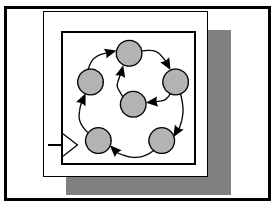
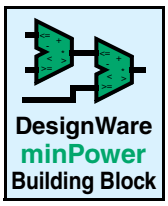




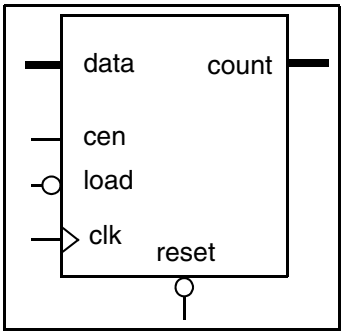
DW03_lfsr_load

LFSR Counter with Loadable Input

Version, STAR and Download Information: IP Directory

Features and Benefits

- Parameterized word length
- Loadable counter registers
- High speed, area-efficient
- Asynchronous reset
- Terminal count



Description

DW03_lfsr_load is a parameterized word-length up counter with loadable data input. DW03_lfsr_load implements a counter as LFSR (linear feedback shift register) which also acts as a pseudo-random counter constructed as primitive characteristic polynomials.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
data	*width* bit(s)	Input	Input data
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

Table 1-2 Parameter Description

Parameter	Values[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_LOAD_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW03_lfsr_load_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW03_lfsr_load.v	Verilog simulation model source code

Table 1-5 Counter Operation Truth Table

reset	load	cen	Operation
0	X	X	Reset
1	0	1	data is XORd with count
1	X	0	Standby
1	1	1	Count up

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

An 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_load can be used in built-in test circuitry for VLSI chips and multiple-input signature registers.

Counter Function

The data input bus, `data`, ranges from *width*–1 to 0.

When the input signal, `load`, is LOW, `data` is exclusive ORd with `count`. When `load` is HIGH, the counter operates as a normal LFSR up counter.

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` signal is an asynchronous reset that is active low. 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. Refer to the Sequential Overview section, Table 1 for a listing of the primitive polynomials. A value of $2^{width-2}$ ("11...11") is an illegal state; therefore, the counter stops at "11...11".

Timing Diagrams

Figure 1-1 and Figure 1-2 show various timing conditions for DW03_lfsr_load.

Figure 1-1 Functional Operation: reset, count and count_enable sequence

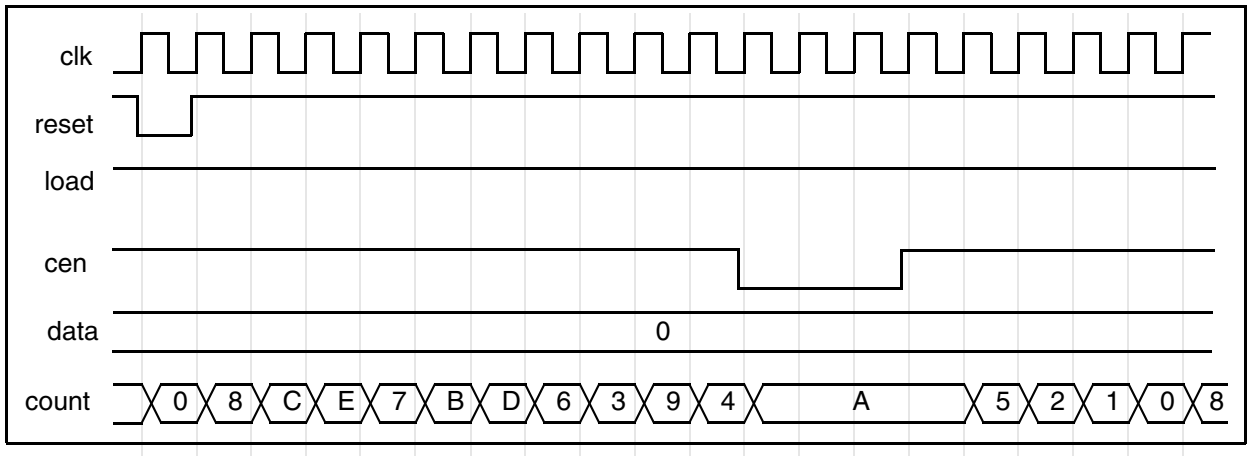
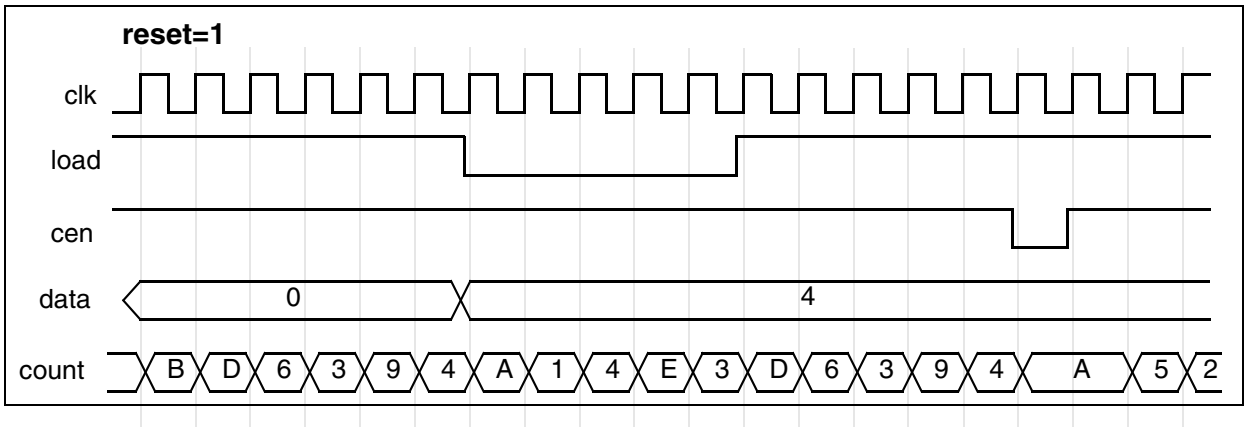


Figure 1-2 Functional Operation: load and count_enable sequence



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_load_inst is
  generic ( inst_width : INTEGER := 8 );
  port ( inst_data  : 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) );
end DW03_lfsr_load_inst;

architecture inst of DW03_lfsr_load_inst is
begin

  -- Instance of DW03_lfsr_load
  U1 : DW03_lfsr_load
    generic map ( width => inst_width )
    port map ( data => inst_data, load => inst_load, cen => inst_cen,
               clk => inst_clk, reset => inst_reset, count => count_inst );
end inst;

-- pragma translate_off
configuration DW03_lfsr_load_inst_cfg_inst of DW03_lfsr_load_inst is
  for inst
  end for; -- inst
end DW03_lfsr_load_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW03_lfsr_load_inst( inst_data, inst_load, inst_cen,
                            inst_clk, inst_reset, count_inst );

  parameter width = 8;

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

  // Instance of DW03_lfsr_load
  DW03_lfsr_load #(width)
    U1 ( .data(inst_data), .load(inst_load), .cen(inst_cen), .clk(inst_clk),
         .reset(inst_reset), .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