DW03_bictr_decode

Up / Down Binary Counter with Output Decode

Features

- Up / down count control
- Asynchronous reset
- Loadable count register
- Counter enable
- Terminal count flag

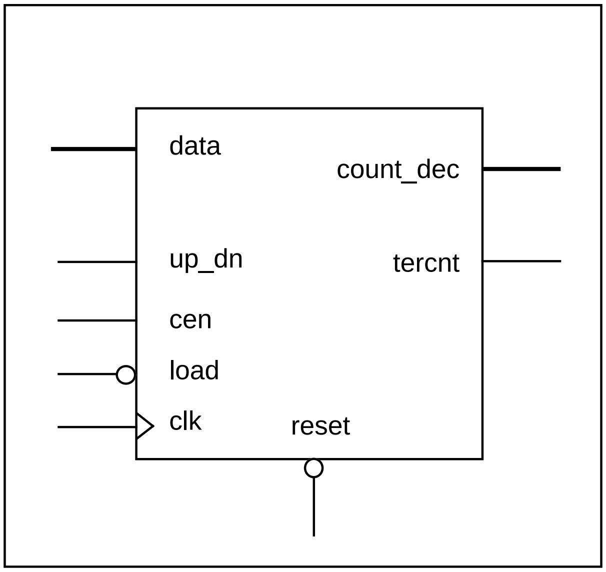
Description

DW03_bictr_decode is a general-purpose up / down counter whose outputs are binary decoded. 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*.

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 *up_dn* input controls whether the counter counts up (*up_dn* HIGH) or down (*up_dn* 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*.



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

count_dec is an output bus that ranges from $2^{width\text{-}1}$ to 0.

tercnt is an output flag that is asserted one state before the count rolls over. When counting up, *tercnt* is HIGH at *count_dec* = "100...000". When counting down, *tercnt* is HIGH at *count_dec* = "000.....001".

Logic: Sequential

Table 1 - Pin Description

Pin Name	Size	Type	Function
data	width	Input	Counter load 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_dec	2^{width}	Output	Binary decoded count value
tercnt	1	Output	Terminal count flag

Table 2 - 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

Table 3 - Parameter Description

Parameter	Function	Legal Range
width	Width of data input bus	≥ 1

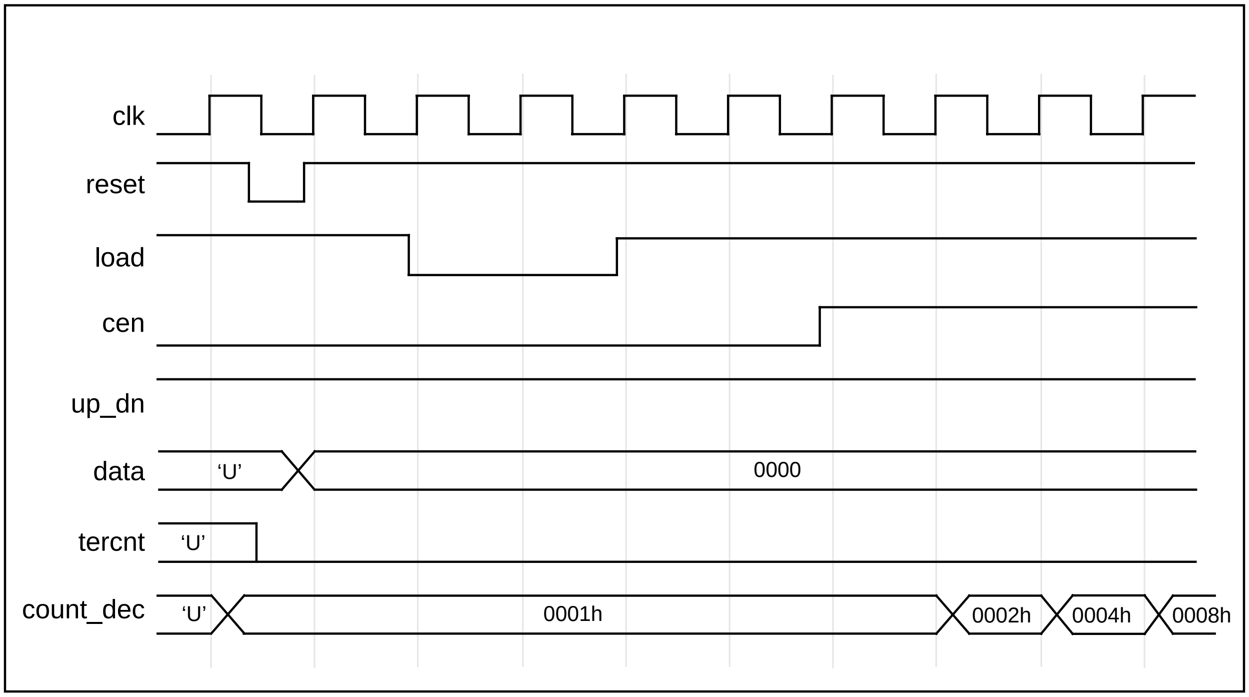
Table 4 - Synthesis Implementations

Implementation Name	Function	License Required
str	Synthesis model	DesignWare-Foundation

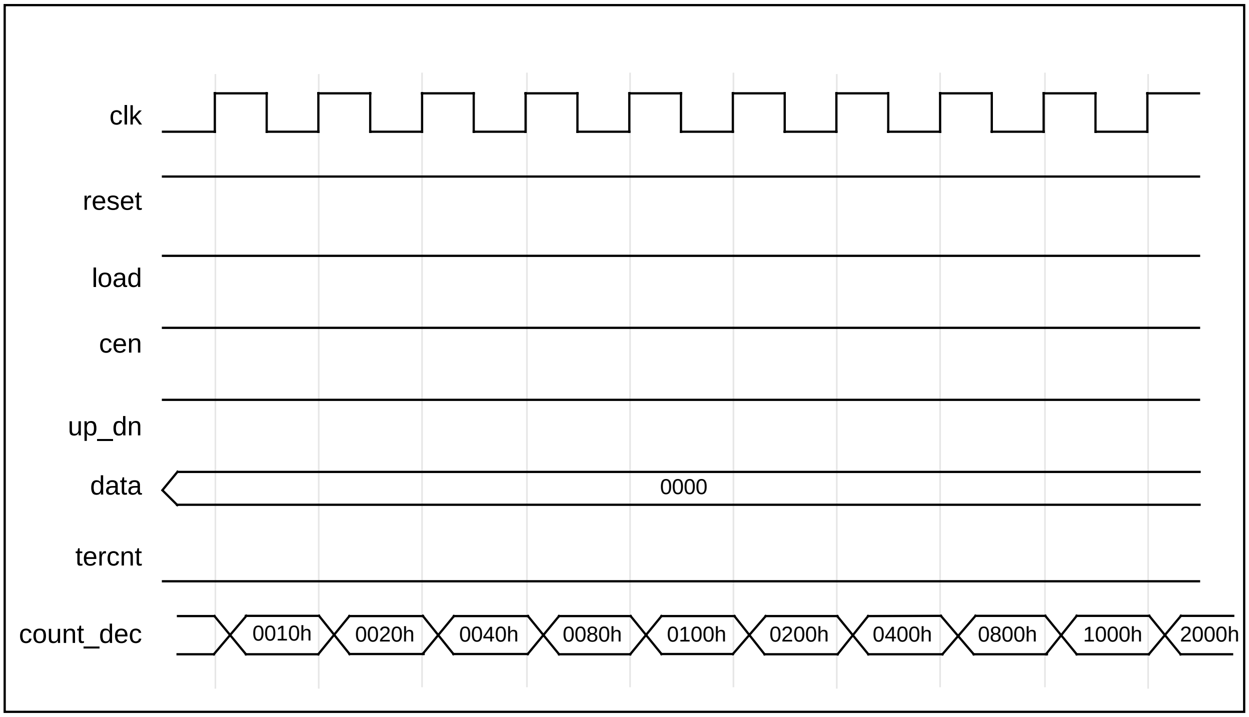
Table 5 - Simulation Models

Model	Function
DW03.DW03_BICTR_DECODE_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW03_bictr_decode_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW03_bictr_decode.v	Verilog simulation model source code

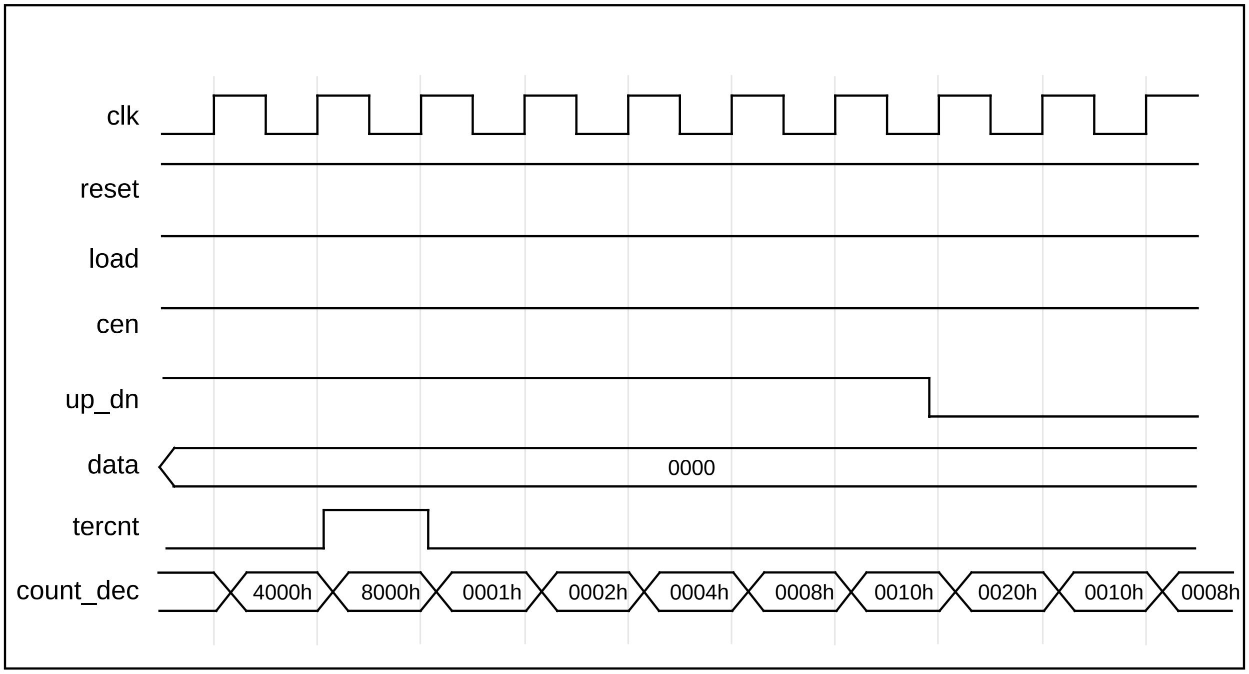
Functional Operation: Reset, Load, and Count Enable



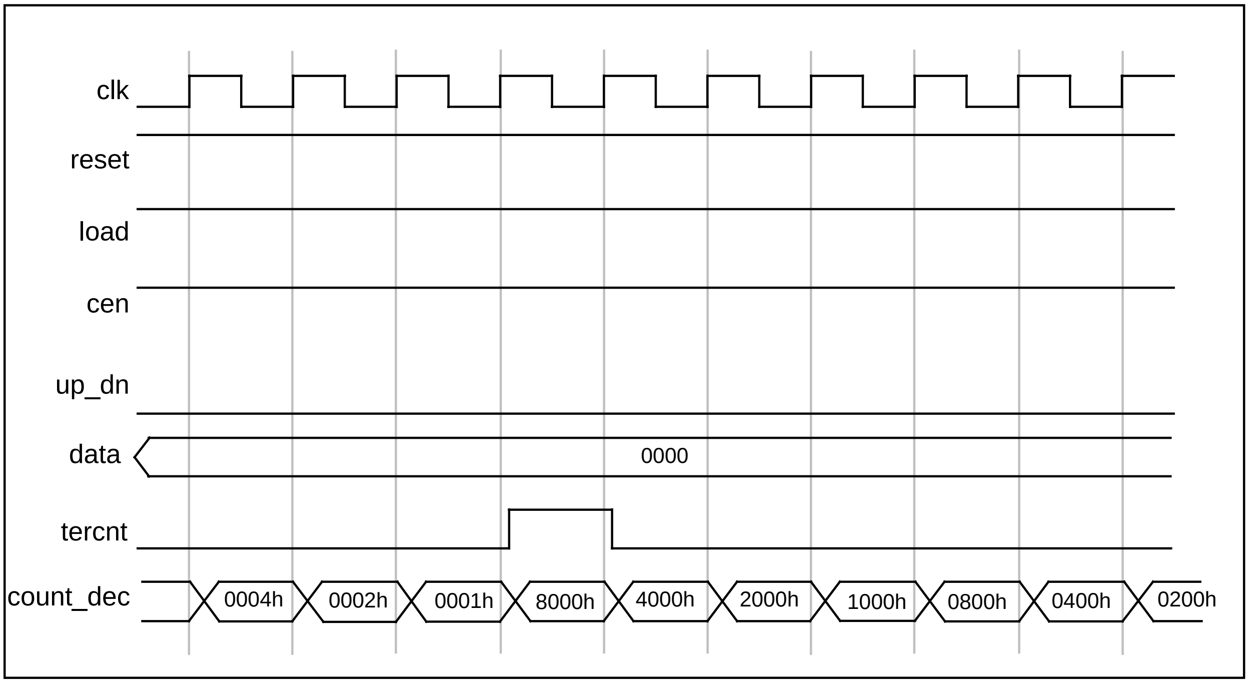
Functional Operation: Up and Down Counting



Functional Operation: Terminal Count Flag



Functional Operation: Effect on Terminal Count



HDL Usage Through Component Instantiation

VHDL

```
library IEEE,DWARE,DW03;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DW03.DW03_components.all;

entity DW03_bictr_decode_inst is
      generic (
       inst_width : POSITIVE := 8
       );
      port (
       inst_data : 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_dec_inst : out std_logic_vector(2**inst_width-1 downto 0);
       tercnt_inst : out std_logic
       );
    end DW03_bictr_decode_inst;

architecture inst of DW03_bictr_decode_inst is

begin

    -- Instance of DW03_bictr_decode
    U1 : DW03_bictr_decode
   generic map ( width => inst_width )
   port map ( data => inst_data, up_dn => inst_up_dn, load => inst_load,
        cen => inst_cen, clk => inst_clk, reset => inst_reset,
        count_dec => count_dec_inst, tercnt => tercnt_inst );

end inst;

-- pragma translate_off
library DW03;
configuration DW03_bictr_decode_inst_cfg_inst of DW03_bictr_decode_inst is
   for inst
        for U1 : DW03_bictr_decode use configuration
                 DW03.DW03_bictr_decode_cfg_sim;
        end for;
   end for; -- inst
end DW03_bictr_decode_inst_cfg_inst;
-- pragma translate_on
```

HOME CONTENTS

Verilog

```
module DW03_bictr_decode_inst( inst_data, inst_up_dn, inst_load, inst_cen,
       inst_clk, inst_reset, count_dec_inst, tercnt_inst );

parameter width = 8;

input [width-1 : 0] inst_data;
input inst_up_dn;
input inst_load;
input inst_cen;
input inst_clk;
input inst_reset;
output [(1<<width)-1 : 0] count_dec_inst;
output tercnt_inst;

    // Instance of DW03_bictr_decode
    DW03_bictr_decode #(width)
     U1 ( .data(inst_data), .up_dn(inst_up_dn), .load(inst_load), .cen(inst_cen),
       .clk(inst_clk), .reset(inst_reset), .count_dec(count_dec_inst),
       .tercnt(tercnt_inst) );

endmodule
```