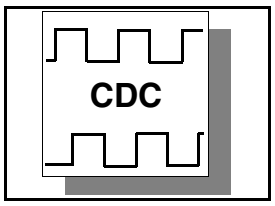
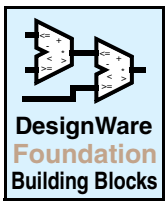


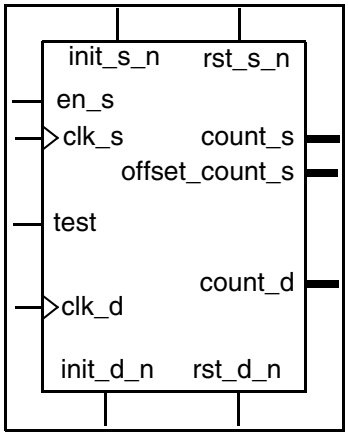


DW_gray_sync

Gray Coded Synchronizer

Version, STAR and Download Information: IP Directory

Features and Benefits



- Interface between dual asynchronous clock domains
- Parameterized counter width
- Parameterized number of synchronizing stages
- Parameterized test feature
- Outputs to source can be configured as registered or not
- Output to destination domain can be configured as registered or not
- Model missampling of data on source clock domain
- Parameterized pipelining source domain results to destination domain
- Provides minPower benefits with the DesignWare-LP license

Description

The DW_gray_sync includes a binary counter in the source domain whose value is recoded as a binary-reflected-Gray code. The recoded count value is synchronized to the destination domain and then decoded from binary-reflected-Gray back to binary. The binary count value and an offset version (which is only useful for non integer power of two sequences) are both available in the clk_s domain as count_s and `offset_count_s`. This synchronizer is used within other synchronizers to pass incrementing pointers across domains.

A unique built-in verification feature allows the designer to turn on a random sampling error mechanism that models skew between bits of the internal binary-reflected-Gray coded counter bus derived from the source domain (see section “Simulation Methodology” for more details). This facility provides an opportunity for determining system robustness during the early development phases and without having to develop special test stimulus.

Table 1-1 Pin Descriptions

Pin Name	Width	Direction	Function
clk_s	1	Input	Source Domain clock source
rst_s_n	1	Input	Source Domain asynchronous reset (active low)
init_s_n	1	Input	Source Domain synchronous reset (active low)
en_s	1	Input	Source Domain counter advance initiation
count_s	width	Output	Source Domain counter value

Table 1-1 Pin Descriptions (Continued)

Pin Name	Width	Direction	Function
offset_count_s	width	Output	Source Domain offset counter value
clk_d	1	Input	Destination Domain clock source
rst_d_n	1	Input	Destination Domain asynchronous reset (active low)
init_d_n	1	Input	Destination Domain synchronous reset (active low)
count_d	width	Output	Destination Domain counter value (synchronized version of `count_s`)
test	1	Input	Scan test mode select

Table 1-2 Parameter Description

Parameter	Values	Description
width	1 to 1024 Default: 8	Vector width of input `data_s` and output `data_d`
offset	0 to $(2^{(width-1)}) - 1$	Offset for non integer power of 2 counter value Default: 0
reg_count_d	0 or 1 Default: 1	Register `count_d` output 0 = `count_d` not registered 1 = `count_d` registered
f_sync_type	0 to 4 Default: 2	Forward synchronization type Defines type and number of synchronizing stages: 0 = single clock design, no synchronizing stages implemented 1 = 2-stage synchronization with 1st stage negative-edge capturing and 2nd stage positive-edge capturing 2 = 2-stage synchronization with both stages positive-edge capturing 3 = 3-stage synchronization with all stages positive-edge capturing 4 = 4-stage synchronization with all stages positive-edge capturing
tst_mode	0 to 2 Default: 0	Test mode 0 = no latch is inserted for scan testing 1 = insert negative-edge capturing flip-flip on `data_s` input vector when `test` input is asserted. 2 = insert hold latch using active-low latch.

Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
verif_en*	0 to 4 Default: 2	Verification enable control 0 = no sampling errors inserted 1 = sampling errors are randomly inserted with 0 or up to 1 destination clock cycle delays 2 = sampling errors are randomly inserted with 0, 0.5, 1, or 1.5 destination clock cycle delays 3 = sampling errors are randomly inserted with 0, 1, 2, or 3 destination clock cycle delays 4 = sampling errors are randomly inserted with 0 or up to 0.5 destination clock cycle delays * For more information about *verif_en*, see the Simulation Methodology section.
pipe_delay	0 to 2 Default: 0	Pipeline Binary to Gray Code results 0 = no pipelining other than re-timing register 1 = one additional pipeline stage 2 = two additional pipeline stages
reg_count_s	0 to 1 Default: 1	Register `count_s` output 0 = `count_s` not registered 1 = `count_s` registered
reg_offset_count_s	0 to 1 Default: 1	Register `offset_count_s` output 0 = `offset_count_s` not registered 1 = `offset_count_s` registered

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW03.DW_GRAY_SYNC _CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW_gray_sync _sim.vhd	VHDL simulation model source code (modeling RTL) - no missampling
dw/dw03/src/DW_gray_sync _sim_ms.vhd	VHDL simulation model source code (modeling RTL) - missampling
dw/sim_ver/DW_gray_sync.v	Verilog simulation model source code

Simulation Methodology

For simulation, there are two methods available. One method is to utilize the simulation models as they emulate the RTL model. The other method is to enable modeling of random skew between bits on the data_s bus of the source domain by the destination domain (called "missampling" in this discussion). When using the simulation models to behave only as the RTL model, no special configuration is required. When using the simulation models to enable missampling, unique considerations must be made between Verilog and VHDL environments.

- For Verilog simulation enabling missampling, a preprocessing variable named DW_MODEL_MISSAMPLES must be defined as follows:

  ```
  `define DW_MODEL_MISSAMPLES
  ```

  Once `DW_MODEL_MISSAMPLES is defined, the value of the *verif_en* parameter comes into play and configures the simulation model as described by Table 2. Note: If `DW_MODEL_MISSAMPLES is not defined, the Verilog simulation model behaves as if *verif_en* was set to '0'.

- For VHDL simulation enabling missampling, an alternative simulation architecture is provided. This architecture is named sim_ms. The parameter *verif_en* has meaning only when utilizing the sim_ms; that is, when binding the "sim" simulation architecture the *verif_en* value is ignored and the model effectively behaves as though *verif_en* is set to 0. For an example on using each architecture, refer to "HDL Usage Through Component Instantiation - VHDL" on page 9.

Block Diagram

Figure 1-1 DW_gray_sync Basic Block Diagram

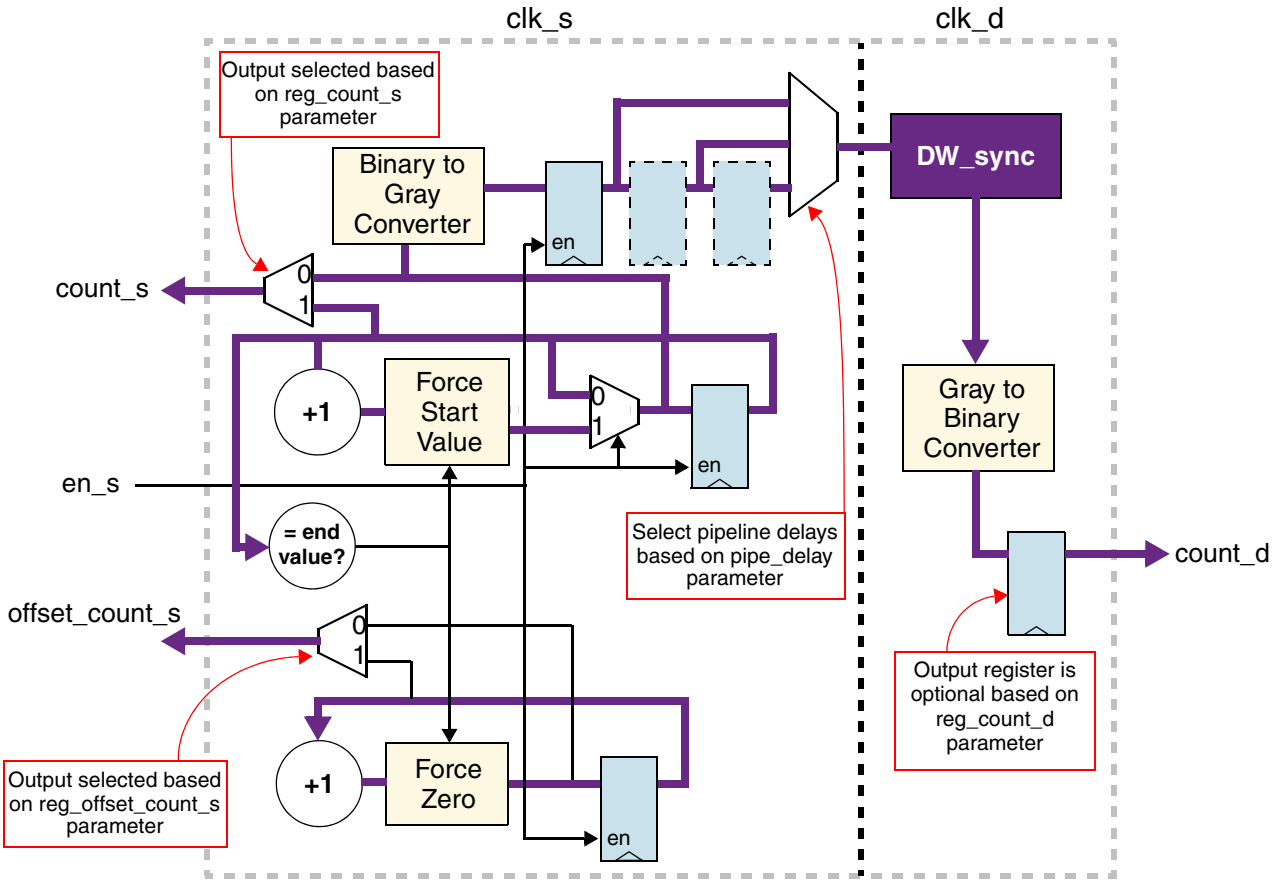


Table 1-5 Reflected Gray Code Sequencing Example

Reflected Binary Gray Code Sequencing for a 4-bit Domain							
Binary Sequence	**Full Gray Sequence**		**14-State Gray Sequence**	**Binary Offset Sequence**		**10-State Gray Sequence**	**Binary Offset Sequence**
0000	0000 (start)		—	—		—	—
0001	0001		0001 (start)	0000		—	—
0010	0011		0011	0001		—	—
0011	0010		0010	0010		0010 (start)	0000
0100	0110		0110	0011		0110	0001
0101	0111		0111	0100		0111	0010
0110	0101		0101	0101		0101	0011
0111	0100		0100	0110		0100	0100

Table 1-5 Reflected Gray Code Sequencing Example (Continued)

Reflected Binary Gray Code Sequencing for a 4-bit Domain							
Binary Sequence	**Full Gray Sequence**		**14-State Gray Sequence**	**Binary Offset Sequence**		**10-State Gray Sequence**	**Binary Offset Sequence**
Axis of Symmetry							
1000	1100		1100	0111		1100	0101
1001	1101		1101	1000		1101	0110
1010	1111		1111	1001		1111	0111
1011	1110		1110	1010		1110	1000
1100	1010		1010	1011		1010 (end)	1001
1101	1011		1011	1100		—	—
1110	1001		1001 (end)	1101		—	—
1111	1000 (end)		—	—		—	—

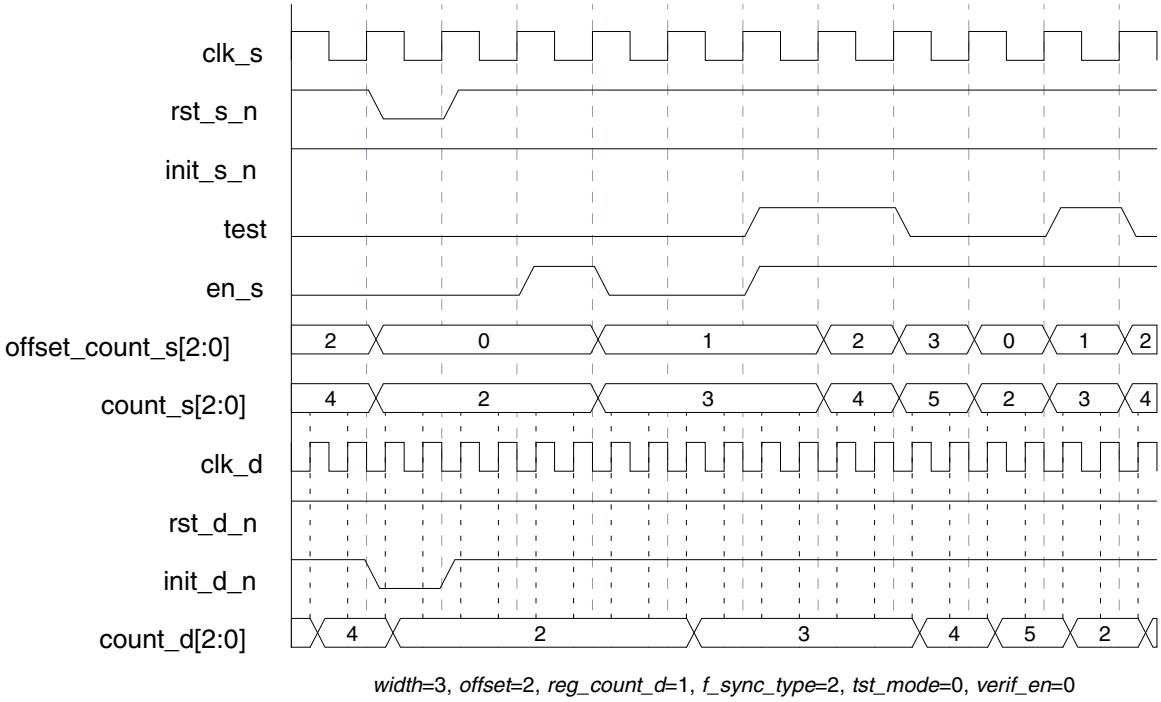
Reset Considerations

It is recommended that both domains be reset at or around the same time. If the source domain interface is reset and the destination domain is not, the destination domain `clk_d` value will be invalid relative to the source domain `clk_s` for at least *f_sync_type* number of destination domain clock cycles depending on the *reg_count_d* setting. If *reg_count_d* is 1, the total number of destination clock cycles before `clk_d` reflects the value on `clk_s` after the source domain reset is *f_sync_type* + 1.

Timing Diagrams

Figure 1-2 depicts the fundamental operation of advancing the counters after resetting both clock domains. Notice that the `offset` parameter is set to 2, which sets the `count_s` initial value upon reset to 2 and advances to a maximum value of 5. The `count_d` result is three `clk_d` cycles after the change in `count_s`, since *f_sync_type* is set to 2 and *reg_count_d* is set to 1. The other parameter settings are: *width* is 3, *offset* is 2, *tst_mode* is 0, *verif_en* is 0, *pipe_delay* is 0, *reg_count_s* is 1, and *reg_offset_count_s* is 1.

Figure 1-2 Advancing Counters Including Reset of Both Domains

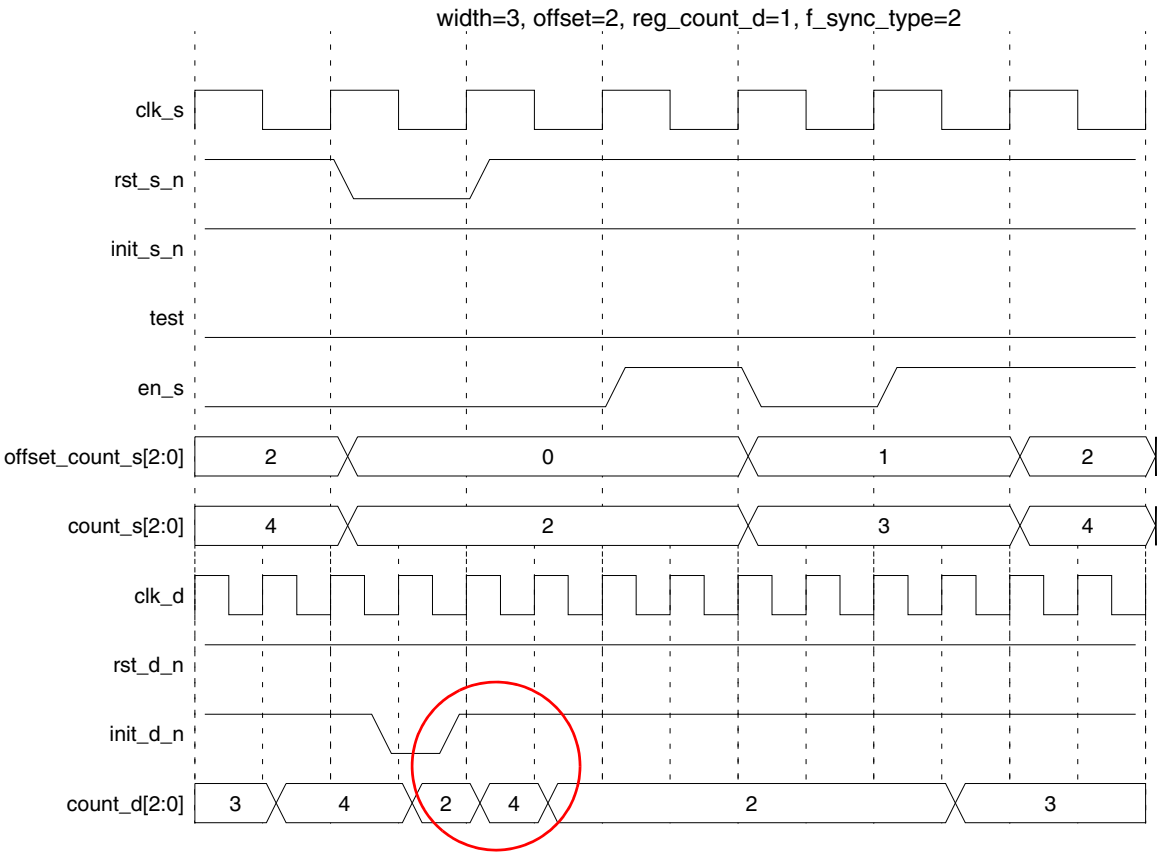


width=3, *offset*=2, *reg_count_d*=1, *f_sync_type*=2, *tst_mode*=0, *verif_en*=0

Beginning in Release F-2011.09, the underlying DW_sync component in the DW_gray_sync will not have its 'init_d_n' connected. Instead, the DW_sync will have its 'init_d_n' deasserted. As a by-product of this, the resetting of 'count_d' is based on the cleared state of the gray code entering and propagating through the DW_sync. Figure 1-3 shows an example of this. In this case, the same scenario is presented as in Figure 1-2 but here 'init_d_n' is only a single-cycle of 'clk_d'. Here it can been seen that the 'count_d[2:0]' goes to its reset state of '2' after 'init_d_n' is asserted, but after 'init_d_n' is de-asserted, 'count_d[2:0]' goes to '4' for one 'clk_d' cycle before going back to '2' on the next cycle (see area circled in red). This exhibits the fact that the reset value going into the DW_sync has to make its way through the DW_sync and into the destination domain and 'init_d_n' was not held asserted long enough to mask the non-reset value. In Figure 1-2, the with 'init_d_n' being asserted 2 cycles of 'clk_d', it was long enough to prevent sampling of the non-reset value exiting DW_sync and, hence, 'count_d[2:0]' remained at '2'.

This behavior, in fact, would also be seen should the source domain not get reset prior to the destination domain reset (see the Reset Considerations section above for more information).

Figure 1-3 Invalid 'count_d' Result During Reset



Related Topics

- Memory – Registers Overview
- DesignWare Building Block IP Documentation Overview

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp.all;

entity DW_gray_sync_inst is
      generic (
        inst_width : INTEGER := 8;
        inst_offset : INTEGER := 0;
        inst_reg_count_d : INTEGER := 1;
        inst_f_sync_type : INTEGER := 2;
        inst_tst_mode : INTEGER := 0;
        inst_verif_en : INTEGER := 2;
            inst_pipe_delay : NATURAL:= 0;
            inst_reg_count_s : NATURAL := 1;
            inst_reg_offset_count_s : NATURAL := 1
        );
      port (
        inst_clk_s : in std_logic;
        inst_rst_s_n : in std_logic;
        inst_init_s_n : in std_logic;
        inst_en_s : in std_logic;
        count_s_inst : out std_logic_vector(inst_width-1 downto 0);
        offset_count_s_inst : out std_logic_vector(inst_width-1 downto 0);

        inst_clk_d : in std_logic;
        inst_rst_d_n : in std_logic;
        inst_init_d_n : in std_logic;
        count_d_inst : out std_logic_vector(inst_width-1 downto 0);

        inst_test : in std_logic
        );
    end DW_gray_sync_inst;


architecture inst of DW_gray_sync_inst is
begin

    -- Instance of DW_gray_sync
    U1 : DW_gray_sync
    generic map ( width => inst_width, offset => inst_offset,
            reg_count_d => inst_reg_count_d, f_sync_type => inst_f_sync_type,
            tst_mode => inst_tst_mode, verif_en => inst_verif_en,
            pipe_delay => inst_pipe_delay, reg_count_s => inst_reg_count_s,
            reg_offset_count_s => inst_reg_offset_count_s )
    port map ( clk_s => inst_clk_s, rst_s_n => inst_rst_s_n,
     init_s_n => inst_init_s_n, en_s => inst_en_s,
     count_s => count_s_inst, offset_count_s => offset_count_s_inst,
```

```
      clk_d => inst_clk_d, rst_d_n => inst_rst_d_n, init_d_n => inst_init_d_n,
      count_d => count_d_inst, test => inst_test );


end inst;

-- Configuration for use with a VHDL simulator
-- pragma translate_off
library DW03;
configuration DW_gray_sync_inst_cfg_inst of DW_gray_sync_inst is
  for inst
    -- NOTE: If desiring to model missampling, uncomment the following
    -- line.  Doing so, however, will cause inconsequential errors
    -- when analyzing or reading this configuration before synthesis.
    -- for U1 : DW_gray_sync use configuration DW03.DW_gray_sync_cfg_sim_ms;  end for;
  end for; -- inst
end DW_gray_sync_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_gray_sync_inst( inst_clk_s, inst_rst_s_n, inst_init_s_n, inst_en_s,
                  count_s_inst, offset_count_s_inst, inst_clk_d, inst_rst_d_n,
                  inst_init_d_n, count_d_inst, inst_test );

parameter width = 8;
parameter offset = 0;
parameter reg_count_d = 1;
parameter f_sync_type = 2;
parameter tst_mode = 0;
parameter verif_en = 2;
parameter pipe_delay = 0;
parameter reg_count_s = 1;
parameter reg_offset_count_s = 1;


input inst_clk_s;
input inst_rst_s_n;
input inst_init_s_n;
input inst_en_s;
output [width-1 : 0] count_s_inst;
output [width-1 : 0] offset_count_s_inst;

input inst_clk_d;
input inst_rst_d_n;
input inst_init_d_n;
output [width-1 : 0] count_d_inst;

input inst_test;

    // Instance of DW_gray_sync
    DW_gray_sync #(width, offset, reg_count_d, f_sync_type, tst_mode, verif_en,
pipe_delay, reg_count_s, reg_offset_count_s)
         U1 ( .clk_s(inst_clk_s), .rst_s_n(inst_rst_s_n), .init_s_n(inst_init_s_n),
              .en_s(inst_en_s), .count_s(count_s_inst),
.offset_count_s(offset_count_s_inst),
              .clk_d(inst_clk_d), .rst_d_n(inst_rst_d_n), .init_d_n(inst_init_d_n),
              .count_d(count_d_inst), .test(inst_test) );

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