

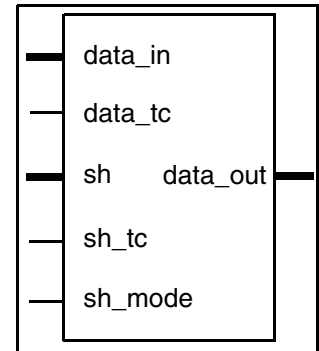
# DW\_shifter

## Combined Arithmetic and Barrel Shifter

Version, STAR and Download Information: [IP Directory](#)

### Features and Benefits

- Dynamically selectable arithmetic or barrel shift mode
- Parameterized input control (inverted and non-inverted logic)
- Parameterized padded logic value control (for arithmetic shift only)
- Parameterized data and shift coefficient word lengths
- Inferable using a function call (support for `inv_mode = 0` only)



### Description

DW\_shifter is a general-purpose shifter that can be dynamically programmed to operate in arithmetic or barrel shift mode.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data_in	<i>data_width</i> bit(s)	Input	Input data
data_tc	1 bit	Input	Two's complement control on data_in 0 = unsigned data_in 1 = signed data_in
sh	<i>sh_width</i> bit(s)	Input	Shift control
sh_tc	1 bit	Input	Two's complement control on sh 0 = unsigned sh 1 = signed sh
sh_mode	1 bit	Input	Arithmetic or barrel shift mode 0 = barrel shift mode 1 = arithmetic shift mode
data_out	<i>data_width</i> bit(s)	Output	Output data

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	$\geq 2$	Word length of data_in and data_out
sh_width	1 to $(\text{ceil}(\log_2[\text{data\_width}]) + 1)$	Word length of sh

**Table 1-2 Parameter Description (Continued)**

Parameter	Values	Description
inv_mode	0 to 3 Default: 0	logic mode 0 = normal input, 0 padding in output; 1 = normal input, 1 padding in output; 2 = inverted input <sup>a</sup> , 0 padding in output; 3 = inverted input, 1 padding in output

a. Inverted input refers to sh, sh\_tc, and data\_tc pins only.

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation	Function	License Feature Required
mx2	Implement using 2:1 multiplexers only.	DesignWare
mx2i	Synthesis model	DesignWare
mx4	Synthesis model	DesignWare
mx8	Synthesis model	DesignWare

a. During synthesis, Design Compiler will select the appropriate architecture for your constraints. However, you may force Design Compiler to use any architectures described in this table. For more, see [DesignWare Building Block IP User Guide](#)

**Table 1-4 Simulation Models**

Model	Function
DW01.DW_shifter_CFG_SIM	Design unit name for VHDL simulation
dw/dw01/src/DW_shifter_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_shifter.v	Verilog simulation model source code

The input data `data_in` is shifted left or right by the number of bits specified by the control input `sh`.

When `sh_mode=0`, barrel shift operation is performed – shifted data wraps around from the MSB to the LSB. An arithmetic shift is performed if `sh_mode=1` – input data is shifted left or right by the number of bits specified by the control input `sh`. For more information, refer to [Table 1-5 on page 3](#) and [Table 1-6 on page 4](#).

When the control signal `sh_tc=0`, the coefficient `sh` is interpreted as an unsigned positive number, and DW\_shifter performs only left shift operations.

When `sh_tc=1`, `sh` is a two's complement number, with a negative coefficient performing a right shift and a positive coefficient performing a left shift.

The input data, `data_in`, is interpreted as an unsigned number when `data_tc=0`. When `data_tc=1`, `data_in` is interpreted as a signed number, and a sign extension is performed for right arithmetic shift operations.

The parameter `inv_mode` determines the following:

- Interpretation of input data as non-inverted or inverted logic (for `data_tc`, `sh`, and `sh_tc` pins only)
- Padded logic value at the output (for arithmetic shift only)

**Table 1-5 Truth Table**

Parameter <code>inv_mode</code>	MSB of <code>sh</code>	<code>sh_tc</code>	<code>data_tc</code>	<code>sh_mode</code>	Operation
0	<code>_a</code>	0	-	1	Left arithmetic shift, logic 0 padding
	0	1	-	1	Left arithmetic shift, logic 0 padding
	1	1	0	1	Right arithmetic shift, logic 0 padding
	1	1	1	1	Right arithmetic shift, sign-extended padding
	-	0	-	0	Left barrel shift
	0	1	-	0	Left barrel shift
	1	1	-	0	Right barrel shift
1	-	0	-	1	Left arithmetic shift, logic 1 padding
	0	1	-	1	Left arithmetic shift, logic 1 padding
	1	1	0	1	Right arithmetic shift, logic 1 padding
	1	1	1	1	Right arithmetic shift, sign-extended padding
	-	0	-	0	Left barrel shift
	0	1	-	0	Left barrel shift
	1	1	-	0	Right barrel shift

Table 1-5 Truth Table (Continued)

Parameter inv_mode	MSB of sh	sh_tc	data_tc	sh_mode	Operation
2	-	1	-	1	Left arithmetic shift, logic 0 padding
	1	0	-	1	Left arithmetic shift, logic 0 padding
	0	0	1	1	Right arithmetic shift, logic 0 padding
	0	0	0	1	Right arithmetic shift, sign-extended padding
	-	1	-	0	Left barrel shift
	1	0	-	0	Left barrel shift
	0	0	-	0	Right barrel shift
3	-	1	-	1	Left arithmetic shift, logic 1 padding
	1	0	-	1	Left arithmetic shift, logic 1 padding
	0	0	1	1	Right arithmetic shift, logic 1 padding
	0	0	0	1	Right arithmetic shift, sign-extended padding
	-	1	-	0	Left barrel shift
	1	0	-	0	Left barrel shift
	0	0	-	0	Right barrel shift

a. “-” indicates a “don’t care” state.

Table 1-6 Example (data\_width=8, sh\_width=3)

sh(2:0) inv_mode <sup>a</sup>		sh_tc inv_mode		data_tc inv_mode		sh_mode	B(7) <sup>b</sup>	B(6)	B(5)	B(4)	B(3)	B(2)	B(1)	B(0)
0, 1	2, 3	0, 1	2, 3	0, 1	2, 3									
000	111	0	1	- <sup>c</sup>	-	1	A(7) <sup>d</sup>	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)
001	110	0	1	-	-	1	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	F
010	101	0	1	-	-	1	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	F	F
011	100	0	1	-	-	1	A(4)	A(3)	A(2)	A(1)	A(0)	F	F	F
100	011	0	1	-	-	1	A(3)	A(2)	A(1)	A(0)	F	F	F	F
101	010	0	1	-	-	1	A(2)	A(1)	A(0)	F	F	F	F	F
110	001	0	1	-	-	1	A(1)	A(0)	F	F	F	F	F	F

Table 1-6 Example (data\_width=8, sh\_width=3) (Continued)

sh(2:0) inv_mode <sup>a</sup>		sh_tc inv_mode		data_tc inv_mode		sh_mode	B(7) <sup>b</sup>	B(6)	B(5)	B(4)	B(3)	B(2)	B(1)	B(0)
111	000	0	1	-	-	1	A(0)	F	F	F	F	F	F	F
000	111	1	0	-	-	1	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)
001	110	1	0	-	-	1	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	F
010	101	1	0	-	-	1	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	F	F
011	100	1	0	-	-	1	A(4)	A(3)	A(2)	A(1)	A(0)	F	F	F
100	011	1	0	0	1	1	F	F	F	F	A(7)	A(6)	A(5)	A(4)
101	010	1	0	0	1	1	F	F	F	A(7)	A(6)	A(5)	A(4)	A(3)
110	001	1	0	0	1	1	F	F	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)
111	000	1	0	0	1	1	F	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)
100	011	1	0	1	0	1	A(7)	A(7)	A(7)	A(7)	A(7)	A(6)	A(5)	A(4)
101	010	1	0	1	0	1	A(7)	A(7)	A(7)	A(7)	A(6)	A(5)	A(4)	A(3)
110	001	1	0	1	0	1	A(7)	A(7)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)
111	000	1	0	1	0	1	A(7)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)
000	111	0	1	-	-	0	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)
001	110	0	1	-	-	0	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	A(7)
010	101	0	1	-	-	0	A(5)	A(4)	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)
011	100	0	1	-	-	0	A(4)	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)
100	011	0	1	-	-	0	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)
101	010	0	1	-	-	0	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)	A(3)
110	001	0	1	-	-	0	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)
111	000	0	1	-	-	0	A(0)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)
100	011	1	0	-	-	0	A(3)	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)
101	010	1	0	-	-	0	A(2)	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)	A(3)
110	001	1	0	-	-	0	A(1)	A(0)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)
111	000	1	0	-	-	0	A(0)	A(7)	A(6)	A(5)	A(4)	A(3)	A(2)	A(1)

a. Fill Value: F = 0 for inv\_mode = 0 or 2. F = 1 for inv\_mode = 1 or 3.

b. "B" stands for data\_out.

c. "-" indicates a "don't care" state.

d. "A" stands for data\_in.

## Application Notes

Proper selection of the `inv_mode` parameter at design time can eliminate the need for glue logic around the shifter. The value of the `inv_mode` parameter depends on the following:

- The signals arriving at the input pins (`sh`, `sh_tc`, and `data_tc`) being positive or negative logic (for example, if the input signals follow negative logic, parameter `inv_mode` can be selected as 2. This eliminates the need for inserting inverters at the shifter inputs).
- The padded value at the output for arithmetic shift operations (for example, if logic 1 is the desired padded value, parameter `inv_mode` can be selected as 1).

All four possible combinations of input logic values and output fill values are covered within the `inv_mode` parameter range.

## Related Topics

- [Math – Arithmetic Overview](#)
- [DesignWare Building Block IP Documentation Overview](#)

## HDL Usage Through Function Inferencing - VHDL

```

library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use DWARE.DW_foundation_arith.all;

entity DW_shifter_func is
  generic(func_data_width:integer:=8; func_sh_width : integer := 3);
  port(func_data_in : in std_logic_vector(func_data_width-1 downto 0);
        func_data_tc : in std_logic;
        func_sh      : in std_logic_vector(func_sh_width-1 downto 0);
        func_sh_tc   : in std_logic;
        func_sh_mode : in std_logic;
        data_out_func: out std_logic_vector(func_data_width-1 downto 0) );
end DW_shifter_func;

architecture func of DW_shifter_func is
begin
  process (func_data_tc, func_sh_tc, func_data_in, func_sh, func_sh_mode)
  begin
    if func_DATA_TC = '0' and func_SH_TC = '0' then
      data_out_func <=
        std_logic_vector(DWF_shifter(unsigned(func_data_in),
                                     unsigned(func_SH), func_sh_mode));
    elsif func_DATA_TC = '1' and func_SH_TC = '0' then
      data_out_func <=
        std_logic_vector(DWF_shifter(signed(func_data_in),
                                     unsigned(func_SH), func_sh_mode) );
    elsif func_DATA_TC = '1' and func_SH_TC = '1' then
      data_out_func <=
        std_logic_vector(DWF_shifter(signed(func_data_in),
                                     signed(func_SH), func_sh_mode) );
    else
      data_out_func <=
        std_logic_vector(DWF_shifter(unsigned(func_data_in),
                                     signed(func_SH), func_sh_mode));
    end if;
  end process;
end func;

```

**Attention**Function inferencing supports *inv\_mode* = 0 only

## HDL Usage Through Function Inferencing - Verilog

```
module DW_shifter_func (func_data_in, func_data_tc, func_sh,
                        func_sh_tc, func_sh_mode, data_out_func);
    parameter func_data_width = 8;
    parameter func_sh_width = 3;

    // Passes the widths to the shifter function
    parameter data_width = func_data_width;
    parameter sh_width = func_sh_width;

    // Please add search_path = search_path + {synopsys_root + "/dw/sim_ver"}
    // to your .synopsys_dc.setup file (for synthesis) and add
    // +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator command line
    // (for simulation).
    `include "DW_shifter_function.inc"

    input [func_data_width-1:0] func_data_in;
    input [func_sh_width-1:0] func_sh;
    input func_data_tc, func_sh_tc, func_sh_mode;

    output [func_data_width-1:0] data_out_func;

    reg [func_data_width-1:0] data_out_func;

    // infer DW_shifter

    always @ (func_data_in or func_data_tc or func_sh
              or func_sh_tc or func_sh_mode)
    begin
        casex({func_data_tc,func_sh_tc}) // synopsys full_case
            2'b00: data_out_func =
                DWF_shifter_uns_uns(func_data_in,func_sh,func_sh_mode);
            2'b10: data_out_func =
                DWF_shifter_tc_uns(func_data_in,func_sh,func_sh_mode);
            2'b01: data_out_func =
                DWF_shifter_uns_tc(func_data_in,func_sh,func_sh_mode);
            2'b11: data_out_func =
                DWF_shifter_tc_tc(func_data_in,func_sh,func_sh_mode);
        endcase
    end

endmodule
```



## 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_shifter_inst is
  generic ( inst_data_width : POSITIVE := 8;
            inst_sh_width   : POSITIVE := 3;
            inst_inv_mode   : INTEGER := 0 );
  port ( inst_data_in   : in std_logic_vector(inst_data_width-1 downto 0);
        inst_data_tc   : in std_logic;
        inst_sh         : in std_logic_vector(inst_sh_width-1 downto 0);
        inst_sh_tc      : in std_logic;
        inst_sh_mode    : in std_logic;
        data_out_inst  : out std_logic_vector(inst_data_width-1 downto 0) );
end DW_shifter_inst;

architecture inst of DW_shifter_inst is
begin
  -- Instance of DW_shifter
  U1 : DW_shifter
    generic map ( data_width => inst_data_width,   sh_width => inst_sh_width,
                  inv_mode => inst_inv_mode)
    port map ( data_in => inst_data_in,   data_tc => inst_data_tc,
              sh => inst_sh,   sh_tc => inst_sh_tc,
              sh_mode => inst_sh_mode,   data_out => data_out_inst );
end inst;

-- pragma translate_off
configuration DW_shifter_inst_cfg_inst of DW_shifter_inst is
  for inst
    end for; -- inst
end DW_shifter_inst_cfg_inst;
-- pragma translate_on
```

## HDL Usage Through Component Instantiation - Verilog

```
module DW_shifter_inst( inst_data_in, inst_data_tc, inst_sh,
                       inst_sh_tc, inst_sh_mode, data_out_inst );

    parameter data_width = 8;
    parameter sh_width = 3;
    parameter inv_mode = 0;
    input [data_width-1 : 0] inst_data_in;
    input inst_data_tc;
    input [sh_width-1 : 0] inst_sh;
    input inst_sh_tc;
    input inst_sh_mode;
    output [data_width-1 : 0] data_out_inst;

    // Instance of DW_shifter
    DW_shifter #(data_width, sh_width)
        U1 ( .data_in(inst_data_in),    .data_tc(inst_data_tc),    .sh(inst_sh),
            .sh_tc(inst_sh_tc),    .sh_mode(inst_sh_mode),
            .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](http://www.synopsys.com)

