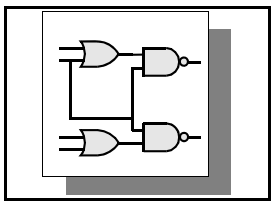
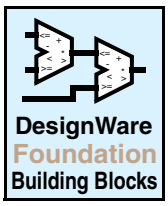




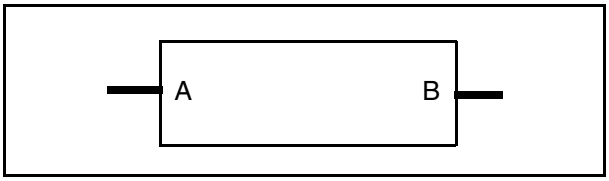
DW01_decode

Decoder

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word length
- Inferable using a function call



Description

DW01_decode decodes an address on input port `A` to a single bitline on output port `B`. The selected bitline on port `B` is active high.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	*width*	Input	Binary input data
B	2^{width}	Output	Decoded output data

Table 1-2 Parameter Description

Parameter	Values	Description
width[a]	≥ 1	Word length of input A is *width*. Word length of output B is 2^{width}

a. The `width` parameter value causes the size of output B to grow exponentially. Therefore, a `width` value greater than 12 will result in abnormally long compile times.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	none

Table 1-4 Simulation Models

Model	Function
DW01.DW01_DECODE_CFG_SIM	Design unit name for VHDL simulation
dw/dw01/src/DW01_decode_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW01_decode.v	Verilog simulation model source code

Table 1-5 Truth Table (width = 3)

A(2:0)	B(7)	B(6)	B(5)	B(4)	B(3)	B(2)	B(1)	B(0)
000	0	0	0	0	0	0	0	1
001	0	0	0	0	0	0	1	0
010	0	0	0	0	0	1	0	0
011	0	0	0	0	1	0	0	0
100	0	0	0	1	0	0	0	0
101	0	0	1	0	0	0	0	0
110	0	1	0	0	0	0	0	0
111	1	0	0	0	0	0	0	0

Related Topics

- Logic – Combinational 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 DW01_decode_func is
  generic(func_width : integer := 5);
  port( func_A:     in  std_logic_vector(func_width-1 downto 0);
        B_func_TC:  out std_logic_vector(2**func_width-1 downto 0);
        B_func_UNS: out std_logic_vector(2**func_width-1 downto 0);
        B_func:     out std_logic_vector(2**func_width-1 downto 0));
end DW01_decode_func;

architecture func of DW01_decode_func is
begin

  B_func_TC  <= std_logic_vector(DWF_decode (signed (func_A)));
  B_func_UNS <= std_logic_vector(DWF_decode (unsigned (func_A)));
  B_func     <= DWF_decode (func_A);
end func;
```

HDL Usage Through Function Inferencing - Verilog

```
module DW01_decode_func (func_A,B_func);
  parameter func_width = 4;

  // Passes the width to the decode function
  parameter width = func_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 "DW01_decode_function.inc"

  input [func_width-1:0] func_A;

  output [(1 << func_width)-1:0] B_func;

  assign B_func = DWF_decode(func_A);

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 DW01_decode_inst is
  generic (inst_width : NATURAL := 8);
  port (inst_A : in std_logic_vector(inst_width-1 downto 0);
        B_inst : out std_logic_vector(2**inst_width-1 downto 0));
end DW01_decode_inst;

architecture inst of DW01_decode_inst is
begin

  -- Instance of DW01_decode
  U1 : DW01_decode
  generic map ( width => inst_width )
  port map ( A => inst_A, B => B_inst );
end inst;

-- pragma translate_off
configuration DW01_decode_inst_cfg_inst of DW01_decode_inst is
  for inst
  end for; -- inst
end DW01_decode_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

Because Verilog does not support an exponentiation operator, you must explicitly set your input and output port widths. Check Table 1-1 on page 1 for details on the relationship between input and output port widths. You also must explicitly set a value for the parameter in your instantiation statement.

```
module DW01_decode_inst( inst_A, B_inst );

  parameter width = 8;

  input [width-1 : 0] inst_A;
  output [(1<<width)-1 : 0] B_inst;

  // Instance of DW01_decode
  DW01_decode #(width)
    U1 ( .A(inst_A), .B(B_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