Stratix II Megafunction Library Functional Description

Version 1.4

April 21, 2006

by

Altera Corporation

Author:	Altera Corporation
System:	
Subsystem:	
Keywords:	Stratix II megafunctions

Table of Contents:

1. OVERVIEW....3

2. NEW MEGAFUNCTIONS....3

3. SYNTHESIS-RELEVENT MEGAFUNCTIONS....3

3.1 Megafunctions with no behavioral changes since Stratix 3
3.1.1 altaccumulate 3
3.1.2 altshift_taps 4
3.1.2.1 Megafunction implementation 5
3.1.3 lpm_add_sub 6
3.1.4 lpm_compare 6
3.1.5 lpm_counter 6
3.1.6 lpm_divide 6
3.1.7 lpm_mult 6
3.2 altmult_accum 6
3.2.1 MegaWizard plug-in 7
3.2.2 Parameter and port list 7
3.2.3 Parameter definitions 9
3.2.4 Port definitions 13
3.3 altmult_add 14
3.3.1 MegaWizard plug-in 15
3.3.2 Parameter and port list 15
3.3.3 Parameter definitions 17
3.3.4 Port definitions 23
3.4 altsyncram 24
3.4.1 Parameter and port list 24
3.4.2 Parameter Definitions 26
3.4.3 Port definitions 28
3.5 parallel_add 29
3.5.1 MegaWizard plug-in 29
3.5.2 Megafunction Ports and Parameters 29
3.6 altclkbuf 30
3.6.1 Behavior Diagram 30
3.6.2 MegaWizard plug-in 30
3.6.3 Megafunction Ports and Parameters 30
3.6.4 Parameter and Port Definitions 30

4. NON-INFERABLE MEGAFUNCTIONS 31

4.1 altddio_in, altddio_out, altddio_bidir 31
4.2 altdqs 31
4.3 altlvds_rx and altlvds_tx 31
4.4 altmemmult 31
4.5 altpll 32
4.6 altpll_reconfig 39
4.7 dcfifo 39

1. Overview

This document gives an overview of the new or updated megafunctions in Quartus II 3.1. Most of the updates are needed to support Stratix II hardware features.

2. New Megafunctions

Many updates to existing megafunctions will be needed to support Stratix II hardware features. Most of the updated megafunctions will be backward compatible with existing Stratix, Stratix GX, and Cyclone designs. The few exceptions are for rarely used megafunctions such as *altremote_update,* and *altpll_reconfig.*

To generate a purely structural clearbox netlist, the parameter CBX_STRUCTURAL_NETLIST=ON should be specified. The generated netlist would then be devoid of any behavioral constructs.

3. Synthesis-Relevent Megafunctions

Arithmetic or memory megafunctions are grouped here since they are of particular interest to synthesis tools. Unless otherwise noted, these megafunctions will have Clearbox support.

3.1 Megafunctions with no behavioral changes since Stratix

Many megafunctions, including all LPM megafunctions, have no behavioral changes when compared with Stratix. For all of these functions, the port, parameter, and MegaWizard plug-in will remain the same.

3.1.1 altaccumulate

This is the accumulator megafunction. It is implemented using ALEs. The Stratix II behavior is identical to the Stratix behavior.

Parameter and port list

```
component altaccumulate
        generic (
                extra_latency           : natural := 0;
                lpm_representation      : string := "UNSIGNED"; -- "UNSIGNED" or "SIGNED"
                right_shift_distance    : natural := 0;   -- clearbox internal use only parameter
                use_wys                 : string := "ON";          -- no behavioral impact
                width_in                : natural;
                width_out               : natural
        );
        port(
                data            : in std_logic_vector(width_in-1 downto 0);
                sign_data       : in std_logic := '0';      -- used to dynamically change signs
                add_sub         : in std_logic := '1';      -- 1 means add, 0 means subtract
```

```
                cin             :  in std_logic := '0';
                clken           :  in std_logic := '1';
                clock           :  in std_logic;
                sload           :  in std_logic := '0';
                aclr            :  in std_logic := '0';
                result          :  out std_logic_vector(width_out-1 downto 0);
                cout            :  out std_logic;
                overflow        :  out std_logic
        );
end component;
```

Parameter definitions

right_shift_distance: 0 means normal accumulator behavior. Non-zero values will result in shift toward the LSB of the given number of bits on each clock cycle. This is especially useful as a building block for multi-cycle shift-and-accumulate multipliers. This parameter is for clearbox internal use and it is not exposed through the megawizard or the megafunction interface.

use_wys: For some device families, chooses an implementation based on LE WYSIWYGs, rather than the alternate carry_sum-based implementation. Ignored for Stratix II.

Port definitions

sign_data: 1 is signed 0 is unsigned. LPM_REPRESENTATION should be unused when sign_data is used.

cout: Carry-out of the accumulator. This is an unregistered output (i.e. it changes between clock edges). Useful for pipelining and chaining multiple accumulators.

3.1.2 altshift_taps

This is the RAM-based shift register with taps megafunction. 'Taps' means that data from the shift register is observed only at certain points only the shift-register chain. The tap points must be exactly evenly spaced (see **tap_distance** parameter). This megafunction is not Stratix II-specific. Any device family with simple dual-port RAM will be supported.

Parameter and port list

```
component altshift_taps
        generic (
                number_of_taps          :  natural;
                power_up_state          :  string := "CLEARED"; -- "CLEARED" , "DONT_CARE"
                tap_distance            :  natural;
                width                   :  natural
        );
        port(
                clken    :      in std_logic := '1';
                clock    :      in std_logic;
                shiftin  :      in std_logic_vector(width-1 downto 0);
                shiftout :      out std_logic_vector(width-1 downto 0);
                taps     :      out std_logic_vector(width*number_of_taps-1 downto 0)
        );
```

```
end component;
```

Parameter definitions

number_of_taps: The number of evenly spaced points in the shift register where intermediate states are visible. In terms of the RAM-based implementation, this parameter corresponds to the number of RAM bits per word that will be used.

tap_distance: Spacing between taps in shift positions (i.e. clock cycles). In terms of the RAM-based implementation, this parameter corresponds to the number of RAM words that will be used.

width: The number of bits per position in the shift register. This can also be described as the number of parallel single-bit shift registers.

power_up_state: Controls whether the shift register powers up with zero contents (“CLEARED”) or with unknown contents (“DONT_CARE”). The value “CLEARED”, which is the default, forces the use of M512 or M4K RAM blocks in Stratix II and Stratix. “DONT_CARE” allows the use of M-RAM blocks.

3.1.2.1 Megafunction implementation

The altshift_taps megafunction implementation was changed in Quartus II 3.0 to ensure that the read and write pointers are always in sync, even if some parts of a chip see Power-On-Reset for longer periods than other parts of the chip. Read and write addresses are now derived from a single counter. For Stratix II and Stratix device families, the simple dual port RAM will return the OLD_DATA in Simple dual port mode for M4K and M512 blocks if read and write address are the same. This avoids the use of two counters as in the previous altshift_taps implementation. For pre-Stratix families, an adder circuit that powers up at 2 is used to feed the read address.

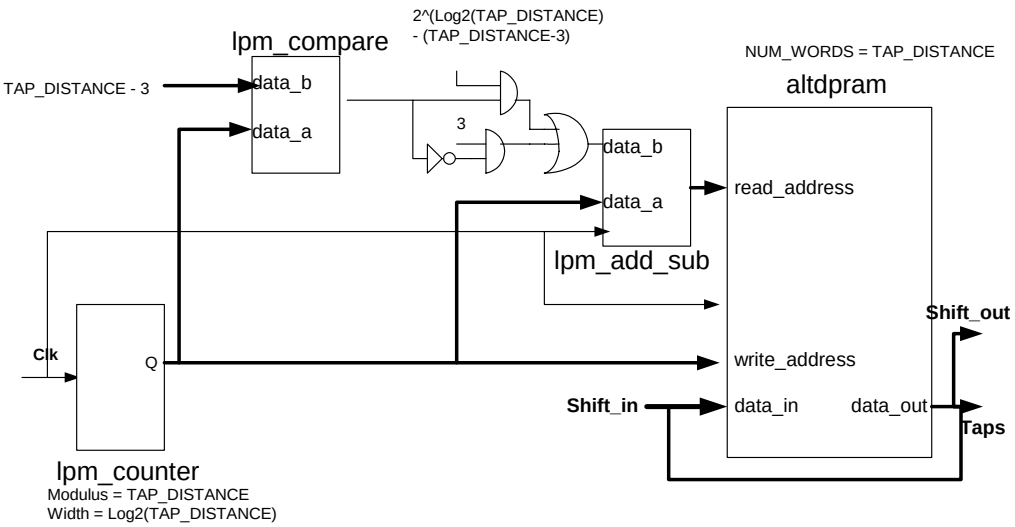


Figure 2: Implementation of altshift_taps for pre-Stratix families

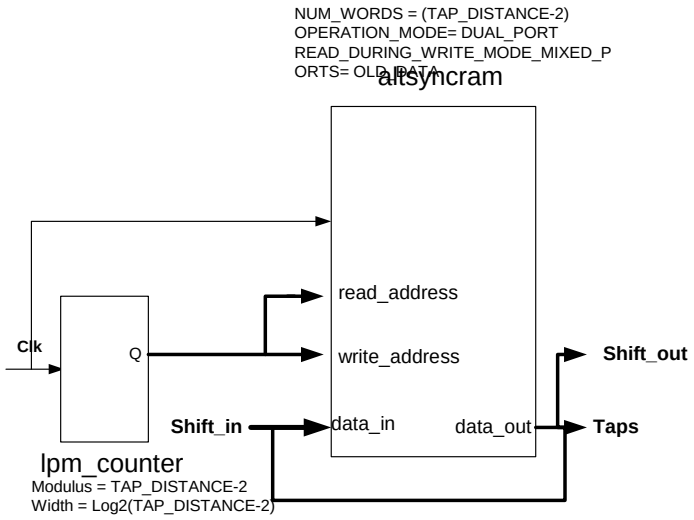


Figure 3: Implementation of altshift_taps for Stratix II and Stratix families

3.1.3 lpm_add_sub

This ALE-based adder/subtractor megafunction consists of "+" and "-" operators, which really means that this logic will be left to the synthesis tool.

3.1.4 lpm_compare

This ALE-based comparator megafunction consists of HDL-native comparison operators such as ">" and "=". This means that this logic will be left to the synthesis tool.

3.1.5 lpm_counter

This ALE-based counter megafunction will most likely be implemented using Stratix II WYSIWYG primitives.

3.1.6 lpm_divide

This ALE-based divider megafunction remain the same as in Stratix, except that "-" operators will be used instead of lpm_add_sub.

3.1.7 lpm_mult

This is the basic multiplier megafunction. It will support DSP Block-based implementations, ALE-based implementations, and most likely RAM-based implementations. The behavior will remain unchanged from previous device families, regardless of the implementation.

3.2 altmult_accum

This is the multiply-accumulate megafunction. From a behavioral perspective, it consists of a single multiplier feeding an accumulator. The user specifies widths and registering options. If the

widths exceed those that can be supported by Stratix II hardware, the megafunction will add logic as needed. The main exception is that if the new Stratix II rounding or saturation features are used, support is limited to that available in native DSP blocks.

New features in Stratix II include an input for upper data bits when loading the accumulator, scanin ports for use with the dynamic scan-chains, rounding and saturation. Dynamic scan chains means that the input source for both A and B inputs can be selected dynamically. There are new parameters, INPUT_SOURCE_A and INPUT_SOURCE_B, with values of “DATAA”, “SCANA”, or “VARIABLE”, and “DATAB”, “SCANB”, or “VARIABLE” respectively, to select the multiplier input source.

3.2.1 MegaWizard plug-in

Some MegaWizard plug-in updates will be needed to support the new features.

3.2.2 Parameter and port list

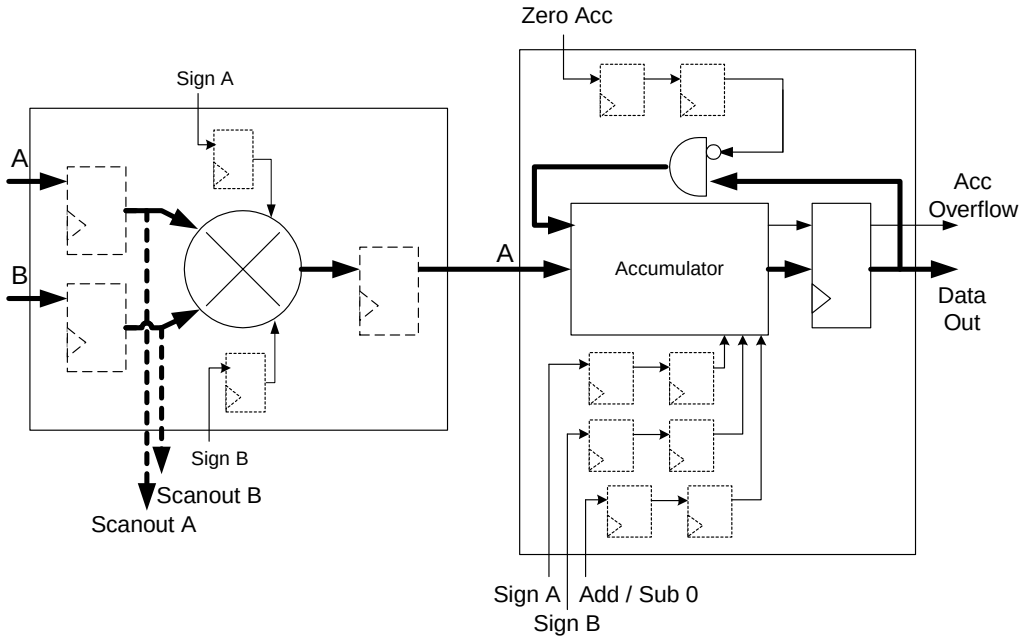


Figure 3-1: This diagram taken from the Stratix MAC Feature FD gives a good view of the parameter options.

```
PARAMETERS
(
    WIDTH_A,
    WIDTH_B,
    WIDTH_RESULT,
    WIDTH_UPPER_DATA,          -- Upper data width for sload, Stratix II-only!

    INPUT_SOURCE_A,            -- “DATAA”, “SCANA”, or “VARIABLE”, Stratix II-only!
    INPUT_SOURCE_B,            -- “DATAB”, “SCANB”, or “VARIABLE”, Stratix II-only!
    INPUT_REG_A,
    INPUT_ACLR_A,
    INPUT_REG_B,
    INPUT_ACLR_B,
    ADDNSUB_REG,
```

```
    ADDNSUB_ACLR,
    ADDNSUB_PIPELINE_REG,
    ADDNSUB_PIPELINE_ACLR,
    ACCUM_DIRECTION = "UNUSED",
    ACCUM_SLOAD_REG,
    ACCUM_SLOAD_ACLR,
    ACCUM_SLOAD_PIPELINE_REG,
    ACCUM_SLOAD_PIPELINE_ACLR,
    ACCUM_SLOAD_UPPER_DATA_REG,                    Stratix II-only!
    ACCUM_SLOAD_ UPPER_DATA_ACLR,                  Stratix II-only!
    ACCUM_SLOAD_ UPPER_DATA_PIPELINE_REG,          Stratix II-only!
    ACCUM_SLOAD_ UPPER_DATA_PIPELINE_ACLR,         Stratix II-only!
    ACCUM_ROUND_REG,                               Stratix II-only!
    ACCUM_ROUND _ACLR,                             Stratix II-only!
    ACCUM_ROUND _PIPELINE_REG,                     Stratix II-only!
    ACCUM_ROUND _PIPELINE_ACLR,                    Stratix II-only!
    ACCUM_SATURATION_REG,                          Stratix II-only!
    ACCUM_ SATURATION _ACLR,                       Stratix II-only!
    ACCUM_ SATURATION _PIPELINE_REG,               Stratix II-only!
    ACCUM_ SATURATION _PIPELINE_ACLR,              Stratix II-only!
    MULT_ROUND_REG,                                Stratix II-only!
    MULT_ROUND _ACLR,                              Stratix II-only!
    MULT_SATURATION_REG,                           Stratix II-only!
    MULT_ SATURATION _ACLR,                        Stratix II-only!
    REPRESENTATION_A = "UNSIGNED",
    SIGN_REG_A,
    SIGN_ACLR_A,
    SIGN_PIPELINE_REG_A,
    SIGN_PIPELINE_ACLR_A,
    REPRESENTATION_B = "UNSIGNED",
    SIGN_REG_B,
    SIGN_ACLR_B,
    SIGN_PIPELINE_REG_B,
    SIGN_PIPELINE_ACLR_B,
    MULTIPLIER_REG,
    MULTIPLIER_ACLR,
    OUTPUT_REG,
    OUTPUT_ACLR,
    EXTRA_MULTIPLIER_LATENCY = 0,
    EXTRA_ACCUMULATOR_LATENCY = 0,
    DEDICATED_MULTIPLIER_CIRCUITRY = "AUTO", -- no behavioral impact
    MULTIPLIER_ROUNDING = "NO",          -- "NO", "YES", "VARIABLE", Stratix II-only!
    MULTIPLIER_SATURATION = "NO",        -- "NO", "YES", "VARIABLE", Stratix II-only!
    ACCUMULATOR_ROUNDING = "NO",         -- "NO", "YES", "VARIABLE", Stratix II-only!
    ACCUMULATOR _SATURATION = "NO",      -- "NO", "YES", "VARIABLE", Stratix II-only!
    PORT_MULT_IS_SATURATED = "UNUSED",       -- "UNUSED", "USED", Stratix II-only!
    PORT_ACCUM_IS_SATURATED = "UNUSED",      -- "UNUSED", "USED", Stratix II-only!
    PORT_SIGNA = "PORT_CONNECTIVITY",
                        -- "PORT_UNUSED", "PORT_USED", "PORT_CONNECTIVITY"
    PORT_SIGNB = "PORT_CONNECTIVITY",
                        -- "PORT_UNUSED", "PORT_USED", "PORT_CONNECTIVITY"
    PORT_ADDNSUB = "PORT_CONNECTIVITY",
                        -- "PORT_UNUSED", "PORT_USED", "PORT_CONNECTIVITY"
    INTENDED_DEVICE_FAMILY = "Stratix"           -- default to Stratix behavior
)
```

```
SUBDESIGN altmult_accum
(
    dataa[WIDTH_A – 1..0]                : INPUT;
    datab[WIDTH_B – 1..0]                : INPUT;

    -- Stratix II-only input ports
    accum_sload_upper_data[WIDTH_RESULT-1..WIDTH_RESULT-WIDTH_UPPER_DATA]
                                         : INPUT = GND; -- Stratix II&CycloneII-only!
    scanina[WIDTH_A-1..0]                : INPUT = GND; -- Stratix II&CycloneII -only!
    scaninb[WIDTH_B-1..0]                : INPUT = GND; -- Stratix II&CycloneII -only!
    sourcea                              : INPUT = GND; -- Stratix II&CycloneII -only!
    sourceb                              : INPUT = GND; -- Stratix II&CycloneII -only!
    mult_round                           : INPUT = GND; -- Stratix II-only!
    mult_saturation                      : INPUT = GND; -- Stratix II-only!
    accum_round                          : INPUT = GND; -- Stratix II-only!
    accum_saturation                     : INPUT = GND; -- Stratix II-only!

    -- control signals
    addnsub                              : INPUT = VCC;
    accum_sload                          : INPUT = GND;
    signa, signb                         : INPUT = GND;

    -- clock ports
    clock0, clock1, clock2, clock3       : INPUT = VCC;
    ena0, ena1, ena2, ena3               : INPUT = VCC;
    aclr0, aclr1, aclr2, alcr3           : INPUT = GND;

    -- output ports
    result[RESULT_WIDTH – 1..0]          : OUTPUT;
    overflow                             : OUTPUT;
    scanouta[WIDTH_A-1..0]               : OUTPUT;
    scanoutb[WIDTH_B-1..0]               : OUTPUT;

    -- Stratix II-only output ports
    mult_is_saturated                    : OUTPUT; -- Stratix II-only!
    accum_is_saturated                   : OUTPUT; -- Stratix II-only!
)
```

3.2.3 Parameter definitions

WIDTH_A: Width of the dataa input bus.

WIDTH_B: Width of the datab input bus.

WIDTH_RESULT: Width of the result output bus.

WIDTH_UPPER_DATA: Width of the accum_sload_upper_data input bus.

INPUT_SOURCE_A: Selects the input source for the dataa input. Default is “DATAA” for Stratix compatibility. Legal values are “DATAA”, “SCANA”, and “VARIABLE”. “VARIABLE” means that the source will be dynamically selected by the *sourcea* input.

INPUT_SOURCE_B: Selects the input source for the datab input. Default is “DATAB” for Stratix compatibility. Legal values are “DATAB”, “SCANB”, and “VARIABLE”. “VARIABLE” means that the source will be dynamically selected by the *sourceb* input.

INPUT_REG_A: Clock source for the dataa registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

INPUT_ACLR_A: Asynchronous clear source for the dataa registers. Legal values are “UNUSED”, “ACLR0“, “ACLR1”, “ACLR2”, and “ACLR3”.

INPUT_REG_B: Clock source for the datab registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, and “CLOCK2”.

INPUT_ACLR_B: Clear source for the datab input registers. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

ADDNSUB_REG: Clock source for the register on the addnsub input port. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB_ACLR: Asynchronous clear source for the addnsub input port. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

ADDNSUB_PIPELINE_REG: Clock source for the second register on the addnsub input port. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB_PIPELINE_ACLR: Asynchronous clear source for the second register on the addnsub input port. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

ACCUM_DIRECTION: Used to specify whether the accumulator will be incrementing or decrementing. When set to “ADD”, the accumulator will add the product to the present accumulator value. When set to “SUB”, the accumulator will subtract the product from the present accumulator value. When this parameter is set to “UNUSED”, the choice of whether to add or subtract is controlled dynamically through the addnsub port. It is illegal to set the ACCUM_DIRECTION to “ADD” or “SUB” and connect the addnsub port.

ACCUM_SLOAD_REG: Clock source for the accum_sload port input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ACCUM_SLOAD_ACLR: Asynchronous clear source for the accum_sload input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

ACCUM_SLOAD_PIPELINE_REG: Clock source for the second stage registers on the accum_sload input port. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

ACCUM_SLOAD_PIPELINE_ACLR: Asynchronous clear source for the second stage registers on the accum_sload input port. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

MULT_SATURATION_REG: Clock source for the mult_saturation port input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULT_ SATURATION _ACLR: Asynchronous clear source for the mult_saturation input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

MULT_ROUND_REG: Clock source for the mult_round port input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULT_ ROUND _ACLR: Asynchronous clear source for the mult_round input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

ACCUM_SATURATION_REG: Clock source for the accum_saturation port input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ACCUM_ SATURATION _ACLR: Asynchronous clear source for the accum_saturation input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

ACCUM_ SATURATION _PIPELINE_REG: Clock source for the second stage registers on the accum_saturation input port. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

ACCUM_ SATURATION _PIPELINE_ACLR: Asynchronous clear source for the second stage registers on the accum_saturation input port. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

ACCUM_ROUND_REG: Clock source for the accum_round port input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ACCUM_ ROUND _ACLR: Asynchronous clear source for the accum_round input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

ACCUM_ ROUND _PIPELINE_REG: Clock source for the second stage registers on the accum_round input port. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

ACCUM_ ROUND _PIPELINE_ACLR: Asynchronous clear source for the second stage registers on the accum_round input port. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

ACCUM_SLOAD_UPPER_DATA_REG: Clock source for the accum_sload_upper_data port input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ACCUM_SLOAD_ UPPER_DATA_ACLR: Asynchronous clear source for the accum_sload_upper_data input registers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

ACCUM_SLOAD_ UPPER_DATA_PIPELINE_REG: Clock source for the second stage registers on the accum_sload_upper_data input port. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

ACCUM_SLOAD_ UPPER_DATA_PIPELINE_ACLR: Asynchronous clear source for the second stage registers on the accum_sload_upper_data input port. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

REPRESENTATION_A: For specifying the number representation of the A input. Legal values are “UNSIGNED”, “SIGNED”, and “UNUSED”. A value of “UNSIGNED” causes the accumulator to interpret the A input as an unsigned. A value of ”SIGNED” causes the accumulator to interpret the A input as a signed two’s complement. A value of “UNUSED” allows for dynamic control of the representation through the signa port.

SIGN_REG_A: Clock source for first register on the signa signal. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

SIGN_ACLR_A: Asynchronous clear signal for the first register on the signa signal. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

SIGN_PIPELINE_REG_A: Clock source for the second register on the signa signal. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

SIGN_PIPELINE_ACLR_A: Asynchronous clear source the second register on the signa signal. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

REPRESENTATION_B: For specifying the number representation of the B input. Legal values are “UNSIGNED”, “SIGNED”, and “UNUSED”. A value of “UNSIGNED” causes the accumulator to interpret the B input as an unsigned. A value of ”SIGNED” causes the accumulator to interpret the B input as a signed two’s complement. A value of “UNUSED” allows for dynamic control of the representation through the signb port.

SIGN_REG_B: Clock source for first register on the signb signal. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

SIGN_ACLR_B: Asynchronous clear signal for the first register on the signb signal. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

SIGN_PIPELINE_REG_B: Clock source for the second register on the signb signal. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

SIGN_PIPELINE_ACLR_B: Asynchronous clear source the second register on the signb signal. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

MULTIPLIER_REG: Clock source for the register immediately after the multiplier stage. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULTIPLIER_ACLR: Asynchronous clear source for the register immediately after the multiplier stage. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

OUTPUT_REG: Clock source for the registers on the outputs. Legal values are “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”

OUTPUT_ACLR: Asynchronous clear source for the registers on the outputs. Legal values are “UNUSED”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

EXTRA_MULTIPLIER_LATENCY: Used to add latency to the multiplier portion of the circuit. If the MULTIPLIER_REG parameter specifies a clock, then that clock will be used to add the latency. If the MULTIPLIER_REG parameter is set to UNREGISTERED then clock0 will be used to add the pipelining. The extra registers are added before the multiplier output register.

EXTRA_ACCUMULATOR_LATENCY: Used to add latency to the accumulator portion of the circuit. This latency will be from the same clock as specified in the OUTPUT_REG parameter.

DEDICATED_MULTIPLIER_CIRCUITRY: Specifies whether or not to use the DSP block to implement the circuit. Legal values are “YES” , “NO”, and “AUTO”. A value of “YES” causes the circuit to be implemented using the DSP block.

MULTIPLIER_ROUNDING: Enable rounding at the output of the multiplier stage. Rounding should only be used with input that conforms to signed 1.15 fractional number representation. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the rounding feature to permanently on or off, and “VARIABLE” allow the rounding feature to be controlled dynamically using the *mult_round* input. Default is “NO” for Stratix compatibility.

MULTIPLIER_SATURATION: Enable saturation handling at the output of the multiplier stage. Useful only for signed multiplication. Saturation at the output of the multiplier can only occur if dataa and datab are both the maximum negative number. In this case, the product is a positive that is too large to be represented in the given number of bits. Turning on saturation handling will give a maximum positive result for this case. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the saturation handling feature to permanently on or off, and “VARIABLE” allow the saturation handling feature to be controlled dynamically using the *mult_saturation* input. Default is “NO” for Stratix compatibility.

ACCUMULATOR_ROUNDING: Enable rounding in the accumulator. Rounding should only be used with input that conforms to signed 1.15 fractional number representation. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the rounding feature to permanently on or off, and “VARIABLE” allow the rounding feature to be controlled dynamically using the *accum_round* input. Default is “NO” for Stratix compatibility.

ACCUMULATOR _SATURATION: Enable saturation handling in the accumulator. Saturation should only be used with input that conforms to signed 1.(x) fractional number representation, such as signed 1.15 or 1.30 representation. Turning on saturation handling will give a maximum positive or negative result in overflow situations. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the saturation handling feature to permanently on or off, and “VARIABLE” allow the saturation handling feature to be controlled dynamically using the *accum_saturation* input. Default is “NO” for Stratix compatibility.

PORT_MULT_IS_SATURATED: Indicates that the *mult_is_saturated* output should be used or unused. Legal values are “UNUSED” (default), and “USED”. If the value is “UNUSED”, the *mult_is_saturated* output will be stuck at GND.

PORT_ACCUM_IS_SATURATED: Indicates that the *accum_is_saturated* output should be used or unused. Legal values are “UNUSED” (default), and “USED”. If the value is “UNUSED”, the *accum_is_saturated* output will be stuck at GND.

PORT_SIGNA: Indicates that the signa input is used or unused. Legal values are “PORT_CONNECTIVITY” (default), “PORT_USED” and “PORT_UNUSED”. If the value is “PORT_CONNECTIVITY”, then if signa is used, it will determine the sign of input A, else the parameter REPRESENTATION_A will determine the sign. If the value is “PORT_USED”, then the port signa controls the sign of input A. If the value is “PORT_UNUSED”, then the parameter REPRESENTATION_A determines the sign of the input A.

PORT_SIGNB: Indicates that the signb input is used or unused. Legal values are “PORT_CONNECTIVITY” (default), “PORT_USED” and “PORT_UNUSED”. If the value is “PORT_CONNECTIVITY”, then if signb is used, it will determine the sign of input B, else the parameter REPRESENTATION_B will determine the sign. If the value is “PORT_USED”, then the port signb controls the sign of input B. If the value is “PORT_UNUSED”, then the parameter REPRESENTATION_B determines the sign of the input B.

PORT_ADDNSUB: Indicates that the addnsub input is used or unused. Legal values are “PORT_CONNECTIVITY” (default), “PORT_USED” and “PORT_UNUSED”. If the value is “PORT_CONNECTIVITY”, then if addnsub is used, it will determine the direction of the accumulator, else the parameter ACCUM_DIRECTION will determine the accumulator direction. If the value is “PORT_USED”, then the port addnsub controls the accumulator direction. If the value is “PORT_UNUSED”, then the parameter ACCUM_DIRECTION determines the accumulator direction.

INTENDED_DEVICE_FAMILY: Defaults to “Stratix”. Normally, this parameter would only be used for behavioral models, since megafunctions look at the Quartus-defined device family. In this case, it is used to select Stratix-compatible assumptions about port and parameter priority resolution. In the original Stratix version of altmult_accum, dynamic control inputs such as signa, when connected, took precedence over the value of conflicting parameters, in this case REPRESENTATION_A. Going forward, we prefer that all behavioral information be taken from parameter and port values, and not from connectivity information, since this requires non-standard Verilog and VHDL instantiations. The MegaWizard plug-in for altmult_accum will always write a value for this parameter starting in Quartus II 3.1. The megafunction will check to see if a value is assigned, and if none is assigned, it will revert to pre-Quartus II 3.1 port/parameter priority rules.

3.2.4 Port definitions

dataa: Bus for one of the two multiplier arguments.

datab: Bus for one of the two multiplier arguments.

addnsub: Allows for dynamic control of whether the adder will perform an add or a subtract. A high value causes an add; a low value causes a subtract.

accum_sload: This port is for causing the present value of the accumulator to go to zero. For example, if the accum_sload signal is high, then the next value stored on the outputs will be the multiplier output (assuming MULTIPLIER_REG is “UNREGISTERED”; otherwise the value in the accumulator will be the output of the multiplier registers). For accum_sload to behave as a true synchronous load, the accumulator must be in addition mode, either via the ACCUM_DIRECTION parameter, or by setting the addnsub input to ‘1’ for addition. In Stratix II

devices, the accum_sload_upper_data[] port is also added to the accumulator value. Combining the accum_sload_upper_data[] port with the multiplier output allows all bits of the accumulator to be loaded with a new value.

signa, signb: These are for dynamically specifying the number representation of the dataa and datab ports. A high value on signa/b causes the multiplier to treat dataa/b as a signed two's complement. A low value on signa/b causes the multiplier to treat dataa/b as an unsigned.

scanina: The optional input to the A scan chain when INPUT_SOURCE_A is "SCANA" or "VARIABLE".

scaninb: The optional input to the B scan chain when INPUT_SOURCE_B is "SCANB" or "VARIABLE".

mult_round: Enables multiplier stage rounding when MULTIPLIER_ROUNDING = "VARIABLE".

mult_saturate: Enables multiplier stage saturation handling when MULTIPLIER_SATURATION = "VARIABLE".

accum_round: Enables accumulator rounding when ACCUMULATOR_ROUNDING = "VARIABLE".

accum_saturate: Enables accumulator saturation handling when ACCUMULATOR _SATURATION = "VARIABLE".

accum_sload_upper_data: The input for accumulator upper data bits during a synchronous load, for Stratix II devices only. This port defaults to all '0' to give Stratix-compatible behavior.

clock0, clock1, clock2, clock3: These are the four clock inputs.

ena0, ena1, ena2, ena3: ena0, ena1, ena2, ena3 are the respective clock enables for clock0, clock1, clock2, and clock3.

aclr0, aclr1, aclr2, alcr3: These are the four asynchronous clear sources.

result: The accumulator output.

overflow: Overflow flag for the accumulator adder.

scanouta: The output of the A scan chain.

scanoutb: The output of the B scan chain.

mult_is_saturated: Indicates that saturation handling has occured. Must be explicitly enabled with the parameter PORT_MULT_IS_SATURATED = "USED".

accum_is_saturated: Indicates that saturation handling has occured. Must be explicitly enabled with the parameter PORT_ACCUM_IS_SATURATED = "USED".

3.3 altmult_add

This is the multiply-add megafunction. From a behavioral perspective, it consists of 1 or more multipliers feeding a parallel adder. The user specifies widths and registering options. If the widths exceed those that can be supported by Stratix II hardware, the megafunction will add logic as needed. The main exception is that if the new Stratix II rounding or saturation features are used, support is limited to that available in native DSP blocks.

New features in Stratix II include scanin ports for use with the dynamic scan-chains, rounding and saturation. Dynamic scan chains means that the input source for both A and B inputs can be selected dynamically. The parameters INPUT_SOURCE_A0/1/2/3 and INPUT_SOURCE_B0/1/2/3, will support a new value of "VARIABLE" for Stratix II only, in addition

to the previously support values of “DATAA” or “SCANA”, and “DATAB” or “SCANB” respectively, to select the multiplier input source.

3.3.1 MegaWizard plug-in

MegaWizard plug-in updates will be needed to support new Stratix II features.

3.3.2 Parameter and port list

```
PARAMETERS
(
    NUMBER_OF_MULTIPLIERS,
    WIDTH_A,
    WIDTH_B,
    WIDTH_RESULT,

    -- A MULTIPLIER INPUTS
    INPUT_REGISTER_A0,
    INPUT_ACLR_A0,
    INPUT_SOURCE_A0 = “DATAA”,
    INPUT_REGISTER_A1,
    INPUT_ACLR_A1,
    INPUT_SOURCE_A1 = “DATAA”,
    INPUT_REGISTER_A2,
    INPUT_ACLR_A2,
    INPUT_SOURCE_A2 = “DATAA”,
    INPUT_REGISTER_A3,
    INPUT_ACLR_A3,
    INPUT_SOURCE_A3 = “DATAA”,
    REPRESENTATION_A= “UNUSED”,
    SIGNED_REGISTER_A,
    SIGNED_ACLR_A,
    SIGNED_PIPELINE_REGISTER_A,
    SIGNED_PIPELINE_ACLR_A,

    -- B MULTIPLIER INPUTS
    INPUT_REGISTER_B0,
    INPUT_ACLR_B0,
    INPUT_SOURCE_B0 = “DATAB”,
    INPUT_REGISTER_B1,
    INPUT_ACLR_B1,
    INPUT_SOURCE_B1 = “DATAB”,
    INPUT_REGISTER_B2,
    INPUT_ACLR_B2,
    INPUT_SOURCE_B2 = “DATAB”,
    INPUT_REGISTER_B3,
    INPUT_ACLR_B3,
    INPUT_SOURCE_B3 = “DATAB”,
    REPRESENTATION_B = “UNUSED”,
    SIGNED_REGISTER_B,
    SIGNED_ACLR_B,
    SIGNED_PIPELINE_REGISTER_B,
    SIGNED_PIPELINE_ACLR_B,

    -- MULTIPLIER OUTPUTS
    MULTIPLIER_REGISTER0,
```

```
MULTIPLIER_ACLR0,
MULTIPLIER_REGISTER1,
MULTIPLIER_ACLR1,
MULTIPLIER_REGISTER2,
MULTIPLIER_ACLR2,
MULTIPLIER_REGISTER3,
MULTIPLIER_ACLR3,
MULTIPLIER1_DIRECTION,
ADDNSUB_MULTIPLIER_REGISTER1,
ADDNSUB_MULTIPLIER_ACLR1,
ADDNSUB_MULTIPLIER_PIPELINE_REGISTER1,
ADDNSUB_MULTIPLIER_PIPELINE_ACLR1,
MULTIPLIER3_DIRECTION,
ADDNSUB_MULTIPLIER_REGISTER3,
ADDNSUB_MULTIPLIER_ACLR3,
ADDNSUB_MULTIPLIER_PIPELINE_REGISTER3,
ADDNSUB_MULTIPLIER_PIPELINE_ACLR3,

-- OUTPUT
OUTPUT_REGISTER,
OUTPUT_ACLR,

EXTRA_LATENCY,
DEDICATED_MULTIPLIER_CIRCUITRY = "AUTO"  -- no behavioral impact

-- Stratix II-only
MULTIPLIER01_ROUNDING = "NO",          -- "NO", "YES", "VARIABLE", Stratix II-only!
MULT01_ROUND_REGISTER,
MULT01_ROUND_ACLR,
MULTIPLIER01_SATURATION = "NO",        -- "NO", "YES", "VARIABLE", Stratix II-only!
MULT01_SATURATION_REGISTER,
MULT01_SATURATION_ACLR,
MULTIPLIER23_ROUNDING = "NO",          -- "NO", "YES", "VARIABLE", Stratix II-only!
MULT23_ROUND_REGISTER,
MULT23_ROUND_ACLR,
MULTIPLIER23_SATURATION = "NO",        -- "NO", "YES", "VARIABLE", Stratix II-only!
MULT23_SATURATION_REGISTER,
MULT23_SATURATION_ACLR,
ADDER1_ROUNDING = "NO",                -- "NO", "YES", "VARIABLE", Stratix II-only!
ADDNSUB1_ROUND_REGISTER,
ADDNSUB1_ROUND_ACLR,
ADDNSUB1_ROUND_PIPELINE_REGISTER,
ADDNSUB1_ROUND_PIPELINE_ACLR,
ADDER3_ROUNDING = "NO",                -- "NO", "YES", "VARIABLE", Stratix II-only!
ADDNSUB3_ROUND_REGISTER,
ADDNSUB3_ROUND_ACLR,
ADDNSUB3_ROUND_PIPELINE_REGISTER,
ADDNSUB3_ROUND_PIPELINE_ACLR,
PORT_MULT0_IS_SATURATED = "UNUSED",    -- "UNUSED", "USED", Stratix II-only!
PORT_MULT1_IS_SATURATED = "UNUSED",    -- "UNUSED", "USED", Stratix II-only!
PORT_MULT2_IS_SATURATED = "UNUSED",    -- "UNUSED", "USED", Stratix II-only!
PORT_MULT3_IS_SATURATED = "UNUSED",    -- "UNUSED", "USED", Stratix II-only!

DEVICE_FAMILY = "Stratix",             -- default to Stratix behavior

PORT_ADDNSUB1,
```

```
    PORT_ADDNSUB3,
    PORT_SIGNA,
    PORT_SIGNB
)
SUBDESIGN ALTMULT_ADD
(
    dataa[NUMBER_OF_MULTIPLIERS * WIDTH_A – 1..0]      : INPUT;
    datab[NUMBER_OF_MULTIPLIERS * WIDTH_B – 1..0]      : INPUT;

    clock3, clock2, clock1, clock0                     : INPUT = VCC;
    aclr3, aclr2, aclr1, aclr0                         : INPUT = GND;
    ena3, ena2, ena1, ena0                             : INPUT = VCC;

    signa, signb                                       : INPUT = GND;
    addnsub1, addnsub3                                 : INPUT = GND;

    -- Stratix II-only input ports
    scanina[WIDTH_A-1..0]                              : INPUT = GND; -- Stratix II-only!
    scaninb[WIDTH_B-1..0]                              : INPUT = GND; -- Stratix II-only!
    sourcea[NUMBER_OF_MULTIPLIERS-1..0]                : INPUT = GND; -- Stratix II-only!
    sourceb[NUMBER_OF_MULTIPLIERS-1..0]                : INPUT = GND; -- Stratix II-only!
    mult01_round                                       : INPUT = GND; -- Stratix II-only!
    mult23_round                                       : INPUT = GND; -- Stratix II-only!
    mult01_saturation                                  : INPUT = GND; -- Stratix II-only!
    mult23_saturation                                  : INPUT = GND; -- Stratix II-only!
    addnsub1_round                                     : INPUT = GND; -- Stratix II-only!
    addnsub3_round                                     : INPUT = GND; -- Stratix II-only!

    result[WIDTH_RESULT – 1..0]                        : OUTPUT;
    scanouta[WIDTH_A-1..0]                             : OUTPUT;
    scanoutb[WIDTH_B-1..0]                             : OUTPUT;

    -- Stratix II-only output ports
    mult0_is_saturated                                 : OUTPUT; -- Stratix II-only!
    mult1_is_saturated                                 : OUTPUT; -- Stratix II-only!
    mult2_is_saturated                                 : OUTPUT; -- Stratix II-only!
    mult3_is_saturated                                 : OUTPUT; -- Stratix II-only!
)
```

3.3.3 Parameter definitions

NUMBER_OF_MULTIPLIERS: The number of multiplier which are to be added together. This value must be greater than or equal to 1.

WIDTH_A: The width of the dataa input busses.

WIDTH_B: The width of the datab input busses.

WIDTH_RESULT: Width of the result output bus.

INPUT_REGISTER_A0: Clock source for the A inputs of the first multiplier. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

INPUT_ACLR_A0: Asynchronous clear source for the A inputs of the first multiplier. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

INPUT_SOURCE_A0: Specifies the source of the A inputs to the first multiplier. Legal values are “DATAA”, “SCANA”, and “VARIABLE”. The value of “VARIABLE” is supported in Stratix II devices only.

INPUT_REGISTER_A1: Clock source for the A inputs to the second multiplier. Legal values are "UNREGISTERED", "CLOCK0", "CLOCK1", "CLOCK2", and "CLOCK3".

INPUT_ACLR_A1: Asynchronous clear source for the A inputs to the second multiplier. Legal values are "UNREGISTERED", "CLOCK0", "CLOCK1", "CLOCK2", and "CLOCK3".

INPUT_SOURCE_A1: Specifies the source for the A inputs to the second multiplier. Legal values are "DATAA", "SCANA", and "VARIABLE". The value of "VARIABLE" is supported in Stratix II devices only.

INPUT_REGISTER_A2: Clock source for the A inputs to the third multiplier. Legal values are "UNREGISTERED", "CLOCK0", "CLOCK1", "CLOCK2", and "CLOCK3".

INPUT_ACLR_A2: Asynchronous clear source the the A inputs to the third multiplier. Legal values are "NONE", "ACLR0", "ACLR1", "ACLR2", and "ACLR3".

INPUT_SOURCE_A2: Specifies the source for the A inputs to the third multiplier. Legal values are "DATAA", "SCANA", and "VARIABLE". The value of "VARIABLE" is supported in Stratix II devices only.

INPUT_REGISTER_A3: Clock source for the fourth and all subsequent multipliers. Legal values are "UNREGISTERED", "CLOCK0", "CLOCK1", "CLOCK2", and "CLOCK3".

INPUT_ACLR_A3: Asynchronous clear source for the fourth and all subsequest multipliers. Legal values are "NONE", "ACLR0", "ACLR1", "ACLR2", and "ACLR3".

INPUT_SOURCE_A3: Specifies the source of the A inputs to the fourth multiplier. Legal values are "DATAA", "SCANA", and "VARIABLE". The value of "VARIABLE" is supported in Stratix II devices only.

REPRESENTATION_A: For specifying the number representation of the A multiplier inputs. Legal values are "UNSIGNED", "SIGNED", and "UNUSED". A value of "UNSIGNED" causes the A inputs to be interpreted as unsigned numbers. A value of "SIGNED" causes the A inputs to be interpreted as signed two's complement numbers. A value of "UNUSED" allows for dynamic control of the representation through the signa port.

SIGNED_REGISTER_A: The clock source for the first signa register. Legal values are "UNREGISTERED", "CLOCK0", "CLOCK1", "CLOCK2", and "CLOCK3".

SIGNED_ACLR_A: Asynchronous clear source for the first signa register. Legal values are "NONE", "ACLR0", "ACLR1", "ACLR2", and "ACLR3".

SIGNED_PIPELINE_REGISTER_A: The clock source for the second signa register. Legal values are "UNREGISTERED", "CLOCK0", "CLOCK1", "CLOCK2", and "CLOCK3".

SIGNED_PIPELINE_ACLR_A: Asynchronous clear source for the second signa register. Legal values are "NONE", "ACLR0", "ACLR1", "ACLR2", and "ACLR3". The width of the datab input buses.

INPUT_REGISTER_B0: Clock source for the B inputs of the first multiplier. Legal values are "UNREGISTERED", "CLOCK0", "CLOCK1", "CLOCK2", and "CLOCK3".

INPUT_ACLR_B0: Asynchronous clear source for the B inputs of the first multiplier. Legal values are "NONE", "ACLR0", "ACLR1", "ACLR2", and "ACLR3".

INPUT_SOURCE_B0: Specifies the source of the B inputs to the first multiplier. Legal values are "DATAB", "SCANB", and "VARIABLE". The value of "VARIABLE" is supported in Stratix II devices only.

INPUT_REGISTER_B1: Clock source for the B inputs to the second multiplier. Legal values are "UNREGISTERED", "CLOCK0", "CLOCK1", "CLOCK2", and "CLOCK3".

INPUT_ACLR_B1: Asynchronous clear source for the B inputs to the second multiplier. Legal values are "UNREGISTERED", "CLOCK0", "CLOCK1", "CLOCK2", and "CLOCK3".

INPUT_SOURCE_B1: Specifies the source for the B inputs to the second multiplier. Legal values are “DATAB”, “SCANB”, and “VARIABLE”. The value of “VARIABLE” is supported in Stratix II devices only.

INPUT_REGISTER_B2: Clock source for the B inputs to the third multiplier. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

INPUT_ACLR_B2: Asynchronous clear source for the B inputs to the third multiplier. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

INPUT_SOURCE_B2: Specifies the source for the B inputs to the third multiplier. Legal values are “DATAB”, “SCANB”, and “VARIABLE”. The value of “VARIABLE” is supported in Stratix II devices only.

INPUT_REGISTER_B3: Clock source for the fourth and all subsequent multipliers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

INPUT_ACLR_B3: Asynchronous clear source for the fourth and all subsequest multipliers. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

INPUT_SOURCE_B3: Specifies the source of the B inputs to the fourth multiplier. Legal values are “DATAB”, “SCANB”, and “VARIABLE”. The value of “VARIABLE” is supported in Stratix II devices only.

REPRESENTATION_B: For specifying the number representation of the B multiplier inputs. Legal values are “USNIGNED”, “SIGNED”, and “UNUSED”. A value of “UNSIGNED” causes the B inputs to be interpreted as unsigned numbers. A value of “SIGNED” causes the B inputs to be interpreted as signed two’s complement numbers. A value of “UNUSED” allows for dynamic control of the representation through the signb port.

SIGNED_REGISTER_B: The clock source for the first signb register. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

SIGNED_ACLR_B: Asynchronous clear source for the first signb register. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

SIGNED_PIPELINE_REGISTER_B: The clock source for the second signb register. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

SIGNED_PIPELINE_ACLR_B: Asynchronous clear source for the second signb register. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

MULTIPLIER_REGISTER0: Clock source for the register immediately after the first mutliplier. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULTIPLIER_ACLR0: Asynchronous clear source for the register immediately after the first multiplier. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

MULTIPLIER_REGISTER1: INPUT_REGISTER_A0: Clock source for the A inputs of the first multiplier. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULTIPLIER_ACLR1: Asynchronous clear source for the register immediately after the second multiplier. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

MULTIPLIER_REGISTER2: Clock source for the register immediately after the third multiplier. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULTIPLIER_ACLR2: Asynchronous clear source for the register immediately after the third multiplier. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

MULTIPLIER_REGISTER3: Clock source for the register immediatley after the fourth and all subsequent multipliers. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULTIPLIER_ACLR3: Asynchronous clear source for the register immediately after the fourth and all subsequent multipliers. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

MULTIPLIER1_DIRECTION: Specifies whether the second multiplier will add or subtract its value from the sum. Legal values are “ADD” and “SUB”.

ADDNSUB_MULTIPLIER_REGISTER1: Clock source for the first register on the addnsub1 input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB_MULTIPLIER_ACLR1: Asynchronous clear source for the first register on the addnsub1 input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

ADDNSUB_MULTIPLIER_PIPELINE_REGISTER1: Clock source for the second register on the addnsub1 input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB_MULTIPLIER_PIPELINE_ACLR1: Asynchronous clear source for the second register on the addnsub1 input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

MULTIPLIER3_DIRECTION: Specifies whether the fourth and all subsequent odd-numbered multipliers will add or subtract their results from the total. Legal values are “ADD” and “SUB”.

ADDNSUB_MULTIPLIER_REGISTER3: Clock source for the first register on the addnsub3 input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB_MULTIPLIER_ACLR3: Asynchronous clear source for the first register on the addnsub3 input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

ADDNSUB_MULTIPLIER_PIPELINE_REGISTER3: Clock source for the second register on the addnsub3 input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB_MULTIPLIER_PIPELINE_ACLR3: Asynchronous clear source for the second register on the addnsub3 input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

OUTPUT_REGISTER: Clock source for the output register. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

OUTPUT_ACLR: Asynchronous clear source for the output register. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

EXTRA_LATENCY: Specifies latency to add to the circuit in addition to any registers which were already specified.

DEDICATED_MULTIPLIER_CIRCUITRY: Specifies whether or not to use the DSP block to implement the circuit. Legal values are “YES”, “NO”, and “AUTO”. A value of “YES” will cause an implementation using the DSP blocks.

MULTIPLIER01_ROUNDING: Enable rounding at the output of the multiplier stage, for multipliers 0 and 1. Rounding should only be used with input that conforms to signed 1.15 fractional number representation. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the rounding feature to permanently on or off, and “VARIABLE” allow the rounding feature to be controlled dynamically using the *mult01_round* input. Default is “NO” for Stratix compatibility.

MULT01_ROUND_REGISTER: Clock source for the register on the mult01_round input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULT01_ROUND_ACLR: Asynchronous clear source for the register on the mult01_round input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”

MULTIPLIER01_SATURATION: Enable saturation handling at the output of the multiplier stage, for multipliers 0 and 1. Saturation at the output of the multiplier can only occur if dataa and datab are both the maximum negative number. In this case, the product is a positive that is too large to be represented in the given number of bits. Turning on saturation handling will give a maximum positive result for this case. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the saturation handling feature to permanently on or off, and “VARIABLE” allow the saturation handling feature to be controlled dynamically using the *mult01_saturation* input. Default is “NO” for Stratix compatibility.

MULT01_SATURATION_REGISTER: Clock source for the register on the mult01_saturation input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULT01_SATURATION_ACLR: Asynchronous clear source for the register on the mult01_saturation input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”

MULTIPLIER23_ROUNDING: Enable rounding at the output of the multiplier stage, for multipliers 2 and 3. Rounding should only be used with input that conforms to signed 1.15 fractional number representation. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the rounding feature to permanently on or off, and “VARIABLE” allow the rounding feature to be controlled dynamically using the *mult23_round* input. Default is “NO” for Stratix compatibility.

MULT23_ROUND_REGISTER: Clock source for the register on the mult23_round input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULT23_ROUND_ACLR: Asynchronous clear source for the register on the mult23_round input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”

MULTIPLIER23_SATURATION: Enable saturation handling at the output of the multiplier stage, for multipliers 2 and 3. Saturation at the output of the multiplier can only occur if dataa and datab are both the maximum negative number. In this case, the product is a positive that is too large to be represented in the given number of bits. Turning on saturation handling will give a maximum positive result for this case. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the saturation handling feature to permanently on or off, and “VARIABLE” allow the saturation handling feature to be controlled dynamically using the *mult23_saturation* input. Default is “NO” for Stratix compatibility.

MULT23_SATURATION_REGISTER: Clock source for the register on the mult23_saturation input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

MULT23_SATURATION_ACLR: Asynchronous clear source for the register on the mult23_saturation input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”

ADDER1_ROUNDING: Enable rounding in the adder stage, for the adder used with multiplier 1. Rounding should only be used with input that conforms to signed 1.15 fractional number representation. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the rounding feature to permanently on or off, and “VARIABLE” allow the rounding feature to be controlled dynamically using the *addnsub1_round* input. Default is “NO” for Stratix compatibility.

ADDNSUB1_ROUND_REGISTER: Clock source for the first register on the addnsub1_round input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB1_ROUND_ACLR: Asynchronous clear source for the first register on the addnsub1_round input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”

ADDNSUB1_ROUND_PIPELINE_REGISTER: Clock source for the second register on the addnsub1_round input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB1_ROUND_PIPELINE_ACLR: Asynchronous clear source for the second register on the addnsub1_round input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

ADDER3_ROUNDING: Enable rounding in the adder stage, for the adder used with multiplier 3. Rounding should only be used with input that conforms to signed 1.15 fractional number representation. Legal values are “NO”, “YES”, and “VARIABLE”. “NO” and “YES” set the rounding feature to permanently on or off, and “VARIABLE” allow the rounding feature to be controlled dynamically using the *addnsub3_round* input. Default is “NO” for Stratix compatibility.

ADDNSUB3_ROUND_REGISTER: Clock source for the first register on the addnsub3_round input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB3_ROUND_ACLR: Asynchronous clear source for the first register on the addnsub3_round input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”

ADDNSUB3_ROUND_PIPELINE_REGISTER: Clock source for the second register on the addnsub3_round input. Legal values are “UNREGISTERED”, “CLOCK0”, “CLOCK1”, “CLOCK2”, and “CLOCK3”.

ADDNSUB3_ROUND_PIPELINE_ACLR: Asynchronous clear source for the second register on the addnsub3_round input. Legal values are “NONE”, “ACLR0”, “ACLR1”, “ACLR2”, and “ACLR3”.

PORT_MULT0_IS_SATURATED: Indicates that the *mult0_is_saturated* output should be used or unused. Legal values are “UNUSED” (default), and “USED”. If the value is “UNUSED”, the *mult0_is_saturated* output will be stuck at GND.

PORT_MULT1_IS_SATURATED: Indicates that the *mult1_is_saturated* output should be used or unused. Legal values are “UNUSED” (default), and “USED”. If the value is “UNUSED”, the *mult1_is_saturated* output will be stuck at GND.

PORT_MULT2_IS_SATURATED: Indicates that the *mult2_is_saturated* output should be used or unused. Legal values are “UNUSED” (default), and “USED”. If the value is “UNUSED”, the *mult2_is_saturated* output will be stuck at GND.

PORT_MULT3_IS_SATURATED: Indicates that the *mult3_is_saturated* output should be used or unused. Legal values are “UNUSED” (default), and “USED”. If the value is “UNUSED”, the *mult3_is_saturated* output will be stuck at GND.

DEVICE_FAMILY: Defaults to “Stratix”. Normally, this parameter would only be used for behavioral models, since megafunctions look at the Quartus-defined device family. In this case, it is used to select Stratix-compatible assumptions about port and parameter priority resolution. In the original Stratix version of altmult_add, dynamic control inputs such as signa, when connected, took precedence over the value of conflicting parameters, in this case REPRESENTATION_A. Going forward, we prefer that all behavioral information be taken from parameter and port values, and not from connectivity information, since this requires non-standard Verilog and VHDL instantiations. The MegaWizard plug-in for altmult_add will always write a value for this parameter starting in Quartus II 3.1. The megafunction will check to see if a value is assigned, and if none is assigned, it will revert to pre-Quartus II 3.1 port/parameter priority rules.

The following 4 parameters are used for formal verification support of dynamic control input ports like signa, signb, addnsub1 and addnsub3, to write out a fully connected netlist :

PORT_ADDNSUB1: Legal values are “PORT_USED”, “PORT_UNUSED” and “PORT_CONNECTIVITY” (default). If the value is “PORT_CONNECTIVITY”, then the megafunction checks if the addnsub1 port is not connected, and if true, uses the parameter value to determine the port value. If value is “PORT_USED” then the port is assumed to be connected; if “PORT_UNUSED” then the port is ignored and only the parameter value is looked at.

PORT_ADDNSUB3: Legal values are “PORT_USED”, “PORT_UNUSED” and “PORT_CONNECTIVITY” (default). If the value is “PORT_CONNECTIVITY”, then the megafunction checks if the addnsub3 port is not connected, and if true, uses the parameter value

to determine the port value. If value is “PORT_USED” then the port is assumed to be connected; if “PORT_UNUSED” then the port is ignored and only the parameter value is looked at.

PORT_SIGNA: Legal values are “PORT_USED”, “PORT_UNUSED” and “PORT_CONNECTIVITY” (default). If the value is “PORT_CONNECTIVITY”, then the megafunction checks if the signa port is not connected, and if true, uses the parameter value to determine the port value. If value is “PORT_USED” then the port is assumed to be connected; if “PORT_UNUSED” then the port is ignored and only the parameter value is looked at.

PORT_SIGNB: Legal values are “PORT_USED”, “PORT_UNUSED” and “PORT_CONNECTIVITY” (default). If the value is “PORT_CONNECTIVITY”, then the megafunction checks if the signb port is not connected, and if true, uses the parameter value to determine the port value. If value is “PORT_USED” then the port is assumed to be connected; if “PORT_UNUSED” then the port is ignored and only the parameter value is looked at.

3.3.4 Port definitions

dataa: These are the A inputs to the multipliers.

datab: These are the B inputs to the multipliers.

clock3, clock2, clock1, clock0: These are the four clock inputs.

aclr3, aclr2, aclr1, aclr0: These are the four asynchronous clear inputs.

ena3, ena2, ena1, ena0: These are the four clock enables. Ena3 corresponds to clock3; ena2 corresponds to clock2, etc.

signa, signb: These are for dynamically controlling the representation of the a and b inputs. A high value on signa/b causes the A/B inputs to be interpreted as signed two’s complement numbers. A low value on signa/b causes the A/B inputs to be interpreted as unsigned numbers.

addnsub1, addnsub3: These are for dynamically controlling whether to do an add or subtract of the corresponding multiplier. A high value on addnsub1/3 causes an addition to be performed of the second/(fourth and subsequent odd) multipliers. A low value causes a subtraction.

scanina: The optional input to the A scan chain when INPUT_SOURCE_A is “SCANA” or “VARIABLE”.

scaninb: The optional input to the B scan chain when INPUT_SOURCE_B is “SCANB” or “VARIABLE”.

mult01_round: Enables multiplier stage rounding for multiplier 0 and 1 when MULTIPLIER01_ROUNDING = “VARIABLE”.

mult23_round: Enables multiplier stage rounding for multiplier 2 and 3 when MULTIPLIER23_ROUNDING = “VARIABLE”.

mult01_saturate: Enables multiplier stage saturation handling for multiplier 0 and 1 when MULTIPLIER01_SATURATION = “VARIABLE”.

mult23_saturate: Enables multiplier stage saturation handling for multiplier 2 and 3 when MULTIPLIER23_SATURATION = “VARIABLE”.

adder1_round: Enables adder stage rounding for the adder used with multiplier 1 when ADDER1_ROUNDING = “VARIABLE”.

adder3_round: Enables adder stage rounding for the adder used with multiplier 3 when ADDER3_ROUNDING = “VARIABLE”.

result: The result of the multiply and addition.

scanouta, scanoutb: These are the outputs of the A and B scan chains.

mult0_is_saturated: Indicates that saturation handling has occured for multiplier 0. Must be explicitly enabled with the parameter PORT_MULT0_IS_SATURATED = “USED”.

mult1_is_saturated: Indicates that saturation handling has occured for multiplier 1. Must be explicitly enabled with the parameter PORT_MULT1_IS_SATURATED = “USED”.

mult2_is_saturated: Indicates that saturation handling has occured for multiplier 2. Must be explicitly enabled with the parameter PORT_MULT2_IS_SATURATED = “USED”.

mult3_is_saturated: Indicates that saturation handling has occured for multiplier 3. Must be explicitly enabled with the parameter PORT_MULT3_IS_SATURATED = “USED”.

3.4 altsyncram

The RAM blocks in Stratix II are compatible with the RAM blocks in Stratix, Stratix GX, and Cyclone device families. There are a few known differences. The only feature loss is that input registers will no longer support asynchronous clear. This is a minor issue because the asynchronous clear of input registers in Stratix did not have a visible effect on the RAM outputs.

Another difference is an enhancement in clock enable choices for RAM input and output registers on both the A and B ports. Four new parameters, *clock_enable_input_a, clock_enable_output_a, clock_enable_input_b, clock_enable_output_b,* will give the user the choice of disabling the effect of the corresponding clock enable input for these groups of registers. A related change is the addition of address stall inputs for both A and B address ports. Address stall acts like an extra clock enable input for the address registers only. It only affects the loading of address registers, and not the related read or write operation. Address stall is added to improve the efficiency in cache-miss applications.

Also, the parameter BYTE_SIZE now supports values of 1, 2, and 4 in addition to the previously supported values of 8 and 9.

MegaWizard plug-in

Minor MegaWizard plug-in updates will be needed.

3.4.1 Parameter and port list

```
PARAMETERS
(
    -- mode selection parameter
    OPERATION_MODE = "BIDIR_DUAL_PORT",

    ----------------------------
    -- port A read parameters
    ----------------------------
    WIDTH _A = 1,
    WIDTHAD _A = 1,
    NUMWORDS _A = 1,
    -- registering parameters
    OUTDATA_REG_A = "UNREGISTERED",
    -- clearing parameters
    ADDRESS_ACLR_A = "NONE",
    OUTDATA_ACLR_A = "NONE",
    CLOCK_ENABLE_INPUT_A = "NORMAL",  -- “NORMAL” or “BYPASS”, Stratix II only!
```

```
CLOCK_ENABLE_OUTPUT_A = " NORMAL ", -- “NORMAL” or “BYPASS”, Stratix II only!

----------------------------
-- port A write parameters
----------------------------
-- clearing parameters
INDATA_ACLR_A = "CLEAR0",
WRCONTROL_ACLR_A = "CLEAR0",
--Clear for the byte enable port registers which are clocked by clk0
BYTEENA_ACLR_A = “NONE”,
-- width of the byte enable ports.  If it is used, must be WIDTH_WRITE_A/8 or /9
WIDTH_BYTEENA_A = 1,


----------------------------
-- Port B read parameters
----------------------------
WIDTH _B = 1,
WIDTHAD _B = 1,
NUMWORDS _B = 1,
-- registering parameters
RDCONTROL_REG_B = "CLOCK1",
ADDRESS_REG_B = "CLOCK1",
OUTDATA_REG_B = "UNREGISTERED",
-- clearing parameters
OUTDATA_ACLR_B = "NONE",
RDCONTROL_ACLR_B = "NONE",
CLOCK_ENABLE_INPUT_B = "NORMAL",  -- “NORMAL” or “BYPASS”, Stratix II only!
CLOCK_ENABLE_OUTPUT_B = " NORMAL ", -- “NORMAL” or “BYPASS”, Stratix II only!

----------------------------
-- port B write parameters
----------------------------
-- registering parameters
INDATA_REG_B = "CLOCK1",
WRCONTROL_WRADDRESS_REG_B = "CLOCK1",
-- registering parameter for the byte enable register for port B
BYTEENA_REG_B = “CLOCK1”,
-- clearing parameters
INDATA_ACLR_B = "NONE",
WRCONTROL_ACLR_B = "NONE",
ADDRESS_ACLR_B = "NONE",
--clear parameter for byte enable port register
BYTEENA_ACLR_B = “NONE”,
-- width of the byte enable ports.  If it is used, must be WIDTH_WRITE_B/8 or /9
WIDTH_BYTEENA_B = 1,


----------------------------
-- global parameters
----------------------------
-- width of a byte for byte enables
BYTE_SIZE = 8,

-- Mixed-port feed-through mode choices are “OLD_DATA” or “DONT_CARE”
READ_DURING_WRITE_MODE_MIXED_PORTS = “DONT_CARE”,
```

```
    -- RAM block type choices are “AUTO”, “M512”, “M4K” and “M-RAM”
    RAM_BLOCK_TYPE = “AUTO”,

    -- General operation parameters
    INIT_FILE = "UNUSED",
    INIT_FILE_LAYOUT = “PORT_A”,
    MAXIMUM_DEPTH = 0
);

SUBDESIGN altsyncram
(
    wren_a                              : INPUT = GND;
    wren_b                              : INPUT = GND;
    rden_b                              : INPUT = VCC;
    data_a[WIDTH_WRITE_A - 1..0]        : INPUT = VCC;
    data_b[WIDTH_WRITE_B - 1..0]        : INPUT = VCC;
    address_a[WIDTHAD_WRITE_A - 1..0]   : INPUT = VCC;
    address_b[WIDTHAD_WRITE_B - 1..0]   : INPUT = VCC;
    addressstall_a                      : INPUT = GND; -- Stratix II only!
    addressstall_b                      : INPUT = GND; -- Stratix II only!
    --two clocks only
    clock0                              : INPUT = VCC;
    clock1                              : INPUT = VCC;
    clocken0                            : INPUT = VCC;
    clocken1                            : INPUT = VCC;
    aclr0                               : INPUT = GND;
    aclr1                               : INPUT = GND;
    byteena_a[WIDTH_BYTEENA_A-1..0]     : INPUT = VCC;
    byteena_b[WIDTH_BYTEENA_B-1..0]     : INPUT = VCC;

    q_a[WIDTH_READ_A - 1..0]            : OUTPUT;
    q_b[WIDTH_READ_B - 1..0]            : OUTPUT;

)
```

3.4.2 Parameter Definitions

OPERATION_MODE: Specifies the operation mode of the ram, It is one of the four choices “ROM”, “SINGLE_PORT”, “DUAL_PORT” or “BIDIR_DUAL_PORT”

WIDTH _A: Width of the input data and output data bus for port A

WIDTHAD_ A: Width of input address bus for port A

NUMWORDS_ A: Number of words in the view of port A

OUTDATA_REG_A: Registering option of output data register for port A, can be one of the four choices “CLOCK0”, “CLOCK1”, “UNREGISTERED” or “UNUSED”. Default is “UNREGISTERED”

ADDRESS_ACLR_A: Clear for address of port A. Can be one of three options: “CLEAR0”, “NONE” or “UNUSED”. Default is “NONE”

OUTDATA_ACLR_A: Clear for output register of port A. Can be one of four options:

"CLEAR0", "CLEAR1", "NONE" or "UNUSED". Default is "NONE"

CLOCK_ENABLE_INPUT_A: Clock enable bypass control for input registers of port A. All input registers are affected, including address, data, wren, rden, and byteena. Choices are "NORMAL" or "BYPASS". Default is "NORMAL" to maintain Stratix behavior.

CLOCK_ENABLE_OUTPUT_A: Clock enable bypass control for output register of port A. Choices are "NORMAL" or "BYPASS". Default is "NORMAL" to maintain Stratix behavior.

WRCONTROL_ACLR_A: Clear for input write enable register of port A. One of three options: "CLEAR0", "NONE" or "UNUSED". Default is "NONE"

INDATA_ACLR_A: Clear for input data register of port A. One of three options: "CLEAR0", "NONE" or "UNUSED". Default is "NONE"

BYTEENA_ACLR_A: Clear for input byte enable register for port A. One of three options: "CLEAR0", "NONE" or "UNUSED". Default is "NONE"

WIDTH_ B: Width of the input data and output data bus for port B

WIDTHAD_ B: Width of input address bus for port B

NUMWORDS_ B: Number of words in the view of port B

OUTDATA_REG_B: Registering option of output data register for port B, can be one of the four choices "CLOCK0", "CLOCK1", "UNREGISTERED" or "UNUSED". Default is "UNREGISTERED"

ADDRESS_ACLR_B: Clear for address of port B. Can be one of four options : "CLEAR0", "CLEAR1", "NONE" or "UNUSED". Default is "NONE"

OUTDATA_ACLR_B: Clear for output register of port B. Can be one of four options: "CLEAR0", "CLEAR1", "NONE" or "UNUSED". Default is "NONE"

WRCONTROL_ACLR_B: Clear for input write enable register of port B. One of four options: "CLEAR0", "CLEAR1", "NONE" or "UNUSED". Default is "NONE"

INDATA_ACLR_B: Clear for input data register of port B. One of four options: "CLEAR0", "CLEAR1", "NONE" or "UNUSED". Default is "NONE"

BYTEENA_ACLR_B: Clear for input byte enable register for port A. One of four options: "CLEAR0", "CLEAR1", "NONE" or "UNUSED". Default is "NONE"

INDATA_REG_B: Clock source for input data register. One of three options "CLOCK0", "CLOCK1" or "UNUSED". Has to be used when data_b is used. Default is "CLOCK1"

WRCONTROL_WRADDRESS_REG_B: Clock source for write control and address register during write mode for port B. One of three options "CLOCK0", "CLOCK1" or "UNUSED". Has to be used when wren_b is used and in write mode. Default is "CLOCK1".

RDCONTROL_REG_B: Clock source for rdenable register during read mode for port B. One of three options "CLOCK0", "CLOCK1" or "UNUSED". Has to be used when rden_b is used. Default is "CLOCK1"

ADDRESS_REG_B: Clock source for address register for port B. One of three options "CLOCK0", "CLOCK1" or "UNUSED". Has to be used when port B is used. Default is "CLOCK1"

OUTDATA_REG_B: Clock source for output register for port B. One of four options “CLOCK0”, “CLOCK1”, “UNREGISTERED” or “UNUSED”. Default is “UNREGISTERED”

BYTEENA_REG_B: Clock source for the byte enable register for port B. One of three options “CLOCK0”, “CLOCK1” or “UNUSED”. Has to be used when port b Byte enable is used. Default is “CLOCK1”

RDCONTROL_ACLR_B: Clear for rdenable control register of port B. One of four options “CLEAR0”, “CLEAR1”, “NONE” or “UNUSED”. Default is “NONE”

CLOCK_ENABLE_INPUT_B: Clock enable bypass control for input registers of port B. All input registers are affected, including address, data, wren, rden, and byteena. Choices are “NORMAL” or “BYPASS”. Default is “NORMAL” to maintain Stratix behavior.

CLOCK_ENABLE_OUTPUT_B: Clock enable bypass control for output register of port B. Choices are “NORMAL” or “BYPASS”. Default is “NORMAL” to maintain Stratix behavior.

WIDTH_BYTEENA_A: Width of byte enable port for port A. Has to be equal to width of port A / Byte size. Default is 1.

WIDTH_BYTEENA_B: Width of byte enable port for port B. Has to be equal to width of port B / Byte size. Default is 1.

BYTE_SIZE: Indicates whether the byte size used by user is 1, 2, 4, 8, or 9. Default is 8.

READ_DURING_WRITE_MODE_MIXED_PORTS: Indicates the type of behavior when read and write happen at different ports on the same RAM address. One of two options “OLD_DATA” or “DONT_CARE” (default)

RAM_BLOCK_TYPE: Indicates the type of RAM_BLOCK preferred by the user. Can be one of the four options “M512” (SEAB), “M4K” (MEAB), “M-RAM” (MRAM) or “AUTO” (default).

INIT_FILE: Specifies the name of the Initialization file used. It has to be unused when RAM_BLOCK_TYPE=”LARGE”.

INIT_FILE_LAYOUT: Specifies whether the init file has to be used with respect to port a depth X width or with respect to port b depth X width. The default is port A. For simple dual, the megafunction assumes a default of port B and for other modes, it assumes a default of port A unless otherwise specified

MAXIMUM_DEPTH: Specifies the maximum depth to which the RAM should be segmented. Default is Zero, when the megafunction chooses the depth based on ram_block_type or on the other requirements (in case of auto).

3.4.3 Port definitions

wren_a: Write Enable port for port A.

wren_b: Write Enable port for port B. Only available in bidir-dual-port mode.

rden_b: Read Enable port for port B. Only available in simple-dual-port mode.

data_a: Data input for port A

data_b: Data input for port B.

address_a: Address port for port A.

address_b: Address port for port B.

addressstall_a: Address stall input for port A. A '1' input stalls the current address. For Stratix II only.

addressstall_b: Address stall input for port B. A '1' input stalls the current address. For Stratix II only.

clock0, clock1: Clock input ports for the RAM. Clock0 should always be connected.

clocken0, clocken1: Clock enable ports for the clocks.

aclr0, aclr1: Clear ports for the RAM input and output registers.

byteena_a: Byte enable ports for port A. Should be used only when width of the port A input data bus is at least two bytes.

byteena_b: Byte enable ports for port B. Should be used only when width of the port B input data bus is at least two bytes.

q_a: Output data bus for port A

q_b: Output data bus for port B

3.5 parallel_add

Parallel adder megafunction, adds a variable number of inputs to produce a single sum result. In previous version of Quartus, this megafunction was treated as a building block function, with no direct user exposure via the MegaWizard GUI. With the arrival of Stratix II, and that device family's ability to implement 3-to-1 addition within a single level of ALEs, the parallel_add megafunction will get Clearbox support and a MegaWizard plug-in. It will continue to support older device families. For 4-LUT-based devices, it uses a 2-to-1 adder tree implementation, and for p-term device families, it uses 3-to-2 carry-save adder trees and 2-to-1 adders as appropriate.

3.5.1 MegaWizard plug-in

Not yet defined.

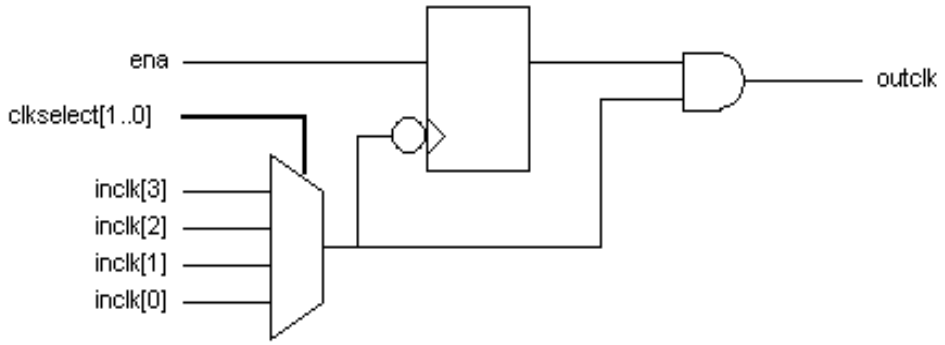
3.5.2 Megafunction Ports and Parameters

```
PARAMETERS
(
        WIDTH,                              -- Width of input data (in bits)
        SIZE,                               -- Number of input numbers
        WIDTHR,                             -- Desired width of result
        SHIFT = 0,                          -- relative shift of data vectors
        REPRESENTATION = "UNSIGNED",        -- SIGNED/UNSIGNED addition
        PIPELINE = 0,                       -- Latency of pipelined adder
        MSW_SUBTRACT = "NO", or YES         -- Most Significant Word is Added or Subtracted
        RESULT_ALIGNMENT = "MSB" or LSB     -- If Widthr is less than optimal result width picks
                                               up the result bits from MSB or LSB
```

```
);
SUBDESIGN parallel_add
(
        data[(SIZE*WIDTH)-1..0]     : INPUT = GND;
        clock                       : INPUT = GND;
        clken                       : INPUT = VCC;
        aclr                        : INPUT = GND;
        result[WIDTHR-1..0]         : OUTPUT;
)
```

3.6 altclkbuf

3.6.1 Behavior Diagram



3.6.2 MegaWizard plug-in

No current wizard exists for altclkbuf, so a new wizard is needed.

3.6.3 Megafunction Ports and Parameters

```
component altclkbuf
        generic (
                clock_type      :       string := "AUTO"
        );
        port(
                clkselect       :       in std_logic_vector(1 downto 0) := (others => '0');
                ena             :       in std_logic := '1';
                inclk           :       in std_logic_vector(3 downto 0) := (others => '0');
                outclk          :       out std_logic
        );
end component;
```

3.6.4 Parameter and Port Definitions

CLOCK_TYPE: Buffer clock type. Legal values are “AUTO,” “GCLK,” “LCLK,” “EXTCLK,” and “SIDE_CLK.”

inclk: clock inputs that drive the clock network

clkselect: input allows dynamically selecting which clock source drives the clock network driven by this clock buffer. The signal selects one of the four clock inputs where a value of ‘00’ selects

inclk[0], ‘01’ selects inclk[1], ‘10’ selects inclk[2], and ‘11’ selects inclk[3]. Defaults to GND if left unconnected. If this signal is connected, then only the Global Clock network can be driven by this clock buffer.

ena: input that allows enabling or disabling the entire clock network driven by this clock buffer. If left unconnected, this signal defaults to VCC.

outclk: clock output that will drive the clock network associated with this clock buffer.

4. Non-Inferable Megafunctions

4.1 altddio_in, altddio_out, altddio_bidir

Double data rate I/O megafunctions. The altddio_out megafunction remains unchanged. Altddio_in and altddio_bidir megafunctions have a new input clock port called “ddioinclk” that is supported only in Stratix II device family.

“ddioinclk” port is optional and if it is connected, it clocks the negative edge of the DDIO input register.

The altddio_in and altddio_bidir megawizards have an extra check box to enable or disable the use of the ddioinclk port. The ddioinclk port has been removed from altddio_bidir and altddio_in megafunctions.

4.2 altdqs

This is the DQ strobe megafunction, intended to be used as a building block for DDR-like I/O interfaces. Changes are not yet known.

4.3 altlvds_rx and altlvds_tx

The receiver side will get significant behavioral changes. Details are not yet finalized. Expect megafunction, behavioral model, and MegaWizard plug-in changes.

4.4 altmemmult

This is the RAM-based coefficient (i.e. constant) multiplier megafunction. This megafunction implements a multi-cycle multiplier by using a ROM or RAM to do partial product look-up, and then a shifting accumulator (altaccumulate) to accumulate the final product. The behavior is very different from lpm_mult, and we believe that it is mainly useful for lower bandwidth DSP applications that need large numbers of moderate bandwidth multipliers.

In Quartus II 3.1, this megafunction will get the ability to load multiplier constants at run time. The number of clock cycles needed to re-load a coefficient varies based on the partial-product look-up table size, but will typically be from 16 cycles to 64 cycles.

New ports and parameters will be added to the megafunction, and the MegaWizard plug-in will also require changes.

See the document *cbx_altmemmult_ffd.doc* for more details.

4.5 altpll

This is the Stratix II and Stratix PLL megafunction. The altclklock megafunction can still be used for very simple PLL needs, since it's generic enough that it supports all device families with PLLs. Some megafunction updates will be needed to support new Stratix II PLL features.

```
component altpll
generic (
    intended_device_family     : string := "Stratix" ;
    operation_mode             : string := "NORMAL" ;
    pll_type                   : string := "AUTO" ;
    qualify_conf_done          : string := "OFF" ;
    compensate_clock           : string := "CLK0" ;
    scan_chain                 : string := "LONG";
    primary_clock              : string := "inclk0" ;
    inclk0_input_frequency     : positive;   -- required parameter
    inclk1_input_frequency     : natural := 0;
    gate_lock_signal           : string := "NO";
    gate_lock_counter          : integer := 0;
    lock_high                  : natural := 1;
      lock_low                 : natural := 5;
    valid_lock_multiplier      : natural := 1;
    invalid_lock_multiplier    : natural := 5;
    switch_over_type           : string := "AUTO";
    switch_over_on_lossclk     : string := "OFF" ;
    switch_over_on_gated_lock  : string := "OFF" ;
    enable_switch_over_counter : string := "OFF";
    switch_over_counter        : natural := 0;
    feedback_source            : string := "EXTCLK0" ;
    bandwidth                  : natural := 0;
    bandwidth_type             : string := "UNUSED";
    spread_frequency           : natural := 0;
    down_spread                : string := "0.0";
    -- simulation-only parameters
    skip_vco                   : string := "off";

    -- internal clock specifications
    clk5_multiply_by           : positive := 1;
    clk4_multiply_by           : positive := 1;
```

```
    clk3_multiply_by          : positive := 1;
    clk2_multiply_by          : positive := 1;
    clk1_multiply_by          : positive := 1;
    clk0_multiply_by          : positive := 1;
    clk5_divide_by            : positive := 1;
    clk4_divide_by            : positive := 1;
    clk3_divide_by            : positive := 1;
    clk2_divide_by            : positive := 1;
    clk1_divide_by            : positive := 1;
    clk0_divide_by            : positive := 1;
    clk5_phase_shift          : string := "0";
    clk4_phase_shift          : string := "0";
    clk3_phase_shift          : string := "0";
    clk2_phase_shift          : string := "0";
    clk1_phase_shift          : string := "0";
    clk0_phase_shift          : string := "0";
    clk5_time_delay           : string := "0";
    clk4_time_delay           : string := "0";
    clk3_time_delay           : string := "0";
    clk2_time_delay           : string := "0";
    clk1_time_delay           : string := "0";
    clk0_time_delay           : string := "0";
    clk5_duty_cycle           : natural := 50;
    clk4_duty_cycle           : natural := 50;
    clk3_duty_cycle           : natural := 50;
    clk2_duty_cycle           : natural := 50;
    clk1_duty_cycle           : natural := 50;
    clk0_duty_cycle           : natural := 50;
    -- external clock specifications
    extclk3_multiply_by       : positive := 1;
    extclk2_multiply_by       : positive := 1;
    extclk1_multiply_by       : positive := 1;
    extclk0_multiply_by       : positive := 1;
    extclk3_divide_by         : positive := 1;
    extclk2_divide_by         : positive := 1;
    extclk1_divide_by         : positive := 1;
    extclk0_divide_by         : positive := 1;
```

```
        extclk3_phase_shift       : string := "0";
        extclk2_phase_shift       : string := "0";
        extclk1_phase_shift       : string := "0";
        extclk0_phase_shift       : string := "0";
        extclk3_time_delay        : string := "0";
        extclk2_time_delay        : string := "0";
        extclk1_time_delay        : string := "0";
        extclk0_time_delay        : string := "0";
        extclk3_duty_cycle        : natural := 50;
        extclk2_duty_cycle        : natural := 50;
        extclk1_duty_cycle        : natural := 50;
        extclk0_duty_cycle        : natural := 50;


        -- advanced user parameters
        vco_min                   : natural := 0;
        vco_max                   : natural := 0;
        vco_center                : natural := 0;
        pfd_min                   : natural := 0;
        pfd_max                   : natural := 0;
        m_initial                 : natural := 1;
        m                         : natural := 0; -- m must default to 0 to force altpll to calculate the internal
parameters for itself
        n                         : natu
        m2                        : natural := 1;
        n2                        : natural := 1;
        ss                        : natural := 0;
        c0_high                   : natural := 1;
        c1_high                   : natural := 1;
        c2_high                   : natural := 1;
        c3_high                   : natural := 1;
        c4_high                   : natural := 1;
        c5_high                   : natural := 1;
        l0_high                   : natural := 1;
        l1_high                   : natural := 1;
        g0_high                   : natural := 1;
        g1_high                   : natural := 1;
        g2_high                   : natural := 1;
```

```
    g3_high              : natural := 1;
    e0_high              : natural := 1;
    e1_high              : natural := 1;
    e2_high              : natural := 1;
    e3_high              : natural := 1;
    c0_low               : natural := 1;
    c1_low               : natural := 1;
    c2_low               : natural := 1;
    c3_low               : natural := 1;
    c4_low               : natural := 1;
    c5_low               : natural := 1;
    l0_low               : natural := 1;
    l1_low               : natural := 1;
    g0_low               : natural := 1;
    g1_low               : natural := 1;
    g2_low               : natural := 1;
    g3_low               : natural := 1;
    e0_low               : natural := 1;
    e1_low               : natural := 1;
    e2_low               : natural := 1;
    e3_low               : natural := 1;
    c0_initial           : natural := 1;
    c1_initial           : natural := 1;
    c2_initial           : natural := 1;
    c3_initial           : natural := 1;
    c4_initial           : natural := 1;
    c5_initial           : natural := 1;
    l0_initial           : natural := 1;
    l1_initial           : natural := 1;
    g0_initial           : natural := 1;
    g1_initial           : natural := 1;
    g2_initial           : natural := 1;
    g3_initial           : natural := 1;
    e0_initial           : natural := 1;
    e1_initial           : natural := 1;
    e2_initial           : natural := 1;
    e3_initial           : natural := 1;
```

```
    c0_mode               : string := "bypass" ;
    c1_mode               : string := "bypass" ;
    c2_mode               : string := "bypass" ;
    c3_mode               : string := "bypass" ;
    c4_mode               : string := "bypass" ;
    c5_mode               : string := "bypass" ;
    l0_mode               : string := "bypass" ;
    l1_mode               : string := "bypass" ;
    g0_mode               : string := "bypass" ;
    g1_mode               : string := "bypass" ;
    g2_mode               : string := "bypass" ;
    g3_mode               : string := "bypass" ;
    e0_mode               : string := "bypass" ;
    e1_mode               : string := "bypass" ;
    e2_mode               : string := "bypass" ;
    e3_mode               : string := "bypass" ;
    c0_ph                 : natural := 0;
    c1_ph                 : natural := 0;
    c2_ph                 : natural := 0;
    c3_ph                 : natural := 0;
    c4_ph                 : natural := 0;
    c5_ph                 : natural := 0;
    l0_ph                 : natural := 0;
    l1_ph                 : natural := 0;
    g0_ph                 : natural := 0;
    g1_ph                 : natural := 0;
    g2_ph                 : natural := 0;
    g3_ph                 : natural := 0;
    e0_ph                 : natural := 0;
    e1_ph                 : natural := 0;
    e2_ph                 : natural := 0;
    e3_ph                 : natural := 0;
    m_ph                  : natural := 0;
    l0_time_delay         : natural := 0;
    l1_time_delay         : natural := 0;
    g0_time_delay         : natural := 0;
    g1_time_delay         : natural := 0;
```

```
g2_time_delay            : natural := 0;
g3_time_delay            : natural := 0;
e0_time_delay            : natural := 0;
e1_time_delay            : natural := 0;
e2_time_delay            : natural := 0;
e3_time_delay            : natural := 0;
m_time_delay             : natural := 0;
n_time_delay             : natural := 0;
c1_use_casc_in           : string := "off";
c2_use_casc_in           : string := "off";
c3_use_casc_in           : string := "off";
c4_use_casc_in           : string := "off";
c5_use_casc_in           : string := "off";
extclk3_counter          : string := "e3" ;
extclk2_counter          : string := "e2" ;
extclk1_counter          : string := "e1" ;
extclk0_counter          : string := "e0" ;
clk5_counter             : string := "l1" ;
clk4_counter             : string := "l0" ;
clk3_counter             : string := "g3" ;
clk2_counter             : string := "g2" ;
clk1_counter             : string := "g1" ;
clk0_counter             : string := "g0" ;
enable0_counter          : string := "l0";
enable1_counter          : string := "l0";
charge_pump_current      : natural := 2;
loop_filter_r            : string := "1.0";
loop_filter_c            : natural := 5;
sclkout0_phase_shift     : natural := 0;
 sclkout1_phase_shift    : naturanl := 0;
clk2_output_frequency    : natural := 0;
clk1_output_frequency    : natural := 0;
clk0_output_frequency    : natural := 0;
 port_clkena0            : string := “port_connectivity”;
port_clkena1             : string := “port_connectivity”;
port_clkena2             : string := “port_connectivity”;
port_clkena3             : string := “port_connectivity”;
```

```
        port_clkena4                    : string := “port_connectivity”;
        port_clkena5                    : string := “port_connectivity”;
        port_extclkena0                 : string := “port_connectivity”;
        port_extclkena1                 : string := “port_connectivity”;
        port_extclkena2                 : string := “port_connectivity”;
        port _clkbad0                   : string := “port_connectivity”;
        port_clkbad1                    : string := “port_connectivity”;
        port_activeclock                : string := “port_connectivity”;
        port_clkloss                    : string := “port_connectivity”;
        vco_multiply_by                 :
        lpm_type                : string := "altpll"
);
port (
        inclk       : in std_logic_vector(1 downto 0) := (others => '0');
        fbin        : in std_logic := '1';
        pllena      : in std_logic := '1';
        clkswitch   : in std_logic := '0';
        areset      : in std_logic := '0';
        pfdena      : in std_logic := '1';
        clkena      : in std_logic_vector(5 downto 0) := (others => '1');
        extclkena   : in std_logic_vector(3 downto 0) := (others => '1');
        scanclk     : in std_logic := '0';
        scanaclr    : in std_logic := '0';
        scanread    : in std_logic := '0';
        scanwrite   : in std_logic := '0';
        scandata    : in std_logic := '0';

        clk         : out std_logic_vector(5 downto 0);
        extclk      : out std_logic_vector(3 downto 0);
        clkbad      : out std_logic_vector(1 downto 0);
        activeclock : out std_logic;
        clkloss     : out std_logic;
        locked      : out std_logic;
        scandataout : out std_logic;
        scandone    : out std_logic;
        sclkcout1   : out std_logic;
        sclkout0 : out std_logic;
```

```
    enable1 : out std_logic;
    enable0 : out std_logic;
);
end component;
```

MegaWizard plug-in

We will use the existing altpll MegaWizard plug-in. Major updates to this plug-in will be needed.

Megafunction Ports and Parameters

See the Stratix altpll for the base port and parameter list.

Please see the Stratix II PLL WYSIWYG document for details and explanations.

4.6 altpll_reconfig

This megafunction allows Stratix II and Stratix PLLs to be dynamically reconfigured. Some megafunction and wizard updates will be needed to support new Stratix II PLL features.

4.7 dcfifo

The dual-clock FIFO megafunction takes advantage of the selectable clock enable on Stratix II RAM blocks. The clock enable bypass feature is used on the read address register to allow the RAM to read internally on every clock cycle. In legacy mode (non-showahead), the clock enable can then be used to preserve valid data in the Q output register. This results in a dramatic LC savings compared to Stratix dual-clock FIFOs, because the LE-based, 3-deep output FIFO can be omitted. It also results in much lower internal latency, and a simpler, cleaner design.

The reduced latency affects the behavior in legacy mode, and addresses a frequent customer complaint with Stratix FIFOs. Show-ahead mode behavior is identical to Stratix “area” show-shead mode. In Stratix II, there is no such thing as “speed” mode, which exists only in Stratix and Cyclone because of the awkward RAM architecture of the Stratix family.

Another difference compared to Stratix dcfifo is in the read input path. The Stratix II implementation takes advantage of the new address stall RAM feature. This simplifies the read input path, and helps boost performance, but it has no impact on behavior.

The dcfifo write path in Stratix II is identical to Stratix. See the schematic below for additional details.

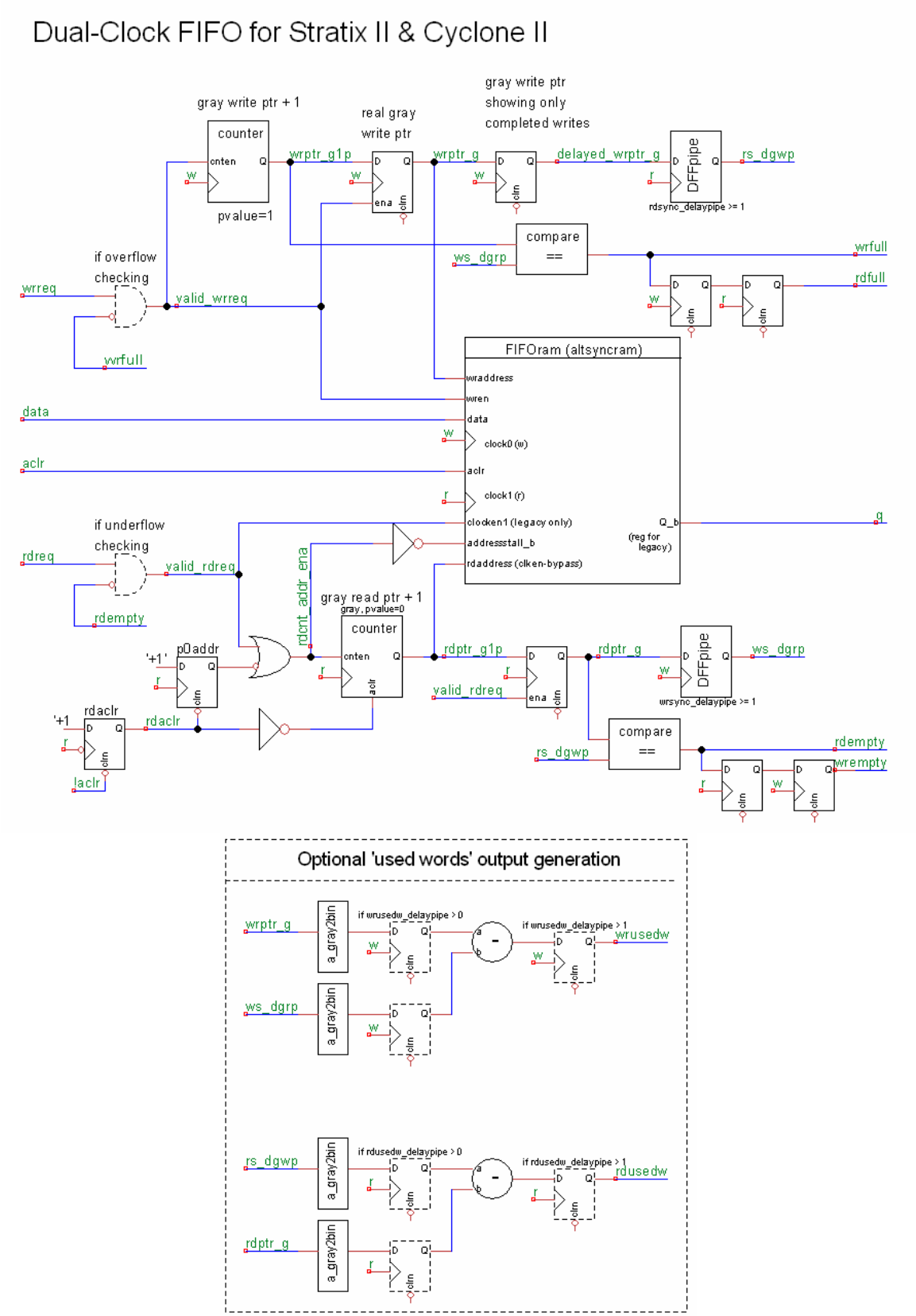


Figure 4-1 dcfifo schematic for Stratix II