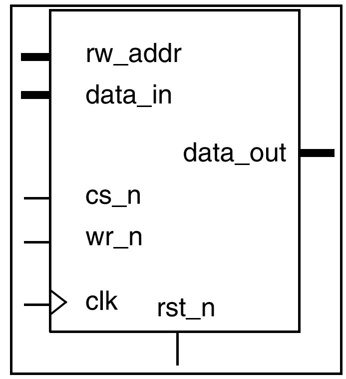
DW_ram_rw_s_dff

Synchronous Single-Port, Read/Write RAM (Flip-Flop-Based)

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word depth
- Parameterized data width
- Synchronous static memory
- Parameterized reset mode (asynchronous or synchronous)
- High testability using DFT Compiler



Description

DW_ram_rw_s_dff implements a parameterized, synchronous, single-port static RAM.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
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
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 2048 Default = none	Width of `data_in` and `data_out` buses
depth	2 to 1024 Default = none	Number of words in the memory array (address width)
rst_mode	0 or 1 Default = 1	Determines the reset methodology: 0 = `rst_n` asynchronously initializes the RAM 1 = `rst_n` synchronously initializes the RAM

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_S_DFF_CFG_SIM	VHDL simulation configuration
dw/dw06/src/DW_ram_rw_s_dff_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_ram_rw_s_dff.v	Verilog simulation model source code

The write operation of the RAM is fully synchronous with respect to the clock, `clk`. The read operation is asynchronous to the clock, allowing the data written into the RAM to be instantly read.

The write data enters the RAM through the `data_in` input port, and is read out through 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 the memory. For addresses beyond the maximum depth (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.

Chip Selection, Reading and Writing

The `cs_n` input is the chip select, active low signal that enables the RAM. When `cs_n` is LOW, data is constantly read from the RAM.

When `wr_n`, the active low write enable, is LOW, and `cs_n` is LOW, data is written into the RAM on the rising edge of `clk`.

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

Reset

The rst_n port is an active low input that initializes the RAM to zeros independent of the value of cs_n. If the *rst_mode* parameter is set to 0, rst_n asynchronously resets the RAM. If the *rst_mode* parameter is set to 1, rst_n synchronously resets the RAM. If the rst_n port is tied high, synthesis optimizes the logic, and builds a non-resetable RAM.

Application Notes

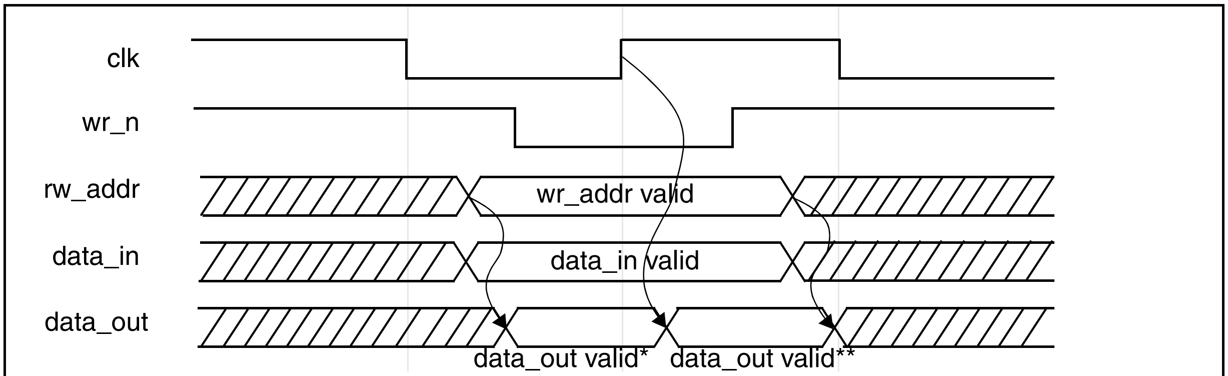
DW_ram_rw_s_dff is intended to be used as small scratch-pad memory or register file. Because DW_ram_rw_s_dff is built from the cells within the ASIC cell library, it should be kept small to obtain an efficient implementation. If a larger memory is required, you should consider using a hard macro RAM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

Timing Waveforms

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

Figure 1-1 Instantiated RAM Timing Waveforms

Write Timing, rst_n = 1, cs_n = 0



* data_out has the value of the previous word stored in the location pointed to by the current address.

** data_out has the value of the new word (value from data_in) which is now stored at the current address.

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

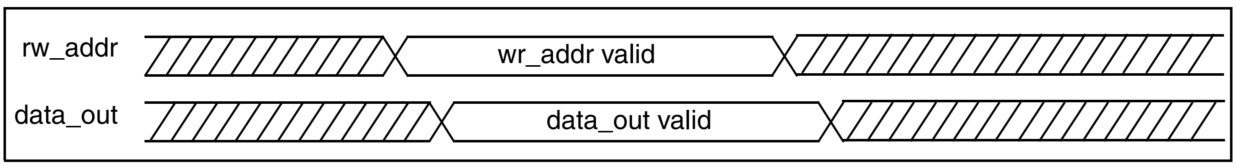
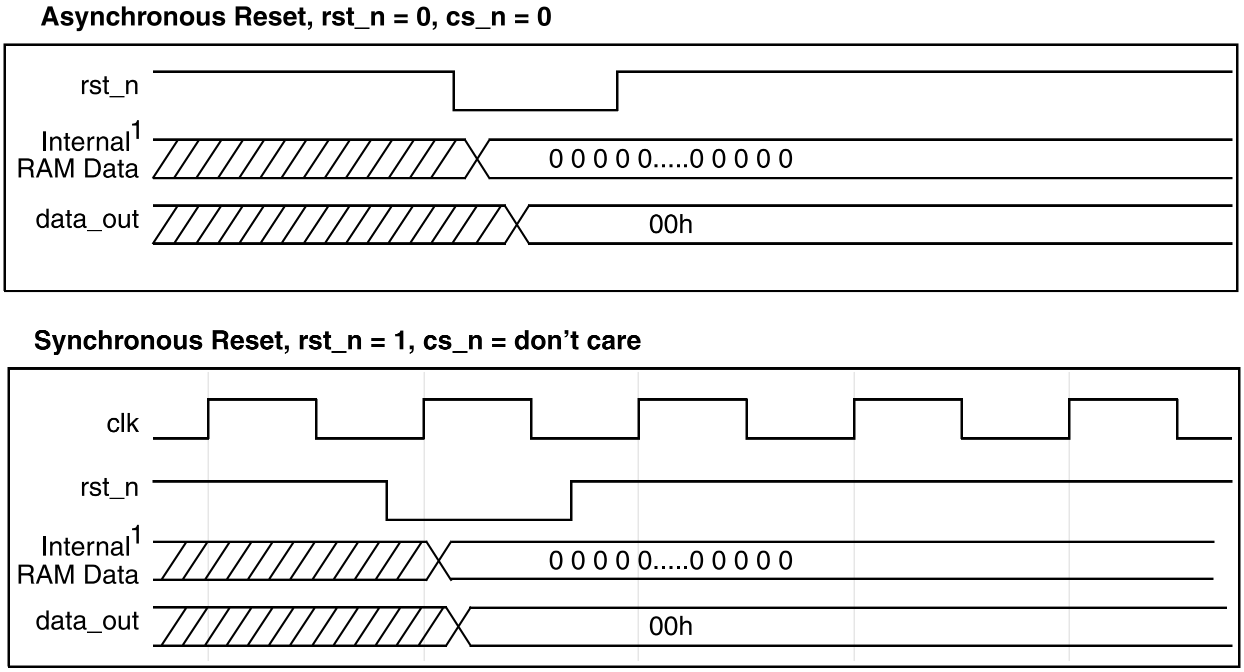


Figure 1-2 RAM Reset Timing Waveforms



[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_rw_s_dff_inst is
  generic (inst_data_width : INTEGER := 8;
           inst_depth      : INTEGER := 8;
           inst_rst_mode   : INTEGER := 0  );
  port (inst_clk     : in std_logic;   inst_rst_n   : in std_logic;
        inst_cs_n    : in std_logic;   inst_wr_n    : 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_s_dff_inst;

architecture inst of DW_ram_rw_s_dff_inst is
begin

  -- Instance of DW_ram_rw_s_dff
  U1 : DW_ram_rw_s_dff
  generic map (data_width => inst_data_width,   depth => inst_depth,
               rst_mode => inst_rst_mode )
  port map (clk => inst_clk,   rst_n => inst_rst_n,   cs_n => inst_cs_n,
            wr_n => inst_wr_n,   rw_addr => inst_rw_addr,
            data_in => inst_data_in,   data_out => data_out_inst );
end inst;

-- pragma translate_off
configuration DW_ram_rw_s_dff_inst_cfg_inst of DW_ram_rw_s_dff_inst is
  for inst
  end for; -- inst
end DW_ram_rw_s_dff_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_ram_rw_s_dff_inst(inst_clk, inst_rst_n, inst_cs_n, inst_wr_n,
                            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_clk;
  input inst_rst_n;
  input inst_cs_n;
  input inst_wr_n;
  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_s_dff
  DW_ram_rw_s_dff #(data_width, depth, rst_mode)
    U1 (.clk(inst_clk),   .rst_n(inst_rst_n),   .cs_n(inst_cs_n),
        .wr_n(inst_wr_n),   .rw_addr(inst_rw_addr),   .data_in(inst_data_in),
        .data_out(data_out_inst) );
endmodule
```

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

www.synopsys.com