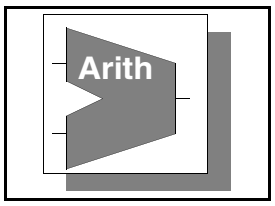
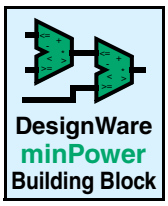


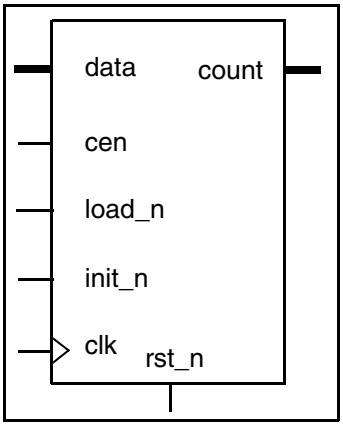


DW_cntr_gray

Gray Code Counter

Version, STAR and Download Information: IP Directory

Features and Benefits



- Gray encoded output
- Asynchronous and synchronous reset
- Count enable
- Provides minPower benefits with the DesignWare-LP license (Get the minPower version of this datasheet.)

Description

DW_cntr_gray is a Gray code counter. The counter is *width* bits wide and has 2^{width} states. The counter is clocked on the positive edge of the `clk` input. Because the count sequence is Gray code, only one counter bit changes value between successive states. DW_cntr_gray also has optional low power benefits.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, asynchronous, active low
init_n	1 bit	Input	Reset, synchronous, active low
load_n	1 bit	Input	Enable data load to counter, active low
data	*width* bit(s)	Input	Counter load input
cen	1 bit	Input	Count enable, active high
count	*width* bit(s)	Output	Gray coded counter output

Table 1-2 Parameter Description

Parameter	Values	Description
width	≥ 1	Word length of counter

Table 1-3 Synthesis Implementations[a]

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	DesignWare
cla	Carry-lookahead 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
DW03.DW_chtr_gray_cfg_sim	Design unit name for VHDL simulation
dw/dw03/src/DW_cntr_gray_sim.vhd	VHDL simulation model source code
dw_sim_ver/DW_cntr_gray.v	Verilog simulation model source code

The active-low `rst_n` signal provides for an asynchronous reset of the counter to “000...0”. The active-low `init_n` signal provides a synchronous reset of the counter to “000...0”.

When the count enable pin `cen` is high, the counter is active. When `cen` is low, the counter is disabled and `count` remains unchanged.

The counter operates according to the following Truth Table:

rst_n	init_n	load_n	cen	Operation
0	X	X	X	Reset (asynchronous)
1	0	X	X	Reset (synchronous)
1	1	0	X	Load
1	1	1	0	Standby
1	1	1	1	Count

Timing Diagrams

Figure 1-1 Functional Operation with asynchronous reset

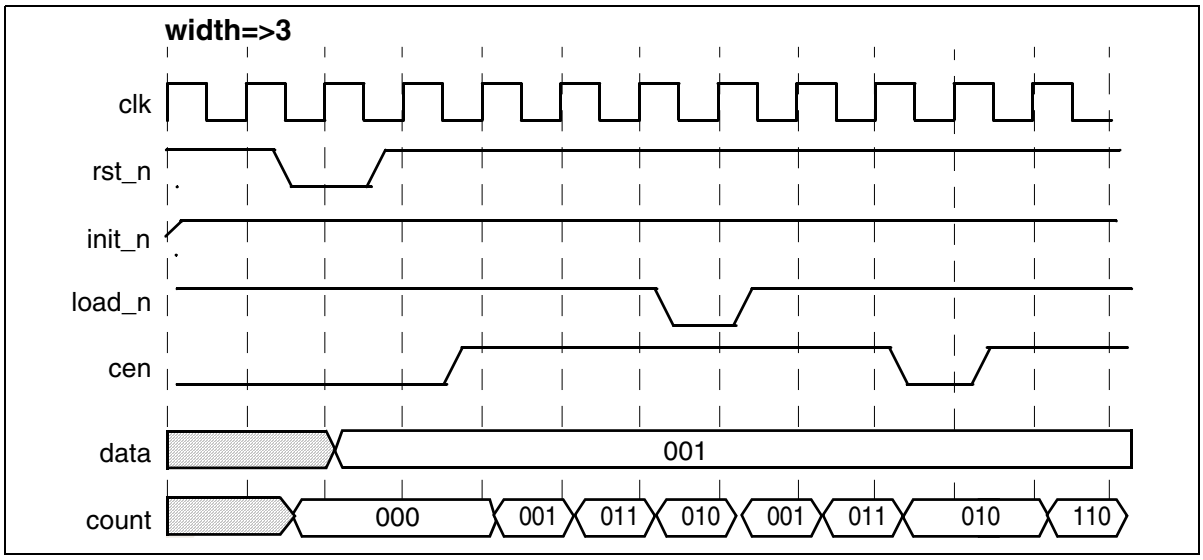
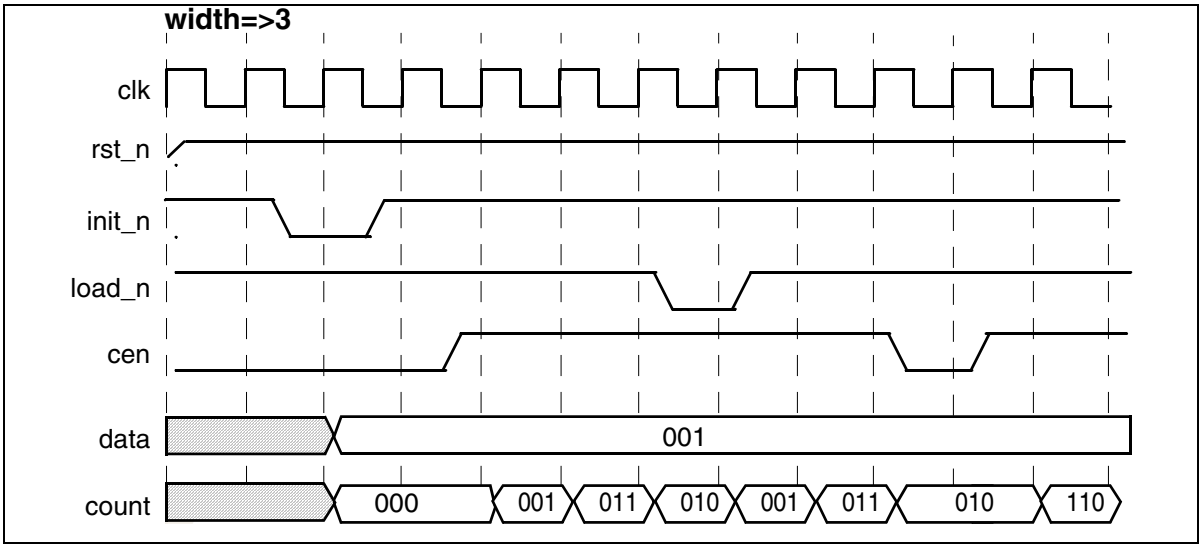


Figure 1-2 Functional Operation with synchronous reset



Related Topics

- Math – Arithmetic Overview
- DesignWare Building Block Documentation Overview

HDL Usage Through Component Instantiation - VHDL

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_cntr_gray_inst is
  generic (inst_width : positive := 8);
  port (inst_clk    : in  std_logic;
        inst_rst_n  : in  std_logic;
        inst_init_n : in  std_logic;
        inst_load_n : in  std_logic;
        inst_data   : in  std_logic_vector(inst_width-1 downto 0);
        inst_cen    : in  std_logic;
        count_inst  : out std_logic_vector(inst_width-1 downto 0));
end DW_cntr_gray_inst;

architecture inst of DW_cntr_gray_inst is
begin
  -- instance of DW_cntr_gray
  U1 : DW_cntr_gray
    generic map (width => inst_width)
    port map (clk    => inst_clk,
              rst_n  => inst_rst_n,
              init_n => inst_init_n,
              load_n => inst_load_n,
              data   => inst_data,
              cen    => inst_cen,
              count  => count_inst);
end inst;

-- pragma translate_off
configuration DW_cntr_gray_inst_cfg_inst of DW_cntr_gray_inst is
  for inst
  end for;
end DW_cntr_gray_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_cntr_gray_inst (inst_clk, inst_rst_n, inst_init_n, inst_load_n,
                          inst_data, inst_cen, count_inst);
  parameter inst_width = 8;

  input  inst_clk;
  input  inst_rst_n;
  input  inst_init_n;
  input  inst_load_n;
  input  [inst_width-1 : 0] inst_data;
  input  inst_cen;
  output [inst_width-1 : 0] count_inst;

  // Please add +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator
  // command line (for simulation).

  // instance of DW_cntr_gray
  DW_cntr_gray #(inst_width)
    U1 (.clk(inst_clk),
        .rst_n(inst_rst_n),
        .init_n(inst_init_n),
        .load_n(inst_load_n),
        .data(inst_data),
        .cen(inst_cen),
        .count(count_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