LUTRAM WYSIWYG Description for Stratix III

Version 2.0

March 9, 2009

by

Altera Corporation

Table of Contents:

1. Overview

This document describes the WYSIWYG primitive for Stratix III LUTRAM. LUTRAM is new in Stratix III. LUTRAM is built on top of the regular Stratix III LAB, with some additional hardware, these LABs can support RAM functionalities. Half of the LABs in Stratix III have the LUTRAM capability. These LABs are called MLABs.

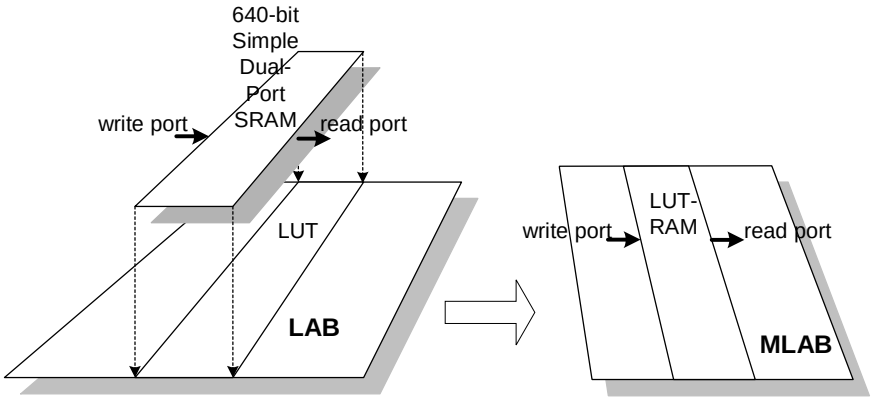

Figure 1. Conceptual view of MLAB

Due to the differences between LUTRAM and the Stratix III block RAM, we will use a different WYSIWYG primitive to model LUTRAM.

This document is organized as follows. In Section 2, a functional diagram of the LUTRAM is shown. In Section 3, the LUTRAM primitive is defined.

2. Stratix III LUTRAM block

The following picture shows the abstract functional I/O interface for Stratix III LUTRAM. Note that it only models the memory core portion of the RAM, i.e. input and output registers are not modeled as part of the LUTRAM WYSIWYG.

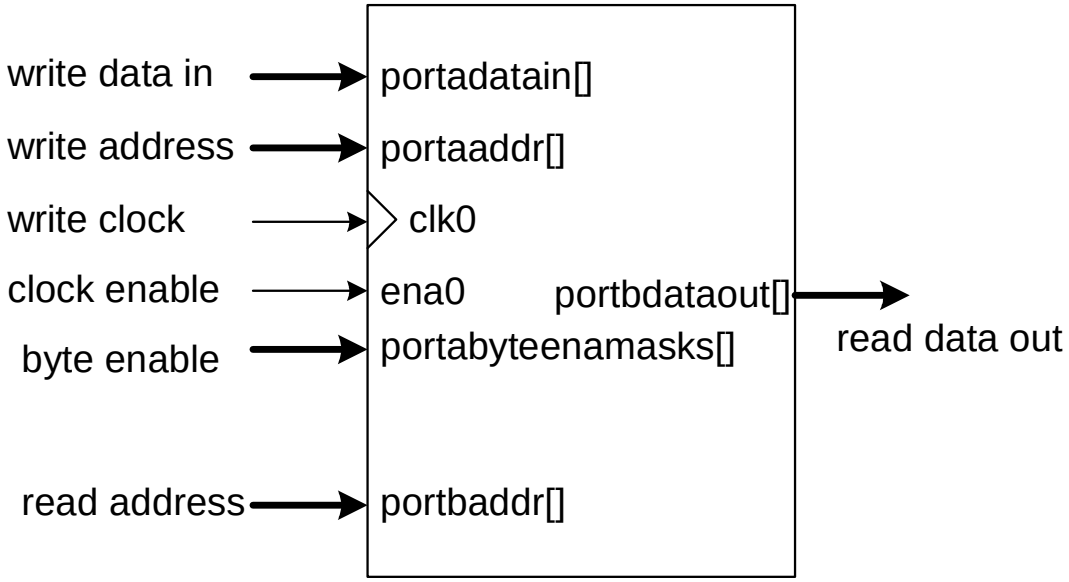

Figure 2. LUTRAM WYSIWYG diagram

3. LUTRAM Primitive

The LUTRAM WYSIWYG primitive is used to model RAM blocks implemented in MLABs.

```
Stratix III_lutram <block_name>
(
      // Write Port inputs
      .portadatain(<write data source bus>),
      .portaaddr(<write addresses bus>),
      .portabyteenamasks(<byte-enable mask source bus>),

      // Read Port inputs
      .portbaddr(<read addresses bus>),

      // Control signals
      .clk0(<clock source>),
      .ena0(<clock enable for clock>),

      // Read Port outputs
      .portbdataout(<data output bus>)
);
defparam <block_name>.logical_ram_name = <logical RAM's name>;
defparam <block_name>.init_file = <name of the initialization
      file>;
defparam <block_name>.logical_ram_depth = <depth of the logical
      RAM >;
defparam <block_name>.logical_ram_width = <width of the logical
      RAM >;
```

```
defparam <block_name>.mixed_port_feed_through_mode = <mixed port
      feed through mode>;

defparam <block_name>.first_address = <starting address for this
      block>;
defparam <block_name>.last_address = <ending address for this
      block>;
defparam <block_name>.first_bit_number = <first logical bit
      position of this block>;
defparam <block_name>.data_width = <width of the data bus of this
      block>;
defparam <block_name>.address_width = <width of the write/read
      address bus of this block>;
defparam <block_name>.byte_enable_mask_width = <width of the
      byte-enable mask bus of this block>;
defparam <block_name>.byte_size = <byte size>;
```

3.1 RAM Input Signals

<*block_name*> is the unique identifier for this particular block. *This field is required.*

Natively, LUTRAM only supports simple dual port mode. For a 'single-port' mode user memory, one needs to connect the address bus to both the waddr and the raddr ports of LUTRAM. For a 'rom' mode user memory, one needs to route the address bus to the raddr port, and tie off the clock enable or byte enable ports. **Note that in the case of 'rom' mode, the user loses the ability to do error detection on the content. It's recommended that random logic be implemented using Stratix III combinational logic cells instead**.

'true dual port' mode user memory is not supported in LUTRAM.

.portadatain(<*data sources*>**),** is the write data input bus to this RAM block. The data input bus should be registered by the same clock signal. *This port is required if this LUTRAM is implementing a non-ROM mode user memory*.

.portaaddr(<*write addresses*>**),** are the address inputs for the write port. Just like the data input bus, the write address bus is always registered by the same clock signal. *This port is required if this LUTRAM is implementing a non-ROM mode user memory*.

.portabyteenamasks(<*byte masks*>**),** are the byte enable masks for the write port. This byte-enable mask port is always registered by the same clock signal. *This port is optional*. The byte-enable mask bus width should be equal to the data bus width divided by byte size.

.portbaddr(<*read addresses*>**),** are the address inputs for reading. **It can be registered or combinational**. The valid address bus width will depend on the operation mode and the block type. *This port is required*.

.clk0(<*clock source*>**),** designates the clocking source for the write operation. All write related inputs (datain, waddr, byteenamasks) should be registers that are clocked by the same source. *This signal should be the same signal for all blocks of the same logical RAM. This port is required if this LUTRAM is implementing a non-ROM mode user memory.*

.ena0(<*clock-enable*>**),** is the clock enable for .clk0. This serves as the write enable for the LUTRAM. *This port is optional*. *This signal should be the same signal for all blocks of the same logical RAM*.

3.2 RAM Output Signals

.portbdataout(<*data outputs*>**),** is the read data output bus of this LUTRAM block. The output can be registered externally if needed.

3.3 RAM Parameters

There are two types of information fields: fields which give logical RAM-wide information, and fields that are block-specific.

For LUTRAM, there is no "operation mode" parameter since it only natively supports simple dual port operation.

Logical RAM-wide information fields are as follows:

<*logical RAM's name*>*,* is the unique identifier for the corresponding logical RAM. *This field is required. It should be the same name for all blocks of the same logical RAM.*

<*name of the initialization file*>, is an identifier for the memory initialization file (.mif or .hex). *This field is optional (memory is not initialized). It should be the same file for all blocks of the same logical RAM.*

<*depth of the logical RAM* >, represents the logical depth of the corresponding logical RAM. *This field is required. It should be the same value for all blocks of the same logical RAM.*

< *width of the logical RAM*>, represents the logical width of the corresponding logical RAM. *This field is required. It should be the same value for all blocks of the same logical RAM.*

<*mixed port feed through mode*> is one of *{ dont_care, old, new }*. *This field is optional.* This field is used to dictate the behavior of 'read-during-write on different ports' at the same location with the same clock. When it's 'dont_care', it means the read output is 'unknown'. When it's 'old', it means the read output is the old data in the address before the write occurs. When it's 'new', it means the read output is the data in the address after the write occurs. The default value is 'dont_care'. **Note that natively, old data will appear at the data out port of LUTRAM before the falling edge of the write clock. After certain delay, new data will appear. This field should be left set as don't-care unless surrounding logic can guarantee the new or old data behavior for the whole cycle**.

The following information fields are block-specific:

<*starting address for this block*>, represents the starting address of this particular block. *This field is required.*

< *ending address for this block*>*,* represents the port A ending address of this particular block. *This field is required.*

< *first logical bit position of this block*> gives the first writing bit position of the data in bus of this particular block within the corresponding logical RAM. *This field is required.*

<*width of the data bus of this block*> gives the width of the port A data in/out bus of this particular block within the corresponding logical RAM. **This parameter is used for simulation only and legality check will assert both the data in bus and the data out bus are single-bit during the compilation flow**. *This field is required.*

<*width of the address bus of this block*> gives the width of the write/read address bus of this particular block within the corresponding logical RAM. It should not exceed 6 since the maximum depth supported by MLAB is 64. *This field is required.*

<*width of the byte-enable mask bus of this block*> gives the width of the byte-enable mask bus of this particular block within the corresponding logical RAM. *This field is required.*

<*byte size*> describes the byte size (the number of data bits each byte-enable mask controls). It should not exceed 10.

3.4 Registering Modes of I/O signals

LUTRAM should always be driven by registers for the write related ports. These registers will be external to the LUTRAM primitive. The read address and data out ports, however, can be combinational.

3.5 Polarities and Default Values

All signals are active high and all secondary inputs have programmable inversions.

Upon power-up, all LUTRAM outputs will have values based on the read address inputs and preloaded content of the LUTRAM so it is unknown. If the LUTRAM is not preloaded, then the output will be 0.