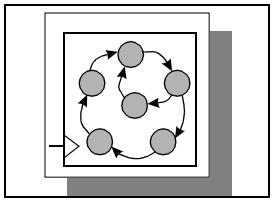


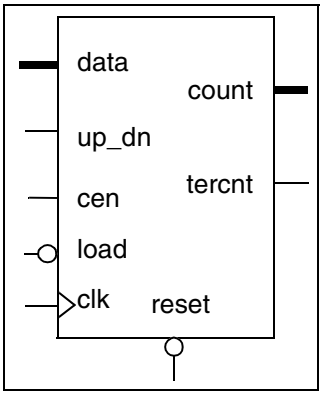
DW03_updn_ctr

Up/Down Counter

Last Revised: Release DWF_0206

Features and Benefits

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



Description

DW03_updn_ctr is a general-purpose binary up-down counter. The counter data path is *width* bits wide and has 2^{width} states from “000...000” to “111...111”, depending on the specified width. The counter is clocked on the positive edge of the `clk` input.

Table 1: Pin Description

Pin Name	Width	Direction	Function
data	*width* bit(s)	Input	Input data bus
up_dn	1 bit	Input	Count up (up_dn = 1) or count down (up_dn = 0)
load	1 bit	Input	Counter load enable, active low
cen	1 bit	Input	Counter enable, active high
clk	1 bit	Input	Clock
reset	1 bit	Input	Asynchronous counter reset, active low
count	*width* bit(s)	Output	Output count bus
tercnt	1 bit	Output	Terminal count flag

Table 2: Parameter Description

Parameter	Value	Function
width	≥ 1	Width of count output bus

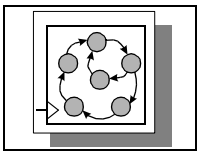


Table 3: Synthesis Implementations[a]

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cla	Carry look-ahead synthesis model	DesignWare
clf	Fast carry look-ahead synthesis model	DesignWare

a. During synthesis, Design Compiler will select the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table. For more details, please refer to the *DesignWare Building Block IP User Guide*.

Table 4: Simulation Models

Model	Function
DW03.DW03_UPDN_CTR_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW03_updn_ctr_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW03_updn_ctr.v	Verilog simulation model source code

Table 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

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

The counter is loaded with data by asserting `load` (LOW) and applying data on the `data` input. 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.

The `tercnt` is an output port. When counting up, `tercnt` is HIGH at `count` = “111....111”. When counting down, `tercnt` is HIGH at `count` = “000....000”.

Timing Diagrams

The following timing diagrams show various conditions for DW03_updn_ctr.

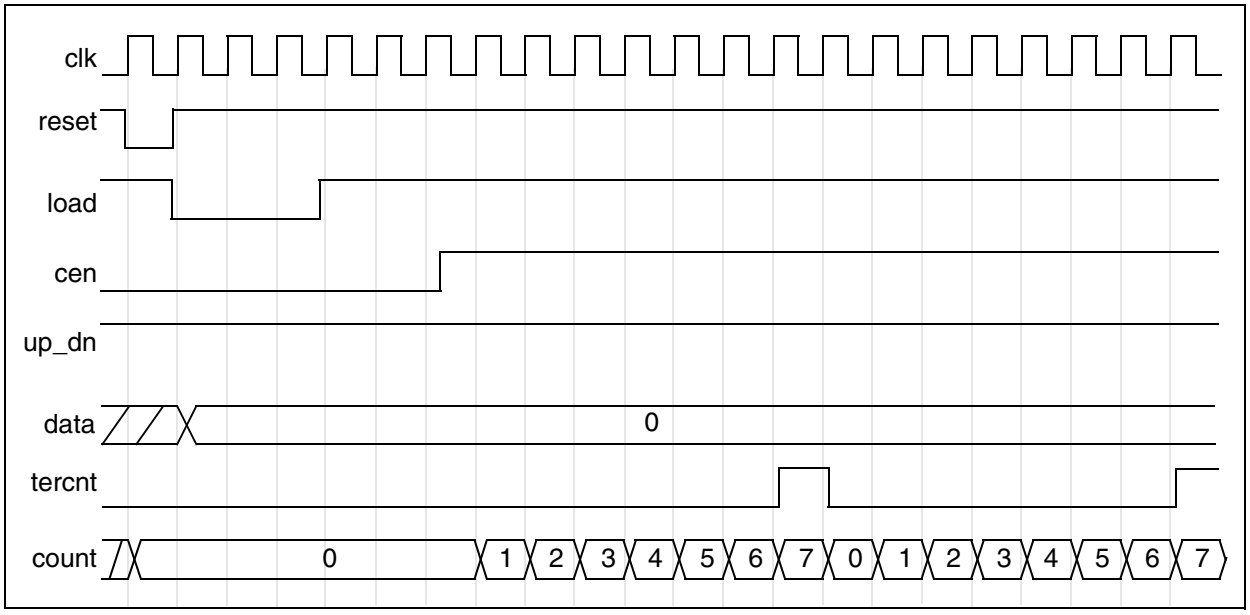


Figure 1: Functional Operation - 1

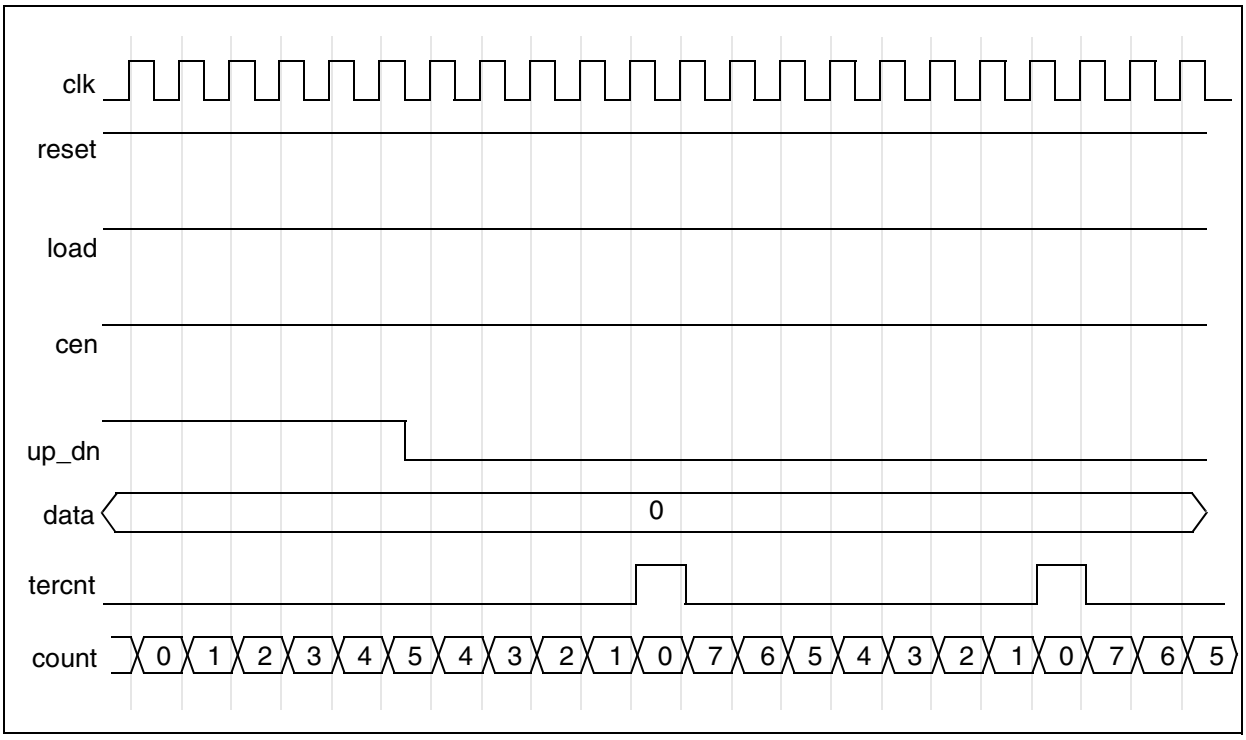


Figure 2: Functional Operation - 2

Related Topics

- Logic – Sequential Overview
- DesignWare Building Block IP Documentation Overview

HDL Usage Through Component Instantiation - VHDL

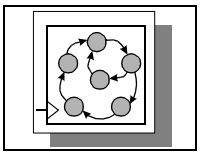
```
library IEEE,DWARE,DW03;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DW03.DW03_components.all;

entity DW03_updn_ctr_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_inst  : out std_logic_vector(inst_width-1 downto 0);
         tercnt_inst : out std_logic );
end DW03_updn_ctr_inst;

architecture inst of DW03_updn_ctr_inst is
begin

  -- Instance of DW03_updn_ctr
  U1 : DW03_updn_ctr
    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 => count_inst,
               tercnt => tercnt_inst );
end inst;

-- pragma translate_off
configuration DW03_updn_ctr_inst_cfg_inst of DW03_updn_ctr_inst is
  for inst
  end for; -- inst
end DW03_updn_ctr_inst_cfg_inst;
-- pragma translate_on
```



HDL Usage Through Component Instantiation - Verilog

```
module DW03_updn_ctr_inst( inst_data, 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 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_updn_ctr
  DW03_updn_ctr #(width)
    U1 ( .data(inst_data), .up_dn(inst_up_dn), .load(inst_load),
         .cen(inst_cen), .clk(inst_clk), .reset(inst_reset),
         .count(count_inst), .tercnt(tercnt_inst) );

endmodule
```