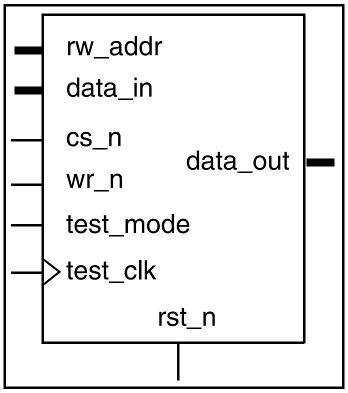
DW_ram_rw_a_dff

Asynchronous Single-Port RAM (Flip-Flop-Based)

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word depth
- Parameterized data width
- Asynchronous static memory
- Parameterized reset implementation
- High testability using DFT Compiler



Description

DW_ram_rw_a_dff implements a parameterized, asynchronous, single-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
test_mode	1 bit	Input	Enables test_clk
test_clk	1 bit	Input	Test clock to capture data during test_mode
rw_addr	ceil($\log_2$[*depth*]) bit(s)	Input	Address bus
data_in	*data_width* bit(s)	Input	Input data bus
data_out	*data_width* bit(s)	Output	Output data bus

Table 1-2 Parameter Description

Parameter	Values	Description
data_width	1 to 256	Width of `data_in` and `data_out` buses. Default = none
depth	2 to 256	Number of words in the memory array (address width). Default = none
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
rtl[a]	Synthesis model	DesignWare

[a] The implementation, “rtl,” replaces the obsolete implementation, “str.” Existing designs that specify the obsolete implementation (“str”) will automatically have that implementation replaced by the new superseding implementation (“rtl”) as will be noted by an information message (SYNDB-36) generated during DC compilation.

Table 1-4 Simulation Models

Model	Function
DW06.DW_RAM_RW_A_DFF_CFG_SIM	VHDL simulation configuration
dw/dw06/src/DW_ram_rw_a_dff_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_ram_rw_a_dff.v	Verilog simulation model source code

The write data enters the RAM through the `data_in` input port and is read out at the `data_out` port. The RAM is constantly reading regardless of the state of `cs_n`.

The `rw_addr` port is used to address the *depth* words in memory. For addresses beyond the maximum depth (for example, `rw_addr` = 7 and `depth` = 6), the `data_out` bus is driven LOW. No warnings are given during simulations when an address beyond the scope of *depth* is used.

Attention This component contains clock signals for internal flip-flops that are derived from the `wr_n` and `test_clk` ports. To keep hold times and internal clock skews 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 that enables data to be written to the RAM. When `cs_n` is LOW, the RAM is enabled to be read, and data is written to the RAM when there is a low-to-high transition of `wr_n`, write enable.

When `cs_n` is HIGH, the RAM is disabled, and the `data_out` bus is driven LOW.

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. In this case, synthesis optimizes the design, and does not use the `rst_n` signal.

Making the RAM Scannable

DW_ram_rw_a_dff may be made scannable using DFT Compiler. Use the `set_test_hold 1 test_mode` command before `insert_scan`.

The `test_mode` signal, when active (HIGH), selects the `test_clk` port to control the capture of data into the RAM. The `test_mode` signal may be tied LOW if a scannable design is not required. When `test_mode` is tied LOW, synthesis optimizes the design, and does not connect the `test_mode` and `test_clk` signals.

Attention

For scannable designs, the `test_mode` signal should only be active during scan shifting (i.e., when scan enable is active). When `test_mode` is active, all RAM addresses are written with the `data_in` value at the rising edge of `test_clk`. When `test_mode` and scan enable are both active, the data currently in the RAM is shifted out for viewing the state of the RAM.

Application Notes

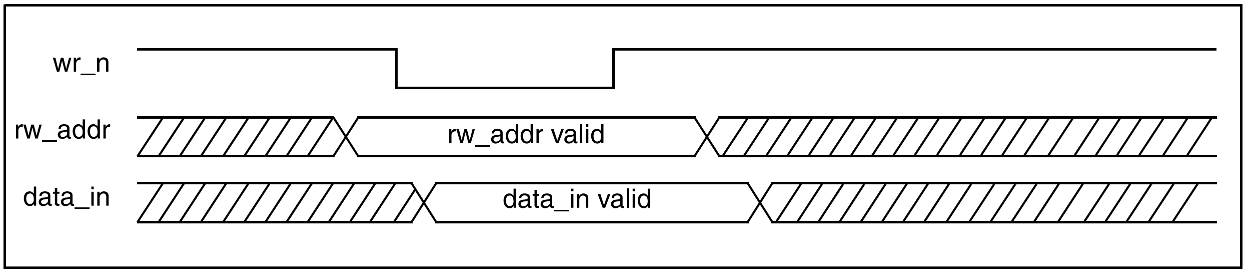
DW_ram_rw_a_dff is intended to be used as a small scratch-pad memory or register file. Because DW_ram_rw_a_dff is built from the cells within the ASIC cell library, it should be kept small to obtain an efficient implementation. If a larger memory is required, you should consider using a hard macro RAM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

Timing Waveforms

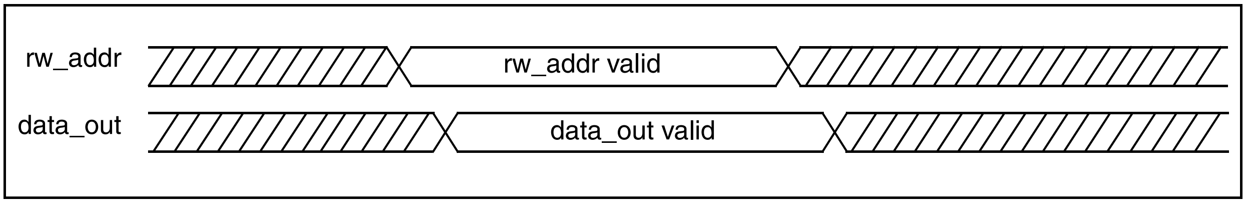
The figures in this section show timing diagrams for various conditions of DW_ram_rw_a_dff.

Figure 1-1 Instantiated RAM Timing Waveforms

Write Timing, wr_n controlled, cs_n = 0; address valid before wr_n transition to low


wr_n
rw_addr
rw_addr valid
data_in
data_in valid

Read Port Timing, address controlled, rst_n = 1, cs_n = 1


rw_addr
rw_addr valid
data_out
data_out valid

Read Port Timing, csn controlled

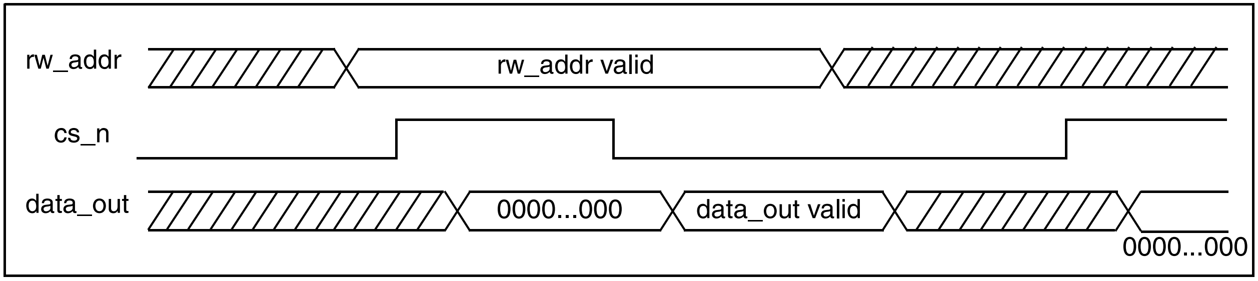
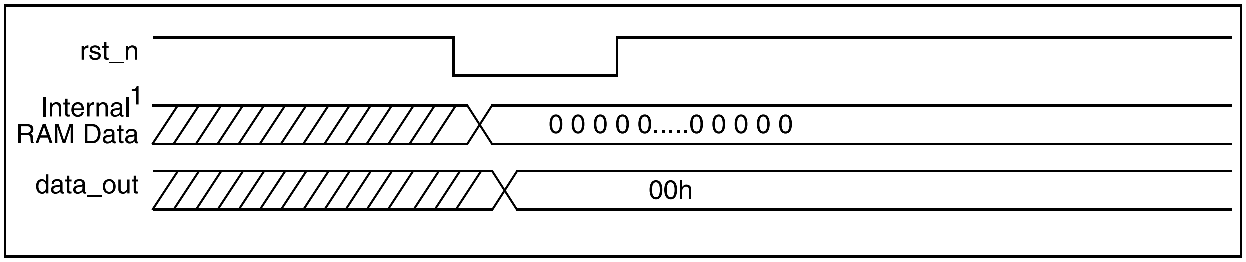

rw_addr
rw_addr valid
cs_n
data_out
0000...000
data_out valid
0000...000

Figure 1-2 RAM Reset Timing Waveforms

Asynchronous Reset, rst_mode = 0, cs_n = 0 (if rst_mode = 1, reset is not connected)


rst_n
Internal[1] RAM Data
0 0 0 0 0.....0 0 0 0 0
data_out
00h

[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_rw_a_dff_inst is
  generic (inst_data_width : INTEGER := 8;
           inst_depth      : INTEGER := 8;
           inst_rst_mode   : INTEGER := 0 );
  port (inst_rst_n     : in std_logic;   inst_cs_n      : in std_logic;
        inst_wr_n      : in std_logic;   inst_test_mode : in std_logic;
        inst_test_clk  : in std_logic;
        inst_rw_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_out_inst: out std_logic_vector(inst_data_width-1 downto 0)  );
end DW_ram_rw_a_dff_inst;

architecture inst of DW_ram_rw_a_dff_inst is
begin

  -- Instance of DW_ram_rw_a_dff
  U1 : DW_ram_rw_a_dff
    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,
              test_mode => inst_test_mode,   test_clk => inst_test_clk,
              rw_addr =>   inst_rw_addr,   data_in => inst_data_in,
              data_out => data_out_inst );
end inst;

-- pragma translate_off
configuration DW_ram_rw_a_dff_inst_cfg_inst of DW_ram_rw_a_dff_inst is
  for inst
  end for; -- inst
end DW_ram_rw_a_dff_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_ram_rw_a_dff_inst( inst_rst_n, inst_cs_n, inst_wr_n,
        inst_test_mode, inst_test_clk, inst_rw_addr, inst_data_in,
        data_out_inst );

  parameter data_width = 8;
  parameter depth = 8;
  parameter rst_mode = 0;
  `define bit_width_depth 3 // ceil(log2(depth))

  input inst_rst_n;
  input inst_cs_n;
  input inst_wr_n;
  input inst_test_mode;
  input inst_test_clk;
  input [`bit_width_depth-1 : 0] inst_rw_addr;
  input [data_width-1 : 0] inst_data_in;
  output [data_width-1 : 0] data_out_inst;

  // Instance of DW_ram_rw_a_dff
  DW_ram_rw_a_dff #(data_width, depth, rst_mode)
    U1 (.rst_n(inst_rst_n),   .cs_n(inst_cs_n),   .wr_n(inst_wr_n),
        .test_mode(inst_test_mode),   .test_clk(inst_test_clk),
        .rw_addr(inst_rw_addr),   .data_in(inst_data_in),
         .data_out(data_out_inst) );
endmodule
```

Copyright Notice and Proprietary Information

© 2018 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

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