.

<*oe_power_up*> is one of *{high, low}* and describes the power-up condition of the oe register. This field is optional and defaults to low.

<*oe_sync_reset*> is one of *{clear, preset or none}*. This field is optional, and defaults to none. This determines if the .**sreset** port clears, presets, or has no effect on the oe register. The .**sreset** port is required if this mode is *clear* or *preset*.

<*tie off oe clock enable*> is one of *{true, false}* and determines if the clock enable for the oe register, controlled by **.outclkena** should be tied off (has no effect on the register). This field is optional and defaults to false.

<*input_register_mode*> is one of *{register or none}*. This field is optional, and defaults to none. This determines if the .**regout** signal is used or not. The .**inclk** port is required if this mode is *register*. Although the use of input register can be inferred from the connection of .**regout** port, we still want to have an explicit parameter to indicate if input register is used or not since formal verification tool can't instantiate the model based on connections.

<*input_async_reset*> is one of *{clear, preset or none}*. This field is optional, and defaults to none. This determines if the .**areset** port clears, presets, or has no effect on the input register(s). The .**areset** port is required if this mode is *clear* or *preset*.

<*input_power_up*> is one of *{high, low}* and described the power-up condition of the input register(s). This field is optional and defaults to low.

<*input_sync_reset*> is one of *{clear, preset or none}*. This field is optional, and defaults to none. This determines if the .**sreset** port clears, presets, or has no effect on the input register(s). The .**sreset** port is required if this mode is *clear* or *preset*.

<***use_differential_input***> is one of *{true, false}* and described if the differential input should be used as input signal or the normal pad input should be used. If it's true, the pin will use the differentialin signal as its source to the IO register or the core. Otherwise, it will use the pin itself to feed the IO register or the core. This field is optional and defaults to false.

Note: An I/O register can use only one of asynchronous clear or preset. If an I/O register uses asynchronous clear the power-up state must be low, if an I/O register uses asynchronous preset the power-up state must be high. If neither asynchronous preset nor clear is used the power-up state can be high or low.

An IO register can use both an asynchronous preset or clear and a synchronous preset or clear. The setting of the synchronous preset or clear has no effect on the power-up state of the register.

2.7 I/O Polarities and Default Values

Table 1 – Polarity of I/O Inputs

Signal	*Polarity*	*Programmable Inversion*
.datain	--	Yes
.oe	Active high	Yes
.outclk	Rising edge	Yes
.outclkena	Active high	Yes
.inclk	Rising edge	Yes
.inclkena	Active high	Yes
.areset	Active high	Yes
.sreset	Active high	Yes

If not explicitly set in the device primitive instantiation, signals default to unconnected on the I/O element. Output enable will default to GND when the I/O element is in “input” mode, and VCC when it is in “output” mode. It should be left unconnected in “input” mode, and it should not be set to VCC in “output “ mode.

All Cyclone II I/O element inputs have programmable inversion, and hence can be provided in either polarity.

2.8 Basic I/O Mode Diagrams

The diagrams below show the basic configurations of the I/O element, showing all possible registers in each configuration. OE and output registers may be removed by changing the appropriate register mode to “none”, while the input register is bypassed by using the .combout port instead of the .regout port.

Any register can use one of asynchronous preset or clear (but not both) by setting the register’s asynchronous reset mode. The asynchronous preset or clear is fed by the .areset port. Similarly, any register can use one of synchronous preset or clear by setting the register’s synchronous reset mode. The synchronous preset or clear is fed by the .sreset port.

Any diagram showing a TRI primitive may have an OPNDRN primitive fed by, or in place of, the TRI primitve by setting open_drain_output = “true.”

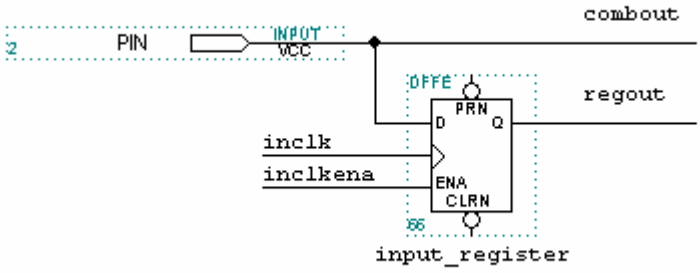

Figure 7: operation_mode = ”input”; either or both of **.combout** and **.regout** can be used. When .regout is used, input_register_mode = “register”. Otherwise, input_register_mode = “none”.

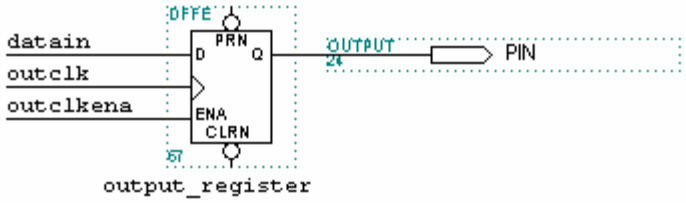

Figure 8: operation_mode = “output”; output_register_mode = “register.” This is the non tri-stated version of the output mode (.oe is disconnected).

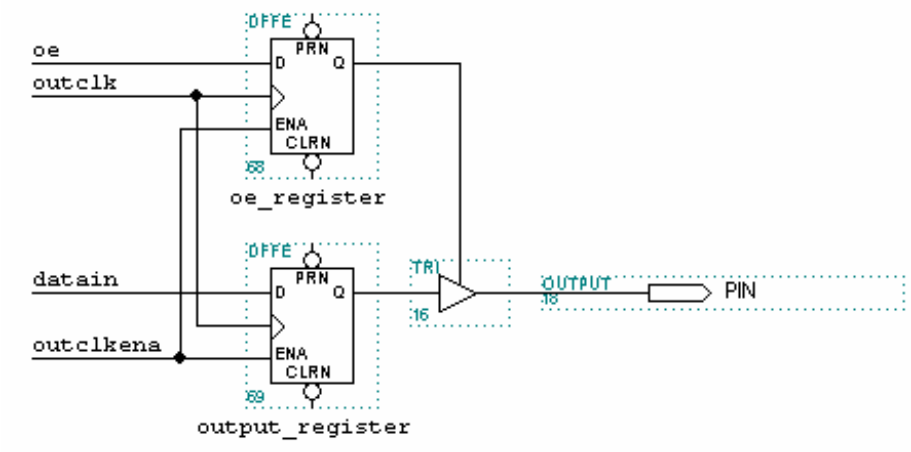

Figure 9: operation_mode = “output”; output_register_mode = “register”; oe_register_mode = “register.” This is the tri-state version of the output mode (**.oe** is connected).

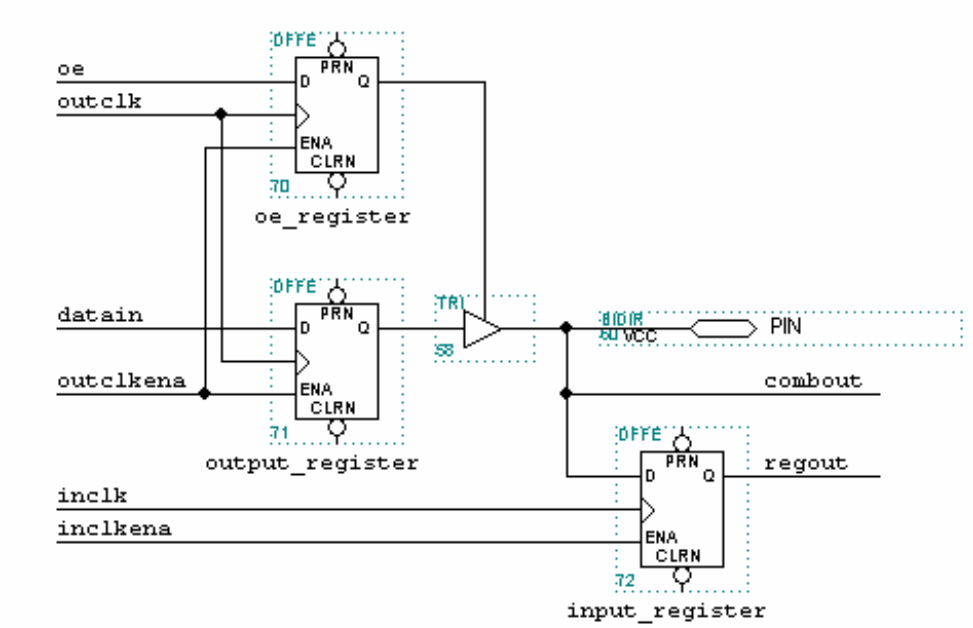

Figure 10: operation_mode = “bidir”; output_register_mode = “register”; oe_register_mode = “register”; either or both of .combout and .regout can be used. When .regout is used, input_register_mode = “register”. Otherwise, input_register_mode = “none”.

3. Backward Compatiblity with a Cyclone Design

The Cyclone II family is a successor of the Cyclone family, so we should allow easy migration of Cyclone designs to Cyclone II designs. The mapping of the Cyclone I/O to the Cyclone II I/O should be trivial since they are identical.