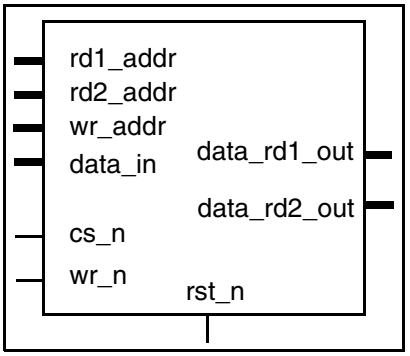
DW_ram_2r_w_a_lat

Write-Port, Dual-Read-Port RAM (Latch-Based)

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word depth
- Parameterized data width
- Asynchronous static memory
- Parameterized reset implementation



Description

DW_ram_2r_w_a_lat implements a parameterized, asynchronous, three-port static RAM.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rd1_addr	ceil($\log_2$[*depth*]) bit	Input	Read1 address bus
rd2_addr	ceil($\log_2$[*depth*]) bit	Input	Read2 address bus
wr_addr	ceil($\log_2$[*depth*]) bit	Input	Write address bus
data_in	*data_width* bit	Input	Input data bus
data_rd1_out	*data_width* bit	Output	Output data bus for read1
data_rd2_out	*data_width* bit	Output	Output data bus for read2

Table 1-2 Parameter Description

Parameter	Values	Description
data_width	1 to 256 Default = none	Width of `data_in` and `data_out` buses
depth	2 to 256 Default = none	Number of words in the memory array (address width)
rst_mode	0 or 1 Default = 1	Determines if the rst_n input is used. 0 = `rst_n` initializes the RAM, 1 = `rst_n` is not connected

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW06.DW_RAM_R_W_A_LAT_CFG_SIM	VHDL simulation configuration
dw/dw06/src/DW_ram_2r_w_a_lat_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_ram_2r_w_a_lat.v	Verilog simulation model source code

The write data enters the RAM through the `data_in` input port, and is read out through either the `data_rd1_out` port or the `data_rd2_out`. The RAM is constantly reading regardless of the state of `cs_n`.

The `rd1_addr` port, `rd2_addr`, and `wr_addr` ports are used to address the *depth* words in memory. If `rd1_addr` or `rd2_addr` contains a value beyond the maximum depth, then that output port is driven LOW. For example, if `rd1_addr` = 7 hex and `depth` = 6), then the `data_rd1_out` is driven LOW. If `rd2_addr` is beyond the maximum depth, then `data_rd2_out` is driven low.

For `wr_addr` beyond the maximum depth, nothing occurs and the data is lost. No warnings are given during simulations when an address beyond the scope of *depth* is used.

Attention This component contains enable signals for internal latches that are derived from the `wr_n` port. To keep hold times to a minimum, you should consider instances of this component to be individual floorplanning elements.

Chip Selection, Reading and Writing

The cs_n input is the chip select, active low signal, which enables data to be written to the RAM. The RAM is constantly reading, regardless of the state of cs_n.

When cs_n and wr_n (write enable, active low) are both LOW, data_in is transparent to the internal memory cell being accessed (data_in = output data of the memory cell). Therefore, during the period when wr_n and cs_n are low, a change in data in is reflected on the output of the internal memory cell being accessed. If rd1_addr port or rd2_addr port are the same value as wr_addr, data_in equals data_rd1_out or data_rd2_out while wr_n is LOW. Data is captured into the memory cell on the low-to-high transition of wr_n.

When cs_n is HIGH, writing to the RAM is disabled.

Reset

rst_n
This signal is an active-low input that initializes the RAM to zeros if the rst_mode parameter is set to 0, independent of the value of cs_n. If the rst_mode parameter is set to 1, rst_n does not affect the RAM, and should be tied HIGH or LOW. Synthesis optimizes the design, and does not use the rst_n signal.

Attention	If the technology library being used does not contain an active low D-latch with clear, synthesis gates the inputs of a D-latch with the rst_n signal, increasing the area of the design.

Application Notes

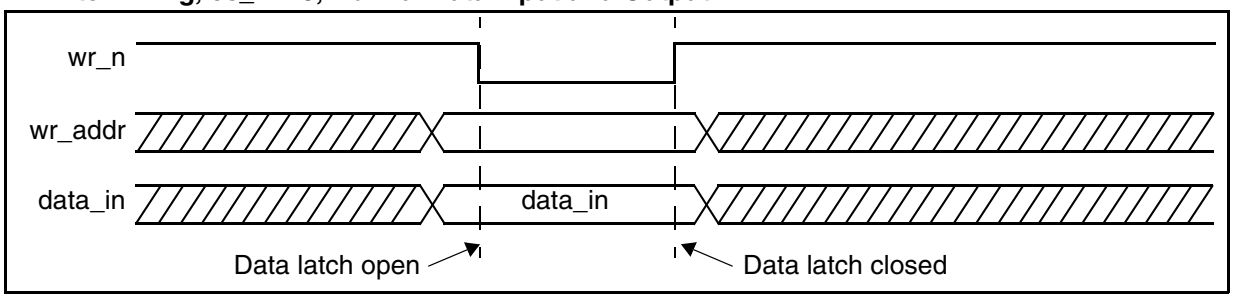
DW_ram_2r_w_a_lat is intended to be used as small scratch-pad memory or register file. Because DW_ram_2r_w_a_lat is built from the cells within the ASIC cell library, it should be kept small to obtain an efficient implementation. If a larger memory is required, you should consider using a hard macro RAM from the ASIC library in use.

Timing Waveforms

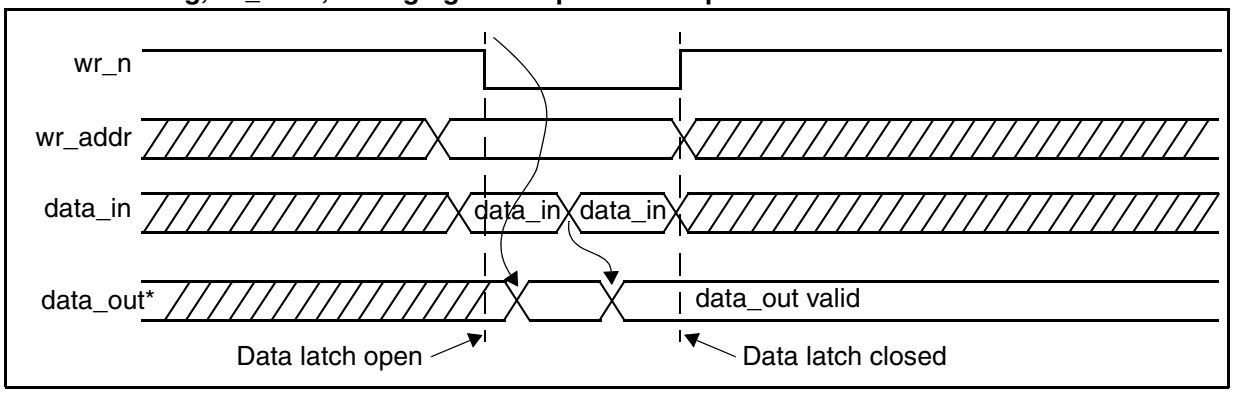
The figures in this section show timing diagrams for various conditions of DW_ram_2r_w_a_lat.

Figure 1-1 Instantiated RAM Timing Waveforms

Write Timing, cs_n = 0, Normal Data Input and Output

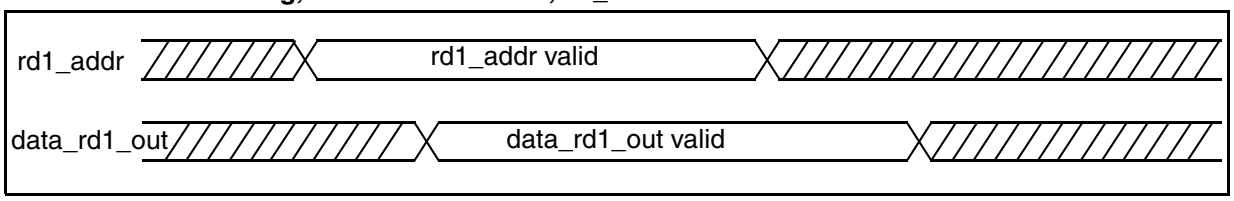


Write Timing, cs_n = 0, Changing Data Input and Output



* Write through is seen only when rd_addr = wr_addr.

Read Port 1 Timing, address controlled, cs_n = don't care



Read Port 2 Timing, address controlled, cs_n = don't care

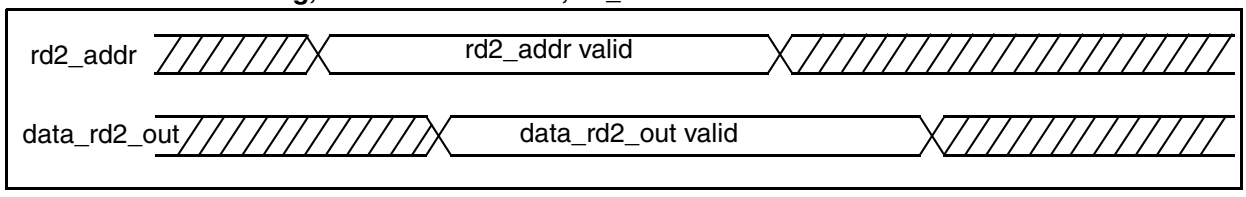
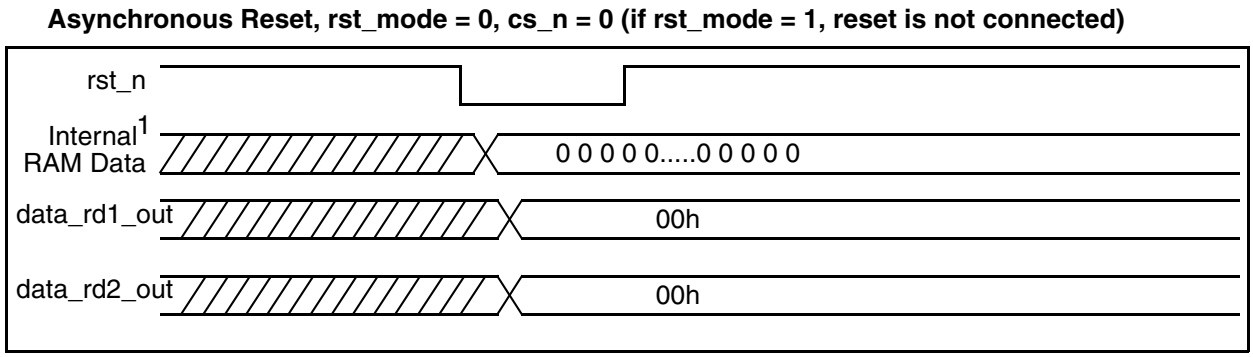


Figure 1-2 RAM Reset Timing Waveforms



[1] Internal RAM Data is the array of memory bits; the memory is not available to users.

Related Topics

- Memory – Synchronous RAMs Listing
- DesignWare Building Block IP Documentation Overview

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_foundation_comp.all;

entity DW_ram_2r_w_a_lat_inst is
  generic (inst_data_width : INTEGER := 8;
           inst_depth      : INTEGER := 8;
           inst_rst_mode   : INTEGER := 1 );
  port (inst_rst_n   : in std_logic;
        inst_cs_n    : in std_logic;
        inst_wr_n    : in std_logic;
        inst_rd1_addr: in std_logic_vector(bit_width(inst_depth)-1 downto 0);
        inst_rd2_addr: in std_logic_vector(bit_width(inst_depth)-1 downto 0);
        inst_wr_addr : in std_logic_vector(bit_width(inst_depth)-1 downto 0);
        inst_data_in : in std_logic_vector(inst_data_width-1 downto 0);
        data_rd1_out_inst : out std_logic_vector(inst_data_width-1 downto 0);
        data_rd2_out_inst : out std_logic_vector(inst_data_width-1 downto 0)
       );
end DW_ram_2r_w_a_lat_inst;

architecture inst of DW_ram_2r_w_a_lat_inst is
begin

  -- Instance of DW_ram_2r_w_a_lat
  U1 : DW_ram_2r_w_a_lat
    generic map (data_width => inst_data_width,   depth => inst_depth,
                 rst_mode => inst_rst_mode )
    port map (rst_n => inst_rst_n,   cs_n => inst_cs_n,   wr_n => inst_wr_n,
              rd1_addr => inst_rd1_addr,   rd2_addr => inst_rd2_addr,
              wr_addr => inst_wr_addr,   data_in => inst_data_in,
              data_rd1_out => data_rd1_out_inst,
              data_rd2_out => data_rd2_out_inst );
end inst;

-- pragma translate_off
configuration DW_ram_2r_w_a_lat_inst_cfg_inst of DW_ram_2r_w_a_lat_inst is
  for inst
  end for; -- inst
end DW_ram_2r_w_a_lat_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_ram_2r_w_a_lat_inst( inst_rst_n, inst_cs_n, inst_wr_n,
                inst_rd1_addr, inst_rd2_addr, inst_wr_addr, inst_data_in,
                data_rd1_out_inst, data_rd2_out_inst );
  parameter data_width = 8;
  parameter depth = 8;
  parameter rst_mode = 1;
  `define bit_width_depth 3 // ceil(log2(depth))

  input inst_rst_n;
  input inst_cs_n;
  input inst_wr_n;
  input [`bit_width_depth-1 : 0] inst_rd1_addr;
  input [`bit_width_depth-1 : 0] inst_rd2_addr;
  input [`bit_width_depth-1 : 0] inst_wr_addr;
  input [data_width-1 : 0] inst_data_in;
  output [data_width-1 : 0] data_rd1_out_inst;
  output [data_width-1 : 0] data_rd2_out_inst;

  // Instance of DW_ram_2r_w_a_lat
  DW_ram_2r_w_a_lat #(data_width,   depth,   rst_mode)
    U1 (.rst_n(inst_rst_n),   .cs_n(inst_cs_n),   .wr_n(inst_wr_n),
        .rd1_addr(inst_rd1_addr),   .rd2_addr(inst_rd2_addr),
        .wr_addr(inst_wr_addr),   .data_in(inst_data_in),
        .data_rd1_out(data_rd1_out_inst),
        .data_rd2_out(data_rd2_out_inst) );
endmodule
```

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043

www.synopsys.com