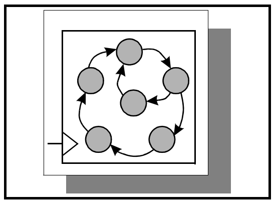


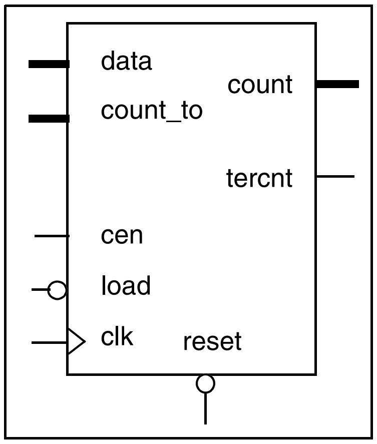
DW03_lfsr_dcnto

LFSR Counter with Dynamic Count-to Flag

Version, STAR and Download Information: IP Directory

Features and Benefits

- Dynamically programmable count-to value that indicates when the counter reaches a specified value
- High speed, area-efficient
- Asynchronous reset
- Terminal count



Description

DW03_lfsr_dcnto is a programmable word-length up counter with a dynamic count-to flag.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
data	*width* bit(s)	Input	Input data
count_to	*width* bit(s)	Input	Input count_to_bus
load	1 bit	Input	Input load data to counter, active low
cen	1 bit	Input	Input count enable
clk	1 bit	Input	Clock
reset	1 bit	Input	Asynchronous reset, active low
count	*width* bit(s)	Output	Output count bus
tercnt	1 bit	Output	Output terminal count

Table 1-2 Parameter Description

Parameter	Legal Range[a]	Description
width	1 to 50	Word length of counter

a. The upper bound of the legal range is a guideline to ensure reasonable compile times

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW03.DW03_LFSR_DCNTO_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW03_lfsr_dcnto_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW03_lfsr_dcnto.v	Verilog simulation model source code

Table 1-5 Counter Operation Truth Table

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

DW03_lfsr_dcnto implements a counter as LFSR (linear feedback shift register), which also acts as a pseudorandom counter constructed as primitive characteristic polynomials.

The shift register is fed back from two or more taps. The number of taps does not increase with a large counter *width*.

A LFSR counter runs faster than a binary counter in synchronous systems because the first bit is calculated and the remaining bits are shifted.

DW03_lfsr_dcnto can be used for built-in test circuitry in VLSI chips and used as a modulus counter. Refer to Figure 1-1 on page 3.

Counter Function

The `width` is a generic parameter with an integer value ranging from 1 to 50.

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_to` is an input of pseudorandom binary sequences that ranges from *width*–1 to 0. Refer to the Logic–Sequential Overview, Table 1 for a listing of the primitive polynomials.

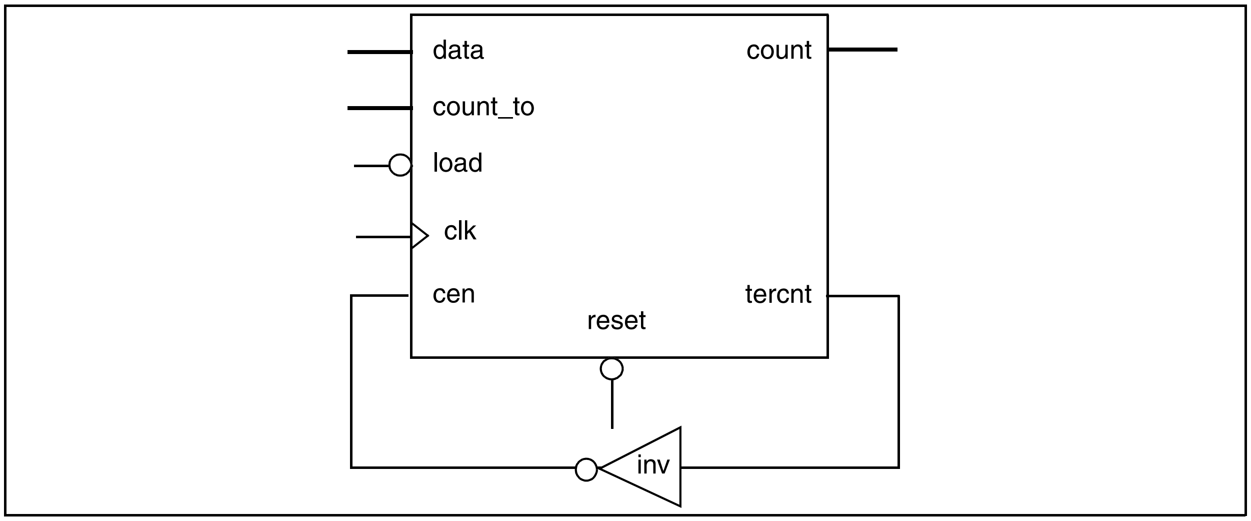
When the count enable pin, `cen`, is HIGH, the counter is active. When `cen` is LOW, the counter is disabled, and count remains at the same value.

The `reset`, active low, is an asynchronous reset signal. When `reset` is LOW, the counter output is “00...00”. When `reset` is HIGH, the counter operates normally.

The `count` is the output port of pseudorandom binary sequences, ranging from `width`-1 to 0. A value of $2^{width\text{-}2}$ (“11...11”) is an illegal state; therefore, the counter stops at “11...11”.

The `tercnt` is an output terminal count signal, active high. The `tercnt` output goes HIGH for one clock cycle to indicate the different number of clock cycles between starting count to `count_to` value.

Figure 1-1 Counter Application: “count_to”



Timing Diagrams

Figure 1-2 and Figure 1-3 show various timing conditions for DW03_lfsr_dcnto.

Figure 1-2 Functional Operation: load and count_enable sequence

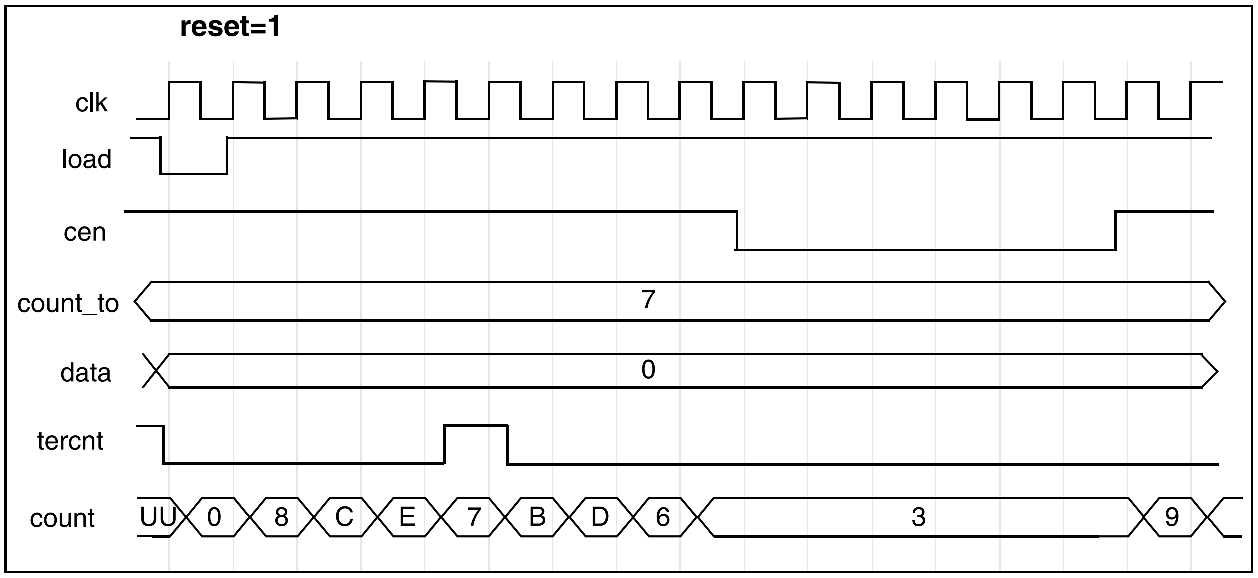
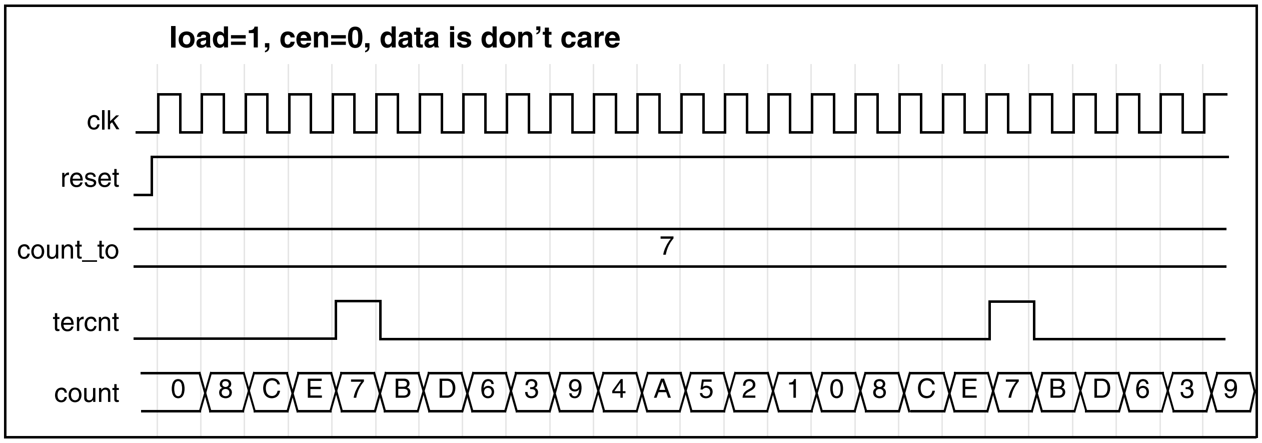


Figure 1-3 Functional Operation: reset and count_to sequence



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_lfsr_dcnto_inst is
  generic (inst_width : INTEGER := 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_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_lfsr_dcnto_inst;

architecture inst of DW03_lfsr_dcnto_inst is
begin

  -- Instance of DW03_lfsr_dcnto
  U1 : DW03_lfsr_dcnto
    generic map ( width => inst_width )
    port map ( data => inst_data, count_to => inst_count_to,
               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_lfsr_dcnto_inst_cfg_inst of DW03_lfsr_dcnto_inst is
  for inst
  end for; -- inst
end DW03_lfsr_dcnto_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW03_lfsr_dcnto_inst( inst_data, inst_count_to,
                             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_load;
  input inst_cen;
  input inst_clk;
  input inst_reset;
  output [width-1 : 0] count_inst;
  output tercnt_inst;

  // Instance of DW03_lfsr_dcnto
  DW03_lfsr_dcnto #(width)
    U1 ( .data(inst_data), .count_to(inst_count_to),
         .load(inst_load), .cen(inst_cen), .clk(inst_clk),
         .reset(inst_reset), .count(count_inst), .tercnt(tercnt_inst) );

endmodule
```