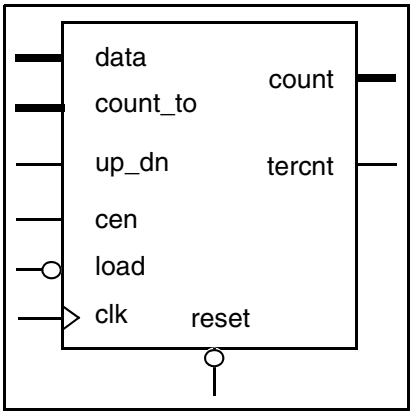
DW03_bictr_dcnto

Up/Down Binary Counter with Dynamic Count-to Flag

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word length
- Terminal count flag for count-to comparison
- Pin-programmable count-to value
- Up/down count control
- Asynchronous reset
- Synchronous counter load
- Synchronous count enable
- Provides minPower benefits with the DesignWare-LP license (Get the minPower version of this datasheet.)



Description

DW03_bictr_dcnto is a general-purpose up/down counter with dynamic count-to logic.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
data	*width*	Input	Counter load input
count_to	*width*	Input	Count compare input
up_dn	1	Input	High for count up and low for count down
load	1	Input	Enable data load to counter, active low
cen	1	Input	Count enable, active high
clk	1	Input	Clock
reset	1	Input	Counter reset, active low
count	*width*	Output	Output count bus
tercnt	1	Output	Terminal count flag, active high

Table 1-2 Parameter Description

Parameter	Values	Description
width	≥ 1	Width of data input bus

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW03.DW03_BICTR_DCNTO_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW03_bictr_dcnto_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW03_bictr_dcnto.v	Verilog simulation model source code

Table 1-5 Counter Operation Truth Table

reset	load	cen	up_dn	Operation
0	X	X	X	Reset
1	0	X	X	Load
1	1	0	X	Standby
1	1	1	0	Count down
1	1	1	1	Count up

When the count value equals the value on the `count_to` pin, the signal `tercnt` (terminal count) is asserted (HIGH). The signal `tercnt` can be connected to `load` through an inversion to synchronously reset the counter to a predefined value on the input pin of the data bus, `data`.

The counter is *width* bits wide and has 2^{width} states from “000...0” to “111...1”. The counter is clocked on the positive edge of `clk`.

The `reset`, active low, provides for an asynchronous reset of the counter to “000...0”. If the reset pin is connected to ‘1’, then the reset logic is not synthesized, resulting in a smaller and faster counter.

The `count_to` is an input bus that ranges from 0 to *width*-1. When the counter output, `count`, equals `count_to`, `tercnt` goes HIGH for one clock cycle.

The `up_dn` input controls whether the counter counts up (`up_dn` is HIGH) or down (`up_dn` is LOW), starting on the next positive edge of `clk`.

The counter is loaded with `data` by asserting `load` (LOW) and applying data to `data`. The data load operation is synchronous with respect to the positive edge of `clk`.

The count enable pin, `cen`, is active high. When `cen` is HIGH, the counter is active. When `cen` is LOW, the counter is disabled, and `count` remains at the same value.

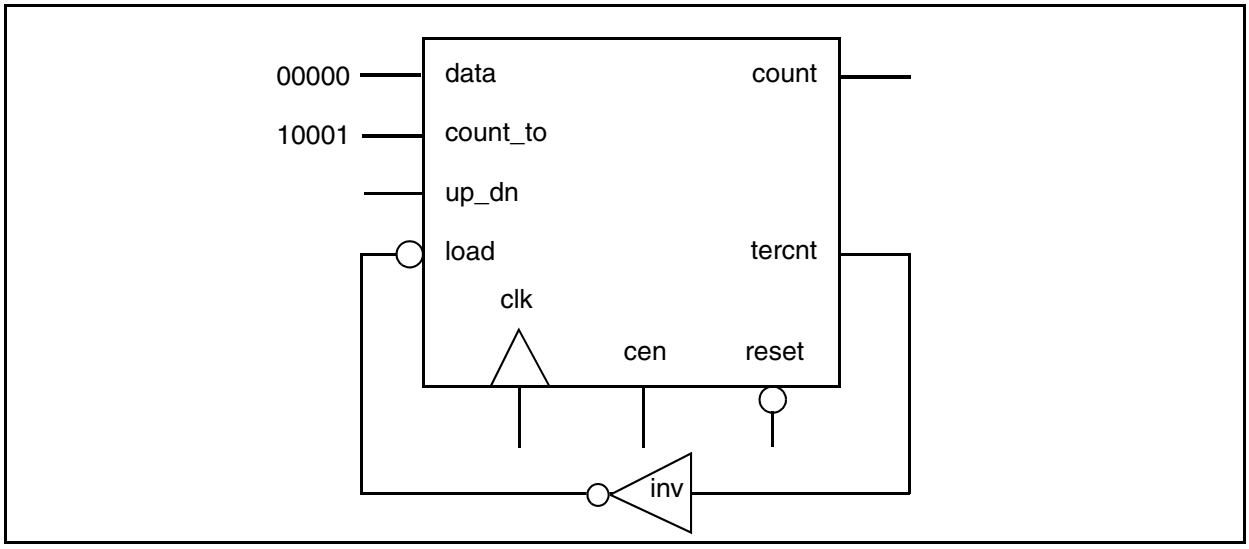
Application Example

An example application of DW03_bictr_dcnto is to count from 0 to 17, repeatedly. This is done by:

1. Connecting the terminal count output signal, `tercnt`, to the `load` input.
2. Setting the `data` input to the start of the count sequence (for example, “00000”).
3. Connecting the `count_to` input to the end of the count sequence (for example, “10001”).

The `count` output then cycles through the states 00000 (data) to 10001 (`count_to`). Refer to Figure 1-1.

Figure 1-1 Counter Application: width = 5



Low Power Implementation

This component provides low power (minPower) benefits when the “lpwr” implementation is chosen, and the DesignWare-LP license is available. Effectiveness of low power design depends on the use of the `-gate_clock` option to `compile_ultra` command.

Timing Diagrams

Figure 1-2 and Figure 1-3 show various timing diagrams for DW03_bictr_dcnto.

Figure 1-2 Functional Operation: reset, load, and count_to

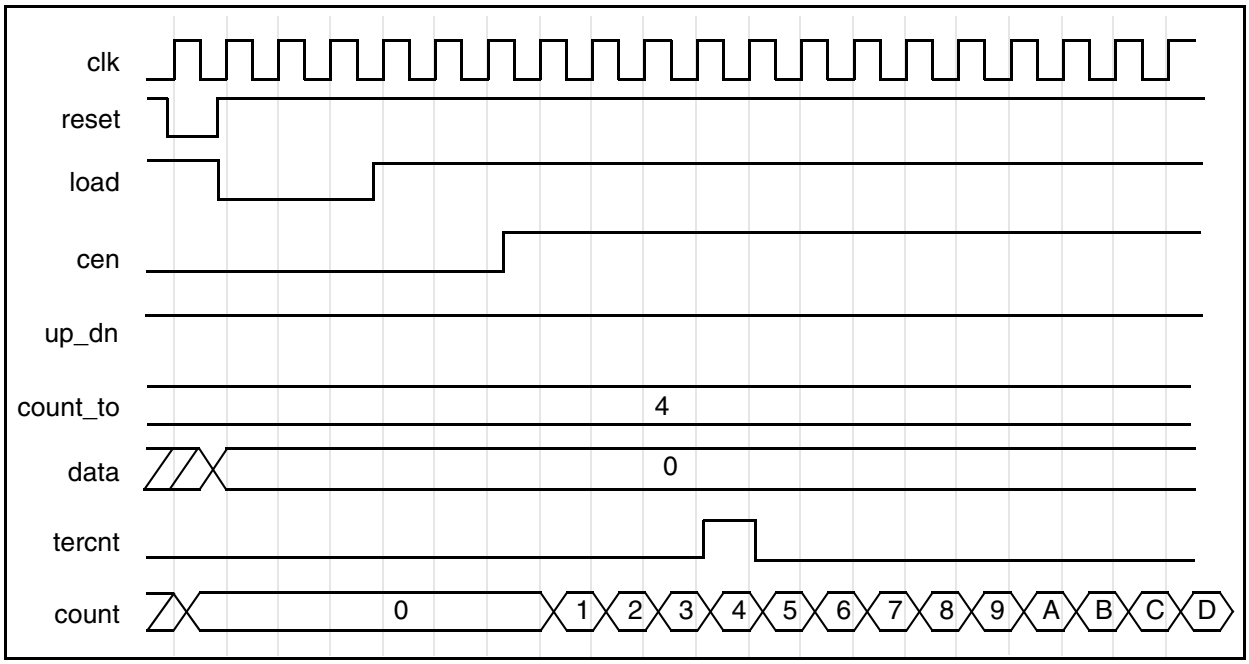
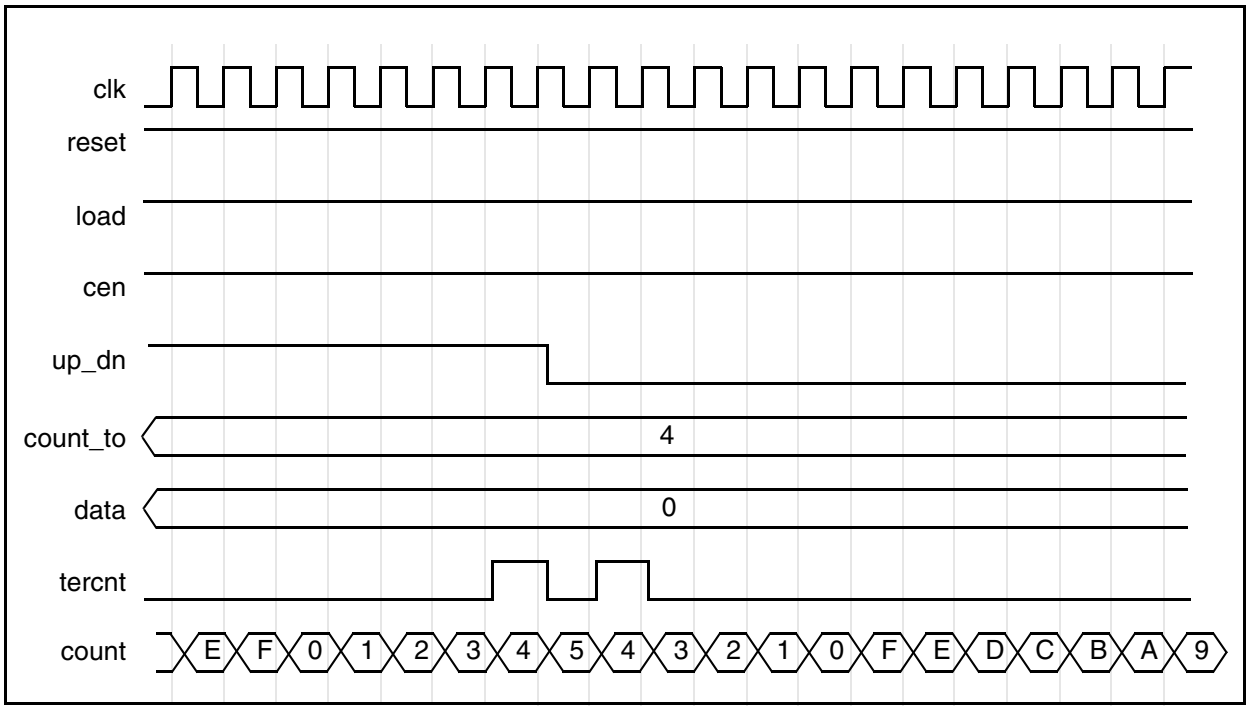


Figure 1-3 Functional Operation: Up and Down Counting, and Count-to Sequence



Related Topics

- Logic – Sequential Overview
- 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 DW03_bictr_dcnto_inst is
  generic (inst_width : POSITIVE := 8);
  port (inst_data     : in std_logic_vector(inst_width-1 downto 0);
        inst_count_to : in std_logic_vector(inst_width-1 downto 0);
        inst_up_dn    : in std_logic;
        inst_load     : in std_logic;
        inst_cen      : in std_logic;
        inst_clk      : in std_logic;
        inst_reset    : in std_logic;
        count_inst    : out std_logic_vector(inst_width-1 downto 0);
        tercnt_inst   : out std_logic);
end DW03_bictr_dcnto_inst;

architecture inst of DW03_bictr_dcnto_inst is
begin

  -- Instance of DW03_bictr_dcnto
  U1 : DW03_bictr_dcnto
    generic map ( width => inst_width )
    port map ( data => inst_data, count_to => inst_count_to,
               up_dn => inst_up_dn, load => inst_load, cen => inst_cen,
               clk => inst_clk, reset => inst_reset, count => count_inst,
               tercnt => tercnt_inst );
  end inst;

-- pragma translate_off
configuration DW03_bictr_dcnto_inst_cfg_inst of DW03_bictr_dcnto_inst is
  for inst
  end for; -- inst
end DW03_bictr_dcnto_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW03_bictr_dcnto_inst( inst_data, inst_count_to, inst_up_dn,
                              inst_load, inst_cen, inst_clk, inst_reset,
                              count_inst, tercnt_inst );

  parameter width = 8;

  input [width-1 : 0] inst_data;
  input [width-1 : 0] inst_count_to;
  input inst_up_dn;
  input inst_load;
  input inst_cen;
  input inst_clk;
  input inst_reset;
  output [width-1 : 0] count_inst;
  output tercnt_inst;

  // Instance of DW03_bictr_dcnto
    DW03_bictr_dcnto #(width)
      U1 ( .data(inst_data), .count_to(inst_count_to), .up_dn(inst_up_dn),
           .load(inst_load), .cen(inst_cen), .clk(inst_clk),
           .reset(inst_reset), .count(count_inst), .tercnt(tercnt_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