



# DesignWare Datapath and Building Block IP

## Quick Reference

---

*DesignWare Datapath and Building Block IP*

## Copyright Notice and Proprietary Information Notice

© 2015 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

### 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

Synopsis and certain Synopsis product names are trademarks of Synopsis, as set forth at <http://www.synopsys.com/Company/Pages/Trademarks.aspx>.

All other product or company names may be trademarks of their respective owners.

Synopsis, Inc.  
690 E. Middlefield Road  
Mountain View, CA 94043  
[www.synopsys.com](http://www.synopsys.com)



# Contents

---

Preface .....	12
About This Manual .....	12
Manual Overview .....	12
Synopsys Common Licensing (SCL) .....	12
Getting Help .....	12
Additional Information .....	13
Comments? .....	13
Chapter DesignWare Overview .....	14
Related DesignWare IP .....	14
Chapter DesignWare Overview .....	15
Chapter DesignWare Building Block Synthesizable IP .....	16
Building Block IP for DC QuickStart .....	17
Setting Up DesignWare Building Block IP in DC-TCL .....	17
Accessing DesignWare Building Block IP in DC .....	17
Synthesizing DesignWare Building Block IP in DC .....	17
Simulating DesignWare Building Block IP .....	18
Chapter Application Specific - Control Logic .....	19
DW_arb_2t	
Two-Tier Arbiter with Dynamic/Fair-Among-Equal Scheme .....	20
DW_arb_dp	
Arbiter with Dynamic Priority Scheme .....	22
DW_arb_fcfs	
Arbiter with First-Come-First-Served Priority Scheme .....	24
DW_arb_rr	
Arbiter with Round Robin Priority Scheme .....	26
DW_arb_sp	
Arbiter with Static Priority Scheme .....	28
Chapter Datapath Generator Overview .....	30
Chapter Datapath - Arithmetic Overview .....	31
DW01_absval	
Absolute Value .....	32
DW01_add	
Adder .....	33
DW01_addsub	
Adder-Subtractor .....	34
DW_addsub_dx	
Duplex Adder/Subtractor with Saturation and Rounding .....	36

DW01_ash	
Arithmetic Shifter .....	38
DW_bin2gray	
Binary to Gray Converter .....	39
DW01_bsh	
Barrel Shifter .....	40
DW01_cmp2	
DW01_cmp6	
6-Function Comparator .....	42
DW_cmp_dx	
DW_cntr_gray	
Gray Code Counter .....	45
DW01_csa	
Carry Save Adder .....	46
DW01_dec	
Decrementer .....	47
DW_div	
Combinational Divider .....	48
DW_div_pipe	
Stallable Pipelined Divider .....	50
DW_div_sat	
Combinational Divider with Saturation .....	52
DW_exp2	
Base-2 Exponential .....	53
DW_gray2bin	
Gray-to-Binary Converter .....	54
DW01_inc	
Incrementer .....	55
DW01_incdec	
Incrementer-Decrementer .....	56
DW_inc_gray	
Gray Incrementer .....	57
DW_inv_sqrt	
Reciprocal of Square-Root .....	58
DW_lbsh	
Barrel Shifter with Preferred Left Direction .....	59
DW_ln	
Natural Logarithm .....	60
DW_log2	
Base 2 Logarithm .....	61
DW02_mac	
Multiplier-Accumulator .....	62
DW_minmax	
Minimum/Maximum Value .....	63
DW02_mult	
Multiplier .....	64
DW02_multp	
Partial Product Multiplier .....	66
DW02_mult_2_stage	
Two-Stage Pipelined Multiplier .....	68
DW02_mult_3_stage	
Three-Stage Pipelined Multiplier .....	69
DW02_mult_4_stage	
Four-Stage Pipelined Multiplier .....	70

DW02_mult_5_stage	Five-Stage Pipelined Multiplier .....	71
DW02_mult_6_stage	Six-Stage Pipelined Multiplier .....	72
DW_mult_dx	Duplex Multiplier .....	73
DW_mult_pipe	Stallable Pipelined multiplier .....	74
DW_norm		
DW_norm_rnd	Normalization and rounding .....	78
DW_piped_mac	Pipelined Multiplier-Accumulator .....	80
DW02_prod_sum	Generalized Sum of Products .....	82
DW02_prod_sum1	Multiplier-Adder .....	83
DW_prod_sum_pipe	Stallable Pipelined Generalized Sum of Products .....	84
DW_rash	Arithmetic Shifter with Preferred Right Direction .....	86
DW_rbsh	Barrel Shifter with Preferred Right Direction .....	87
DW01_satrnd		
DW_shifter	Combined Arithmetic and Barrel Shifter .....	89
DW_sla	Arithmetic Shifter with Preferred Left Direction (VHDL Style) .....	91
DW_sra	Arithmetic Shifter with Preferred Right Direction (VHDL Style) .....	92
DW_square	Integer Squarer .....	93
DW_squarep	Partial Product Integer Squarer .....	94
DW_sqrt	Combinational Square Root .....	96
DW_sqrt_pipe		
DW01_sub	Subtractor .....	98
DW02_sum	Vector Adder .....	99
DW02_tree	Wallace Tree Compressor .....	100
Chapter Datapath - Floating Point Overview	.....	101
DW_fp_add		
DW_fp_add	Floating Point Adder .....	102
DW_fp_addsub	Floating Point Adder/Subtractor .....	103
DW_fp_cmp		
DW_fp_div	Floating Point Divider .....	107
DW_fp_div_seq	Floating Point Sequential Divider .....	109



DW_fp_dp2 2-Term Floating Point Dot-product .....	110
DW_fp_dp3 3-Term Floating Point Dot-product .....	112
DW_fp_dp4 4-Term Floating Point Dot-product .....	114
DW_fp_exp Floating Point Exponential .....	116
DW_fp_exp2 Floating Point Base 2 Exponential .....	117
DW_fp_flt2i Floating Point-to-Integer Converter .....	118
DW_fp_i2flt Integer-to-Floating Point Converter .....	119
DW_fp_invsqrt Floating Point Reciprocal of Square Root .....	120
DW_fp_ln Floating Point Natural Logarithm .....	121
DW_fp_log2 Floating Point Base-2 Logarithm .....	122
DW_fp_mac Floating Point Multiply and Add .....	123
DW_fp_mult Floating Point Multiplier .....	124
DW_fp_recip Floating Point Reciprocal .....	125
DW_fp_sincos Floating Point Sine and Cosine .....	126
DW_fp_sqrt Floating Point Square Root .....	127
DW_fp_square Floating Point Square .....	128
DW_fp_sub Floating Point Adder .....	129
DW_fp_sum3 3-Input Floating Point Adder .....	130
DW_fp_sum4 4-Input Floating Point Adder .....	131
 Chapter Datapath – Sequential Overview .....	132
DW_div_seq Sequential Divider .....	133
DW_mult_seq Sequential Multiplier .....	135
DW_sqrt_seq Sequential Square Root .....	137
 Chapter Datapath – Trigonometric Overview .....	139
DW_sincos Combinational Sine - Cosine .....	140
 Chapter Datapath Functions Overview .....	141
DWF_dp_absval Absolute Value Function .....	142

DWF_dp_blend Graphics Alpha Blend Functions .....	143
DWF_dp_count_ones Count Ones .....	144
DWF_dp_mult_comb Combined Unsigned/Signed Multiply .....	145
DWF_dp_mult_comb_sat Combined Unsigned/Signed Multiply and Saturate .....	146
DWF_dp_mult_ovldet Multiply and Overflow Detection .....	147
DWF_dp_mult_sat Multiply and Saturate Functions .....	148
DWF_dp_rnd Arithmetic Rounding Functions .....	149
DWF_dp_rndsat Arithmetic Rounding and Saturation Functions .....	150
DWF_dp_sat Arithmetic Saturation Functions .....	151
DWF_dp_sign_select Sign Selection / Conditional Two's Complement Functions .....	152
DWF_dp_simd_addc SIMD Add with Carries .....	153
DWF_dp_simd_add SIMD Add .....	154
DWF_dp_simd_mult SIMD Multiply .....	155
 Chapter Data Integrity .....	156
DW_crc_p Universal Parallel (Combinational) CRC Generator/Checker .....	157
DW_crc_s Universal Synchronous (Clocked) CRC Generator/Checker .....	159
DW_ecc Error Checking and Correction .....	161
DW04_par_gen Parity Generator and Checker .....	163
 Chapter Data Integrity - Coding Group Overview .....	164
DW_8b10b_dec 8b10b Decoder .....	165
DW_8b10b_enc 8b10b Encoder .....	167
DW_8b10b_unbal 8b10b Coding Balance Predictor .....	169
 Chapter Digital Signal Processing (DSP) .....	170
DW_dct_2d Two Dimensional Discrete Cosine Transform .....	171
DW_decode_en Binary Decoder with Enable .....	174
DW_fir High-Speed Digital FIR Filter .....	175
DW_fir_seq Sequential Digital FIR Filter .....	177

DW_iir_dc	High-Speed Digital IIR Filter with Dynamic Coefficients .....	179
DW_iir_sc	High-Speed Digital IIR Filter with Static Coefficients .....	181
DW_thermdec	Binary Thermometer Decoder with Enable .....	183
<b>Chapter Logic - Combinational Overview</b>	.....	184
DW01_binenc	Binary Encoder .....	185
DW01_decode	Decoder .....	186
DW_lod	Leading One's Detector .....	187
DW_lsd	Leading Signs Detector .....	188
DW_lza	Leading Zero's Anticipator .....	189
DW_lzd	Leading Zero's Detector .....	190
DW01_mux_any	Universal Multiplexer .....	191
DW_pricod	Priority Coder .....	192
DW01_prienc	Priority Encoder .....	193
<b>Chapter Logic - Sequential Overview</b>	.....	194
DW03_bictr_dcnto	Up/Down Binary Counter with Dynamic Count-to Flag .....	195
DW03_bictr_scnto	Up/Down Binary Counter with Static Count-to Flag .....	196
DW03_bictr_decode	Up/Down Binary Counter with Output Decode .....	197
DW_dpll_sd	Digital Phase Locked Loop .....	198
DW03_lfsr_dcnto	LFSR Counter with Dynamic Count-to Flag .....	200
DW03_lfsr_scnto	LFSR Counter with Static Count-to Flag .....	201
DW03_lfsr_load	LFSR Counter with Loadable Input .....	202
DW03_lfsr_updn	LFSR Up/Down Counter .....	203
DW03_updn_ctr	Up/Down Counter .....	204
<b>Chapter Memory - FIFO Overview</b>	.....	205
DW_asymdata_inbuf	Asymmetric Data Input Buffer .....	206
DW_asymdata_outbuf	Asymmetric Data Output Buffer .....	208
DW_asymfifo_s1_df	Asymmetric I/O Synchronous (Single Clock) FIFO with Dynamic Flag .....	210

DW_asymfifo_s1_sf Asymmetric I/O Synchronous (Single Clock) FIFO with Static Flags .....	213
DW_asymfifo_s2_sf Asymmetric Synchronous (Dual Clock) FIFO with Static Flags .....	216
DW_fifo_2c_df Dual Independent Clock FIFO .....	219
DW_fifo_s1_df Synchronous (Single Clock) FIFO with Dynamic Flags .....	222
DW_fifo_s1_sf Synchronous (Single Clock) FIFO with Static Flags .....	224
DW_fifo_s2_sf Synchronous (Dual-Clock) FIFO with Static Flags .....	226
DW_asymfifoctl_s1_df Asymmetric Synchronous (1-Clock) FIFO Controller with Dynamic Flags .....	229
DW_asymfifoctl_s1_sf Asymmetric Synchronous (1-Clock) FIFO Controller with Static Flags .....	232
DW_asymfifoctl_s2_sf Asymmetric Synchronous (Dual-Clock) FIFO Controller with Static Flags .....	235
DW_fifocntl_2c_df Dual Clock FIFO Controller with Dynamic Flags .....	239
DW_fifocntl_s1_df Synchronous (Single Clock) FIFO Controller with Dynamic Flag .....	244
DW_fifocntl_s1_sf Synchronous (Single Clock) FIFO Controller with Static Flags .....	246
DW_fifocntl_s2_sf Synchronous (Dual Clock) FIFO Controller with Static Flags .....	248
<b>Chapter Memory - Registers</b> .....	251
DW03_pipe_reg Pipeline Register .....	252
DW_pl_reg Pipeline Register .....	253
DW03_reg_s_pl Register with Synchronous Enable Reset .....	255
DW04_shad_reg Shadow and Multibit Register .....	256
DW03_shftreg Shift Register .....	258
<b>Chapter Memory - Synchronous RAMs</b> .....	259
DW_ram_r_w_s_dff Synchronous Write-Port, Asynchronous Read-Port RAM (Flip-Flop-Based) .....	260
DW_ram_r_w_s_lat Synchronous Write Port, Asynchronous Read Port RAM (Latch-Based) .....	261
DW_ram_2r_w_s_dff Synchronous Write Port, Asynchronous Dual Read Port RAM (FF-Based) .....	262
DW_ram_2r_2w_s_dff Synchronous Dual Write Port, Async. Dual Read Port RAM (FF-Based) .....	264
DW_ram_2r_w_s_lat Synchronous Write Port, Asynchronous Dual Read Port RAM (Latch-Based) .....	266
DW_ram_rw_s_dff Synchronous Single Port Read/Write RAM (Flip-Flop-Based) .....	267
DW_ram_rw_s_lat Synchronous Single Port Read/Write RAM (Latch-Based) .....	268

Chapter Memory - Asynchronous RAMs .....	269
DW_ram_r_w_a_dff	
Asynchronous Dual Port RAM (Flip-Flop-Based) .....	270
DW_ram_r_w_a_lat	
Asynchronous Dual Port RAM (Latch-Based) .....	271
DW_ram_2r_w_a_dff	
Write Port, Dual Read Port RAM (Flip-Flop-Based) .....	272
DW_ram_2r_w_a_lat	
Write Port, Dual Read Port RAM (Latch-Based) .....	273
DW_ram_rw_a_dff	
Asynchronous Single Port RAM (Flip-Flop-Based) .....	274
DW_ram_rw_a_lat	
Asynchronous Single-Port RAM (Latch-Based) .....	275
Chapter Memory - Stacks .....	276
DW_stack	
Synchronous (Single-Clock) Stack .....	277
DW_stackctl	
Synchronous (Single-Clock) Stack Controller .....	279
Chapter Interface - Clock Domain Crossing .....	281
DW_data_qsync_hl	
Quasi-Synchronous Data Interface for H-to-L Frequency Clocks .....	282
DW_data_qsync_lh	
Quasi-Synchronous Data Interface for L-to-H Frequency Clocks .....	284
DW_data_sync	
DW_data_sync_na	
Data Bus Synchronizer without Acknowledge .....	288
DW_data_sync_1c	
DW_gray_sync	
Gray Coded Synchronizer .....	292
DW_pulse_sync	
Dual Clock Pulse Synchronizer .....	294
DW_pulseack_sync	
Pulse Synchronizer with Acknowledge .....	296
DW_reset_sync	
Reset Sequence Synchronizer .....	298
DW_stream_sync	
Data Stream Synchronizer .....	300
DW_sync	
Single Clock Data Bus Synchronizer .....	303
Chapter Test - JTAG Overview .....	305
DW_tap	
TAP Controller .....	306
DW_tap_uc	
TAP Controller with USERCODE support .....	308
DW_bc_1	
Boundary Scan Cell Type BC_1 .....	311
DW_bc_2	
Boundary Scan Cell Type BC_2 .....	312
DW_bc_3	
Boundary Scan Cell Type BC_3 .....	313



DW_bc_4 Boundary Scan Cell Type BC_4 .....	314
DW_bc_5 Boundary Scan Cell Type BC_5 .....	315
DW_bc_7 Boundary Scan Cell Type BC_7 .....	316
DW_bc_8 Boundary Scan Cell Type BC_8 .....	317
DW_bc_9 Boundary Scan Cell Type BC_9 .....	319
DW_bc_10 Boundary Scan Cell Type BC_10 .....	320
Chapter GTECH Library Overview .....	321
Index .....	322

# Preface

## 1.1 About This Manual

This document is a Quick Reference overview of the DesignWare Building Block components. For detailed product information, refer to individual component databooks and manuals mentioned in the following chapters.

### 1.1.1 Manual Overview

This manual contains the following chapters:

Preface	Describes the manual and typographical conventions and symbols; tells how to get technical assistance.
Chapter “DesignWare Overview”	Contains an overview and general description of the DesignWare Library product offering.
Chapter “DesignWare Building Block Synthesizable IP”	Contains a brief description of each DesignWare Library Synthesizable IP.
Chapter “GTECH Library Overview”	This chapter briefly describes the DesignWare Foundry Libraries.

## 1.2 Synopsys Common Licensing (SCL)

You can find general SCL information on the Web at:

<http://www.synopsys.com/Support/Licensing>

## 1.3 Getting Help

If you have a question about using Synopsys products, please consult product documentation that is installed on your network or located at the root level of your Synopsys product CD-ROM (if available). You can also access documentation for DesignWare products on the Web:

- ❖ Product documentation for many DesignWare products:  
<http://www.synopsys.com/dw/dwlibdocs.php>
- ❖ Datasheets for individual DesignWare IP components, located using “Search for IP”:  
<http://www.synopsys.com/designware>

You can also contact the Synopsys Support Center in the following ways:

- ❖ Open a call to your local support center using this page:  
<http://www.synopsys.com/Support>
- ❖ Send an e-mail message to [support\\_center@synopsys.com](mailto:support_center@synopsys.com).



- ❖ Telephone your local support center:
  - ◆ United States:  
Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific Time, Mon – Fri.
  - ◆ Canada:  
Call 1-650-584-4200 from 7 AM to 5:30 PM Pacific Time, Mon – Fri.
  - ◆ All other countries:  
Find other local support center telephone numbers at the following URL:  
<http://www.synopsys.com/Support/GlobalSupportCenters>

## 1.4 Additional Information

For additional Synopsys documentation, refer to the following page:

<http://www.synopsys.com/dw/dwlibdocs.php>

For up-to-date information about the latest implementation IP and verification models, visit the DesignWare home page:

<http://www.synopsys.com/designware>

## 1.5 Comments?

To report errors or make suggestions, please send e-mail to:

[support\\_center@synopsys.com](mailto:support_center@synopsys.com).

To report an error that occurs on a specific page, select the entire page (including headers and footers), and copy to the buffer. Then paste the buffer to the body of your e-mail message. This will provide us with information to identify the source of the problem.

# DesignWare Overview

Synopsys DesignWare IP, the world's most widely-used, silicon-proven IP provides designers with a broad portfolio of synthesizable implementation IP, hardened PHYs and verification IP for ASIC and SoC designs.

The DesignWare family includes:

- ❖ **DesignWare Library** - contains the principal ingredients for design and verification including high speed datapath components, AMBA On-Chip Bus, memory portfolio, verification models of standard bus and I/Os, foundry libraries, popular Star IP cores and board verification IP.
- ❖ **Discovery Verification Platform** - contains reusable, pre-verified verification IP of the industry's most popular bus and interface standards such as AMBA, PCI Express, PCI-X, PCI, USB On-the-Go, Ethernet, I2C and thousands of memory models.
- ❖ **DesignWare Cores** (Digital and Mixed-Signal) - silicon-proven, digital and mixed-signal standards-based connectivity IP such as PCI Express, PCI-X, PCI, USB 2.0 On-the-Go (OTG), USB 2.0 PHY, USB 1.1 and Ethernet.
- ❖ **DesignWare Building Block IP** - a subset of DesignWare Library. Building Block IP is a collection of reusable intellectual property blocks that are tightly integrated into Synopsys synthesis environment. Using DesignWare Building Block IP allows significant productivity and performance improvement through pre-designed, pre-verified, highly optimized critical building blocks. The purpose of this document is to provide a quick reference of all the Building Block IP.

For more detail, refer to "["DesignWare Building Block Synthesizable IP" on page 16.](#)

## 1.6 Related DesignWare IP

Visit the Discovery Verification Platform web page at:

<http://www.synopsys.com/Tools/Verification/FunctionalVerification/VerificationIP>

Visit the DesignWare Interface and Standards (Cores) web page at:

<http://www.synopsys.com/IP/InterfaceIP>

Visit the Analog IP web page at:

<http://www.synopsys.com/IP/AnalogIP>



# DesignWare Overview

Synopsys DesignWare IP, the world's most widely-used, silicon-proven IP provides designers with a broad portfolio of synthesizable implementation IP, hardened PHYs and verification IP for ASIC and SoC designs.

The DesignWare family includes:

- ❖ **DesignWare Library** - contains the principal ingredients for design and verification including high speed datapath components, AMBA On-Chip Bus, memory portfolio, verification models of standard bus and I/Os, foundry libraries, and board verification IP.

Details available at: <http://www.synopsys.com/IP/SoCInfrastructureIP>

- ❖ **DesignWare Building Block IP** - a subset of DesignWare Library. Building Block IP is a collection of reusable intellectual property blocks that are tightly integrated into Synopsys synthesis environment. Using DesignWare Building Block IP allows significant productivity and performance improvement through pre-designed, pre-verified, highly optimized critical building blocks. The purpose of this document is to provide a quick reference of all the Building Block IP.

For more detail, refer to "["DesignWare Building Block Synthesizable IP" on page 16](#)".

- ❖ **DesignWare minPower Components** - offer unique, power-optimized datapath architectures that enable DC Ultra™ to automatically generate circuits that suppress switching activity and glitches, reducing both dynamic and leakage power for mobile devices and high-performance applications. DesignWare minPower Components also include more than 40 instantiable blocks that incorporate low power design techniques such as enhanced clock gating, built-in datapath gating.

[http://www.synopsys.com/dw/ipdir.php?ds=dw\\_minpower](http://www.synopsys.com/dw/ipdir.php?ds=dw_minpower)

- ❖ **DesignWare Cores** (Digital and Mixed-Signal) - silicon-proven, digital and mixed-signal standards-based connectivity IP such as PCI Express, PCI-X, PCI, USB 2.0 On-the-Go (OTG), USB 2.0 PHY, USB 1.1 and Ethernet.

For details, refer to the DesignWare Interface and Standards (Cores) web page at:

<http://www.synopsys.com/IP/InterfaceIP>

- ❖ **Analog IP Solutions** - Analog IP optimized for system-on-chip (SoC) integration in a variety of applications, including broadband communications, multimedia and embedded data acquisition. Synopsys Analog IP portfolio is optimized to serve as analog interfaces to the SoC, deploying high-performance analog-to-digital converters (ADCs), digital-to-analog converters (DACs), audio analog codecs and complete analog front-ends (AFE) for communications and analog video.

See more at: <http://www.synopsys.com/IP/AnalogIP>

Related IP includes:

- ❖ **Discovery Verification Platform** - contains reusable, pre-verified verification IP of the industry's most popular bus and interface standards such as AMBA, PCI Express, PCI-X, PCI, USB On-the-Go, Ethernet, I2C and thousands of memory models.

For details, refer to the Verification web page:

<http://www.synopsys.com/Tools/Verification>

# DesignWare Building Block Synthesizable IP

The DesignWare Building Block IP (formally called Foundation Library) is a collection of nearly 200 technology-independent, high-quality, high-performance and reusable intellectual property blocks that are tightly integrated into the Synopsys synthesis environment. Using DesignWare Building Block IP allows transparent, high-level optimization of performance during synthesis. With the large number of parts available, design reuse is enabled and significant productivity gains are possible.

This library contains high-performance implementations of Basic Library IP plus many IP that implement more advanced arithmetic and sequential logic functions. The DesignWare Building Block IP consists of:

**Table 1-1**

Component Group	Description	Component Type
Datapath	Arithmetic, floating point, trigonometric, and sequential math IP ( <a href="#">page 30</a> )	Synthesizable RTL
Data Integrity	Data integrity IP such as CRC, ECC, 8b10b... ( <a href="#">page 156</a> )	Synthesizable RTL
Digital Signal Processing (DSP)	FIR and IIR filters, DCT and iDCT ( <a href="#">page 170</a> )	Synthesizable RTL
Logic	Combinational, sequential, and control IP ( <a href="#">page 19</a> )	Synthesizable RTL
Interface	Clock Domain Crossing (CDC) ( <a href="#">page 281</a> )	Synthesizable RTL
Memory	Registers, FIFO, synchronous and asynchronous RAM, and stack IP ( <a href="#">page 251</a> )	Synthesizable RTL
Test	JTAG IP such as boundary scan, TAP controller... ( <a href="#">page 305</a> )	Synthesizable RTL
GTECH	Technology-independent IP library to aid users in developing technology-independent parts ( <a href="#">page 321</a> )	Synthesizable RTL

## 1.1 Building Block IP for DC QuickStart

The following topics provide the basic information for you to get started using the DesignWare Building Block IP.

### 1.1.1 Setting Up DesignWare Building Block IP in DC-TCL

Include the following lines in your .synopsys\_dc.setup file and ensure that you have a valid DesignWare Library license:

```
set target_library "your_library.db"
set synthetic_library "dw_foundation.sldb"
set link_library "{*} $target_library"
set link_library [concat $link_library $synthetic_library]

set search_path [list . [format "%s%s" $SYNOPSYS {/libraries/syn }] \
[format "%s%s" $SYNOPSYS {/dw/sim_ver/ }] ./your_library_path ]
```

If DesignWare license is required and it is not available, DC terminates the run. You can override this default behavior by adding “DesignWare” to the list of licenses as the value of synlib\_wait\_for\_design\_license.

The get\_license command obtains a DesignWare license. This licensed checked out by the current user until remove\_license is used or until the program is exited.

For information on synlib\_wait\_for\_design\_license and get\_license commands, see:

<https://www.synopsys.com/dw/doc.php/doc/dwf/manuals/dwug.pdf>

### 1.1.2 Accessing DesignWare Building Block IP in DC

You can access DesignWare Building Block IP either by operator or functional inference, or by instantiating the component directly. The example below shows how to access these IP:

```
Verilog
assign PROD = IN1 * IN2; // Operator Inference
assign PROD = DWF_mult_tc(IN1, IN2); // Function Inference
DW02_mult #(8, 8) U1 (A, B, TC, PRODUCT); // Instantiation
```

Details about inference and instantiation in VHDL and Verilog are in the following directory:  
\$SYNOPSYS/dw/examples.

### 1.1.3 Synthesizing DesignWare Building Block IP in DC

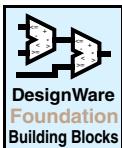
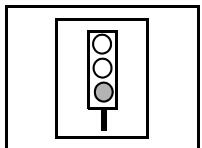
Design Compiler automatically selects the best implementation for combinational DesignWare Building Block IP. You can also force Design Compiler to select the implementation of your choice either by adding Synopsys Compiler directives or by using the following commands:

```
dc_shell-t> set_dont_use standard.sldb/DW01_add/rpl
dc_shell-t> set_implementation pparc {add_68}
```

#### 1.1.4 Simulating DesignWare Building Block IP

Synopsys VCS simulator uses the default setup file while simulating DesignWare Building Block IP. Use the following options to simulate DesignWare Building Block IP with a Verilog simulator:

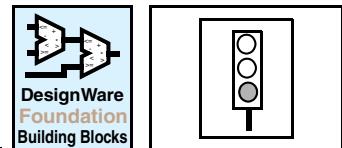
```
-y $SYNOPSYS/dw/sim_ver +libext+.v+
```



## Application Specific – Control Logic

The Control Logic IP consist of a family of arbiters. The arbiter components are distinguished from each other primarily by the arbitration scheme they embody.

The components DW\_arb\_sp and DW\_arb\_dp are based on the static fixed priority scheme and dynamically programmable priority scheme, respectively. Each of these components has multiple architectural implementations optimized for timing or area. The number of clients connected to the arbiter is parametrizable from 2 to 32. Other features like parking and locking are available through parameter selection.

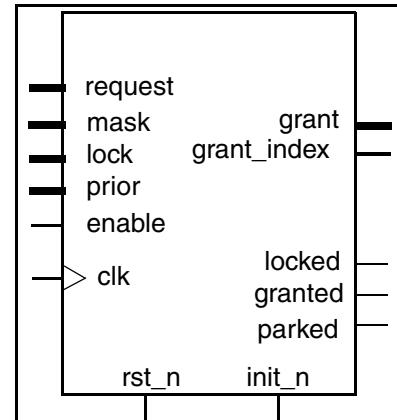
**DW\_arb\_2t**

Two-Tier Arbiter with Dynamic/Fair-Among-Equal Scheme

**DW\_arb\_2t**

Two-Tier Arbiter with Dynamic/Fair-Among-Equal Scheme

- ❖ Parameterizable number of clients
- ❖ Programmable mask for all clients
- ❖ Park feature - default grant when no requests are pending
- ❖ Lock feature - ability to lock the currently granted client
- ❖ Registered/unregistered outputs

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Asynchronous reset for all registers (active low)
init_n	1 bit	Input	Synchronous reset for all registers (active low)
enable	1 bit	Input	Enables clocking (active high)
request	$n$ bit(s)	Input	Input request from clients
prior	$n \times p\_width$ bit(s)	Input	Priority vector from the clients of the arbiter
lock	$n$ bit(s)	Input	Active high signal to lock the grant to the current request. By setting <code>lock(i) = 1</code> , the arbiter is locked to the <code>request(i)</code> if it is currently granted. For <code>lock(i) = 0</code> , the lock on the arbiter is removed.
mask	$n$ bit(s)	Input	Active high input to mask specific clients. By setting <code>mask(i) = 1</code> , <code>request(i)</code> is masked. For <code>mask(i) = 0</code> , the mask on the <code>request(i)</code> is removed.
parked	1 bit	Output	Flag to indicate that there are no requesting clients and the grant of resources has defaulted to <code>park_index</code>
granted	1 bit	Output	Flag to indicate that arbiter has issued a grant to one of the clients
locked	1 bit	Output	Flags that the arbiter is locked by a client
grant	$n$ bit(s)	Output	Grant output
grant_index	$\text{ceil}(\log_2 n)$ bit(s)	Output	Index of the requesting client that has been currently granted or the client designated by <code>park_index</code> in <code>park_mode</code>

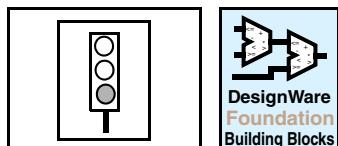
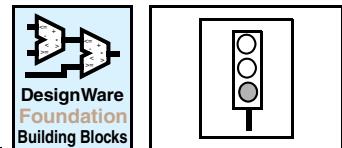


Table 1-2 Parameter Description

Parameter	Values	Description
n	2 to 32 Default: 4	Number of arbiter clients
p_width	1 to 5 Default: 2	Width of the priority vector of each client
park_mode	0 or 1 Default: 1	<i>park_mode</i> = 1 includes logic to enable parking when no clients are requesting and <i>park_mode</i> = 0 contains no logic for parking.
park_index	0 to $n-1$ Default: 0	Index of the client used for parking
output_mode	0 or 1 Default: 1	<i>output_mode</i> = 1 includes registers at the outputs <i>output_mode</i> = 0 contains no output registers

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis Model	DesignWare

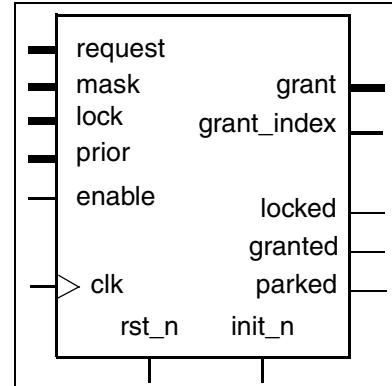
**DW\_arb\_dp**

Arbiter with Dynamic Priority Scheme

**DW\_arb\_dp**

Arbiter with Dynamic Priority Scheme

- ❖ Parameterizable number of clients
- ❖ Programmable mask for all clients
- ❖ Park feature - default grant when no requests are pending
- ❖ Lock feature - ability to lock the currently granted client
- ❖ Registered/unregistered outputs
- ❖ Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Asynchronous reset for all registers (active low)
init_n	1 bit	Input	Synchronous reset for all registers (active low)
enable	1 bit	Input	Enables clocking (active high)
request	$n$ bit(s)	Input	Input request from clients
prior	$n \cdot \text{ceil}(\log_2 n)$ bit(s)	Input	Priority vector from the clients of the arbiter
lock	$n$ bit(s)	Input	Signal to lock the grant to the current request. By setting lock ( $i$ ) = 1, the arbiter is locked to the request ( $i$ ) if it is currently granted. For lock ( $i$ ) = 0 the lock on the arbiter is removed.
mask	$n$ bit(s)	Input	Input to mask specific clients. By setting mask ( $i$ ) = 1, request ( $i$ ) is masked. For mask ( $i$ ) = 0 the mask on the request ( $i$ ) is removed.
parked	1 bit	Output	Flag to indicate that there are no requesting clients and the grant of resources has defaulted to client designated by park_index
granted	1 bit	Output	Flag to indicate that the arbiter has issued a grant to one of the requesting clients
locked	1 bit	Output	Flag to indicate that the arbiter is locked by a client
grant	$n$ bit(s)	Output	Grant output
grant_index	$\log_2 n$ bit(s)	Output	Index of the requesting client that has been currently granted or the client designated by park_index in park_mode

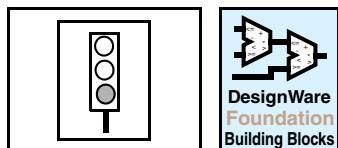


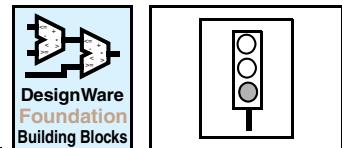
Table 1-2 Parameter Description

Parameter	Values	Description
n	2 to 32 Default: 4	Number of arbiter clients
park_mode	0 or 1 Default: 1	park mode = 1 includes logic to enable parking when no clients are requesting park mode = 0 contains no logic for parking.
park_index	0 to $n-1$ Default: 0	Index of the client used for parking
output_mode	0 or 1 Default: 1	output_mode = 1 includes registers at the outputs output_mode = 0 contains no output registers

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare
lpwr <sup>a</sup>	Low Power Synthesis model	DesignWare-LP

a. Effectiveness of low power design depends on the use of the `-gate_clock` option to `compile_ultra` command

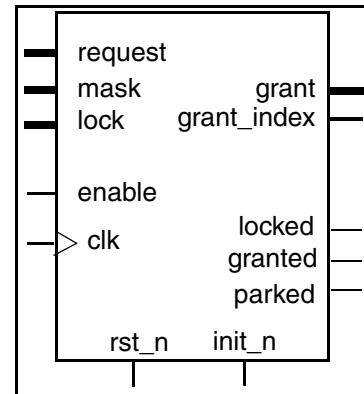
**DW\_arb\_fcfs**

Arbiter with First-Come-First-Served Priority Scheme

**DW\_arb\_fcfs**

Arbiter with First-Come-First-Served Priority Scheme

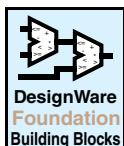
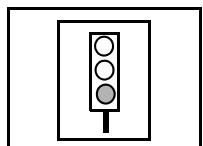
- ❖ Parameterizable number of clients
- ❖ Programmable mask for all clients
- ❖ Park feature - default grant when no requests are pending
- ❖ Lock feature - ability to lock the currently granted client
- ❖ Registered/unregistered outputs

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Asynchronous reset for all registers (active low)
init_n	1 bit	Input	Synchronous reset for all registers (active low)
enable	1 bit	Input	Enables clocking (active high)
request	$n$ bit(s)	Input	Input request from clients
lock	$n$ bit(s)	Input	Active high signal to lock the grant to the current request. By setting lock( $i$ ) = 1, the arbiter is locked to the request ( $i$ ) if it is currently granted. For lock ( $i$ ) = 0, the lock on the arbiter is removed.
mask	$n$ bit(s)	Input	Active high input to mask specific clients. By setting mask( $i$ ) = 1, request( $i$ ) is masked. For mask( $i$ ) = 0, the mask on the request( $i$ ) is removed.
parked	1 bit	Output	Flag to indicate that there are no requesting clients and the grant of resources has defaulted to park_index
granted	1 bit	Output	Flag to indicate that arbiter has issued a grant to one of the clients
locked	1 bit	Output	Flags that the arbiter is locked by a client
grant	$n$ bit(s)	Output	Grant output
grant_index	ceil(log <sub>2</sub> $n$ ) bit(s)	Output	Index of the requesting client that has been currently granted or the client designated by park_index in park_mode

**Table 1-2 Parameter Description**

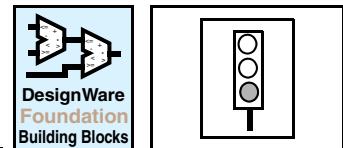
Parameter	Values	Description
n	2 to 32 Default: 4	Number of arbiter clients

**Table 1-2 Parameter Description**

Parameter	Values	Description
park_mode	0 or 1 Default: 1	park mode = 1 includes logic to enable parking when no clients are requesting and park_mode = 0 contains no logic for parking.
park_index	0 to $n-1$ Default: 0	Index of the client used for parking
output_mode	0 or 1 Default: 1	output_mode = 1 includes registers at the outputs output_mode = 0 contains no output registers

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

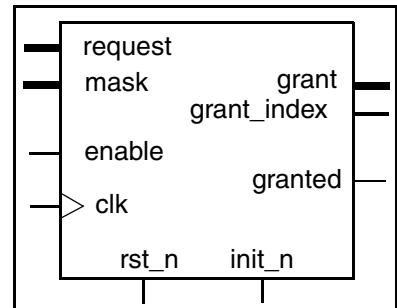
**DW\_arb\_rr**

Arbiter with Round Robin Priority Scheme

**DW\_arb\_rr**

Arbiter with Round Robin Priority Scheme

- ❖ Parameter controlled number of clients
- ❖ Programmable mask for all clients
- ❖ Parameter controlled optional registered output
- ❖ Parameter controlled grant\_index coding scheme
- ❖ Single cycle arbitration

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Active Low Asynchronous reset
init_n	1 bit	Input	Synchronous reset for all registers (active low)
enable	1 bit	Input	Enables clocking (active high)
request	$n$ bit(s)	Input	Input request from clients
mask	$n$ bit(s)	Input	Active high input to mask specific clients. By setting <code>mask(<math>i</math>) = 1</code> , <code>request(<math>i</math>)</code> is masked.
granted	1 bit	Output	Flag to indicate that arbiter has issued a grant to one of the clients
grant	$n$ bit(s)	Output	Grant output, one bit per client.
grant_index	$\text{ceil}(\log_2(n + \text{mod2}(index\_mode)))$ bit(s)	Output	Index of the client that has been currently granted

**Table 1-2 Parameter Description**

Parameter	Values	Description
n	2 to 32 Default: 4	Number of arbiter clients
output_mode	0 or 1 Default: 1	<code>output_mode = 1</code> includes registers at the outputs <code>output_mode = 0</code> contains no output registers
index_mode <sup>a</sup>	0 to 2 Default: 0	<code>index_mode = 0</code> or <code>1</code> causes <code>grant_index</code> value to be <code>grant</code> bit position plus 1 <code>index_mode = 2</code> causes <code>grant_index</code> values to be the bit position of <code>grant</code>

a. The `index_mode` parameter does not exist in DW\_arb\_rr prior to the 201206.3 DWBB release. For backward compatibility, the default `index_mode = 0` behavior is the same as DW\_arb\_rr prior to the 201206.3 DWBB release.

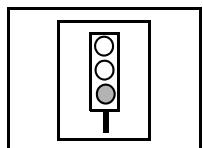
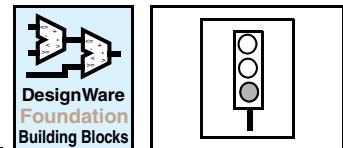


Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis Model	DesignWare

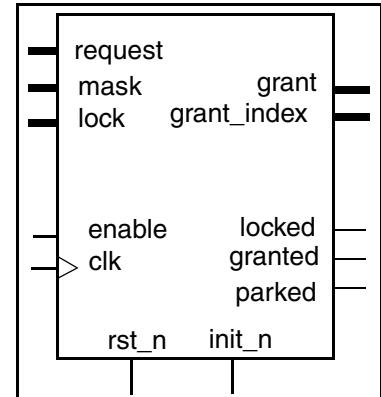
**DW\_arb\_sp**

Arbiter with Static Priority Scheme

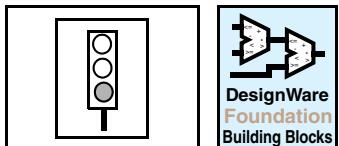
**DW\_arb\_sp**

Arbiter with Static Priority Scheme

- ❖ Parameterizable number of clients
- ❖ Programmable mask for all clients
- ❖ Park feature - default grant to a client when no requests are pending
- ❖ Lock feature - ability to lock the currently granted client
- ❖ Registered/unregistered outputs
- ❖ Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

<b>Pin Name</b>	<b>Width</b>	<b>Direction</b>	<b>Function</b>
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Asynchronous reset for all registers (active low)
init_n	1 bit	Input	Synchronous reset for all registers (active low)
enable	1 bit	Input	Enables clocking (active high)
request	$n$ bit(s)	Input	Input request from clients
lock	$n$ bit(s)	Input	Active high signal to lock input. By setting lock( $i$ ) = 1, the arbiter is locked to the request ( $i$ ) if it is currently granted. For lock ( $i$ ) = 0, the lock on the arbiter is removed.
mask	$n$ bit(s)	Input	Active high input to mask specific clients. By setting mask( $i$ ) = 1, request( $i$ ) is masked. For mask( $i$ ) = 0, the mask on the request( $i$ ) is removed.
parked	1 bit	Output	Flag to indicate that there are no requesting clients and the grant of resources has defaulted to park_index
granted	1 bit	Output	Flag to indicate that arbiter has issued a grant to one of the clients
locked	1 bit	Output	Flags that the arbiter is locked by a client
grant	$n$ bit(s)	Output	Grant output
grant_index	$\log_2 n$ bit(s)	Output	Index of the requesting client that has been currently issued the grant or the client designated by park_index in park_mode

**Table 1-2 Parameter Description**

Parameter	Values	Description
n	2 to 32 Default: 4	Number of arbiter clients
park_mode	0 or 1 Default: 1	<i>park mode</i> = 1 includes logic to enable parking when no clients are requesting and <i>park_mode</i> = 0 contains no logic for parking.
park_index	0 to n-1 Default: 0	Index of the client used for parking
output_mode	0 or 1 Default: 1	<i>output_mode</i> = 1 includes registers at the outputs <i>output_mode</i> = 0 contains no output registers

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	RTL synthesis model	DesignWare
lpwr <sup>a</sup>	Low Power synthesis model	DesignWare-LP

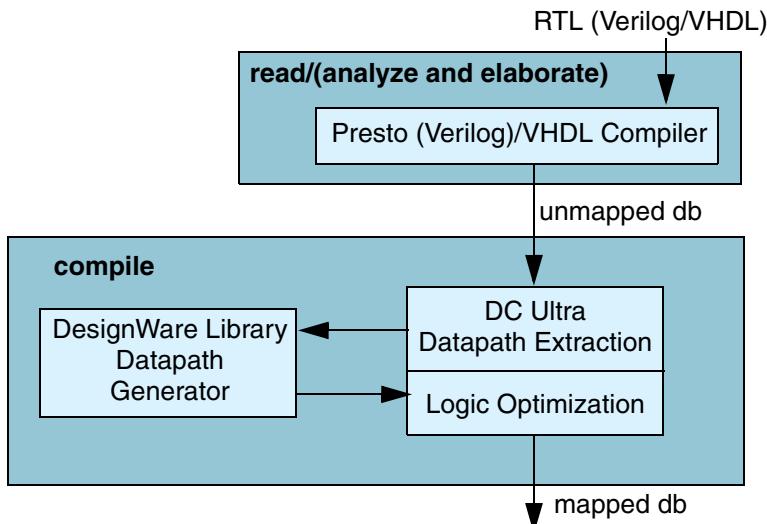
a. Effectiveness of low power design depends on the use of the `-gate_clock` option to `compile_ultra` command

# Datapath Generator Overview

The new datapath generators improve the quality of synthesized datapaths in two steps:

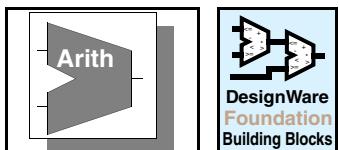
1. By using more sophisticated extraction and partitioning of datapaths from RTL code,
2. By improved synthesis of the extracted datapaths.

The following figure shows the flow for datapath synthesis. After the RTL code is analyzed and elaborated by the Presto (Verilog)/VHDL Compiler, the datapath portions of the RTL are extracted by DC Ultra and then synthesized by the datapath generators in the DesignWare Library.



DC Ultra partitions datapaths that are extracted from RTL into large sum-of-product and product-of-sum blocks. This reduces the number of expensive carry propagations to a minimum, resulting in faster and smaller circuits. In sum-of-products, a multiplication can be followed by an addition without a carry propagation before the addition. Similarly, in product-of-sums, an addition can be followed by a multiplication without a carry propagation before the multiplication. The same techniques are also applied to reduce the number of carry propagations in magnitude comparisons of complete sum-of-products. Resource and common subexpression sharing allow for further area savings.

The datapath generators then perform a constraint- and technology-driven synthesis of the extracted sum-of-product and product-of-sum blocks. Enhanced algorithms are used to construct optimized adder reduction trees and carry-propagate adders to meet the given timing constraints with minimal area requirements for the specified technology and conditions. A smart generation feature selects the best among alternative implementation variants. Special datapath library cells are automatically used where available and beneficial. Optimized structures are generated for special arithmetic operations, like constant multiplication or squaring.



## Datapath – Arithmetic Overview

The Datapath arithmetic DesignWare Building Block IP, many of which are inferred, are applicable to ASIC or SoC designs. These IP are high-performance arithmetic implementations (based on a fast carry look-ahead architecture) to augment those in the Basic IP Library. The Basic IP Library is included in your (V)HDL Compiler product.

Most IP in this category have multiple architectures for each function (architecturally optimized for either performance or area). This provides you with the best architecture for your design goals. All IP have a parameterized word length.

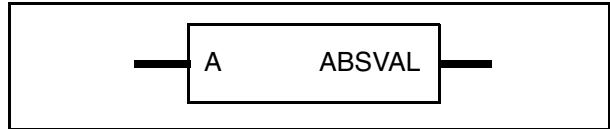
**DW01\_absval**

Absolute Value

**DW01\_absval**

Absolute Value

- ❖ Parameterized word length
- ❖ Inferable through a function call.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data
ABSVAL	<i>width</i> bit(s)	Output	Absolute value of A

**Table 1-2 Parameter Description**

Parameter	Values	Function
width	$\geq 1$	Word length of A and ABSVAL

**Table 1-3 Synthesis Implementations<sup>a</sup>**

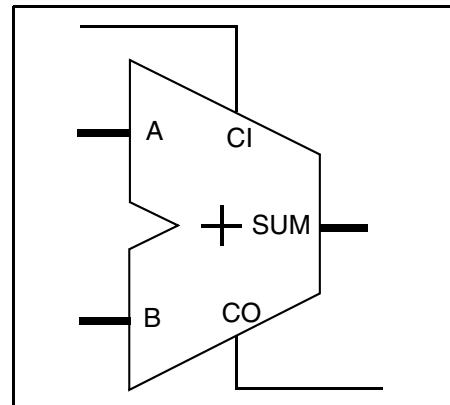
Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	none
cla	Carry-look-ahead synthesis model	none
clf	Fast carry-look-ahead synthesis model	DesignWare

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*.

## DW01\_add

Adder

- ❖ Parameterized word length
- ❖ Carry-in and carry-out signals



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data
B	<i>width</i> bit(s)	Input	Input data
CI	1 bit	Input	Carry-in
SUM	<i>width</i> bit(s)	Output	Sum of (A + B + CI)
CO	1 bit	Output	Carry-out

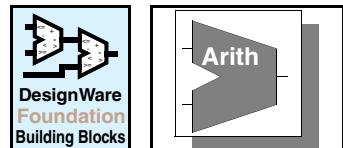
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Word length of A, B, and SUM

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation	Function	License Feature Required
rpl	Ripple-carry synthesis model	none
cla	Carry-look-ahead synthesis model	none
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	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*.

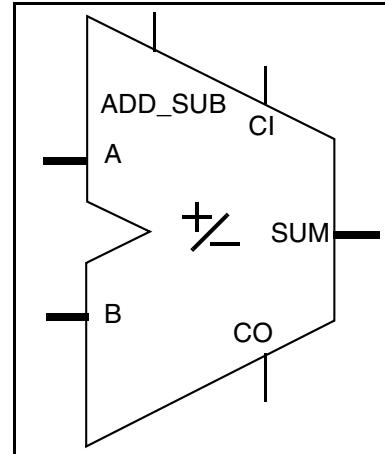
**DW01\_addsub**

Adder-Subtractor

**DW01\_addsub**

Adder-Subtractor

- ❖ Parameterized word length
- ❖ Carry-in and carry-out signals

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data
B	<i>width</i> bit(s)	Input	Input data
CI	1 bit	Input	Carry/borrow-in
ADD_SUB	1 bit	Input	Addition/subtraction control
SUM	<i>width</i> bit(s)	Output	Sum ( $A + B + CI$ ) or difference ( $A - B - CI$ )
CO	1 bit	Output	Carry/borrow-out

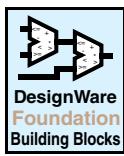
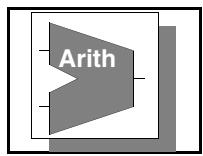
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Word length of A, B, and SUM

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Implementation	License Feature Required
rpl	Ripple-carry synthesis model	none
cla	Carry-look-ahead synthesis model	none
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	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*.



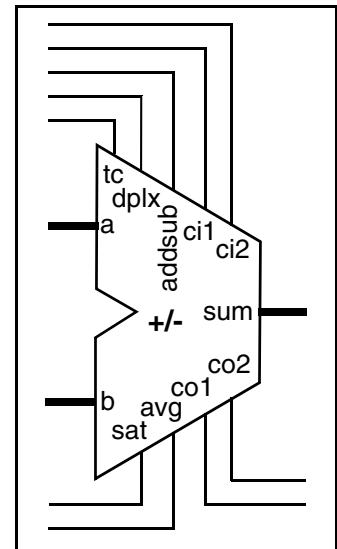
**DW\_addsub\_dx**

Duplex Adder/Subtractor with Saturation and Rounding

**DW\_addsub\_dx**

Duplex Adder/Subtractor with Saturation and Rounding

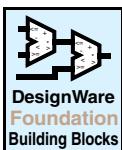
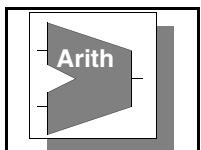
- ❖ Selectable single full-width Add/Sub (simplex) or two smaller width Add/Sub operations (duplex)
- ❖ Selectable saturation mode
- ❖ Selectable average mode
- ❖ Selectable number system (unsigned or twos complement)
- ❖ Parameterized full word width
- ❖ Parameterized partial word width (allowing for asymmetric partial width operations)
- ❖ Carry-out signals (one for lower half and one for full and upper half) that numerically extend the calculated sum (maintaining full precision)
- ❖ Carry-in signals (one for full and lower half and one for upper half)

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	width bit(s)	Input	Input data
b	width bit(s)	Input	Input data
ci1	1 bit	Input	Full or part1 carry input
ci2	1 bit	Input	Part2 carry input
addsub	1 bit	Input	Add/subtract select input 0 = performs add 1 = performs subtract
tc	1 bit	Input	Two's complement select (active high)
sat	1 bit	Input	Saturation mode select (active high)
avg	1 bit	Input	Average mode select (active high)
dplx	1 bit	Input	Duplex mode select (active high)
sum	width bit(s)	Output	Output data
co1	1 bit	Output	Part1 carry output
co2	1 bit	Output	Full width or part2 carry output

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 4$	Word width of a, b, and sum
p1_width	2 to $width - 2$	Word width of part1 of duplex Add/Sub

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple Carry Synthesis Model	DesignWare
rpcs	Ripple Carry Select Synthesis Model	DesignWare
csm	Conditional Sum 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*.

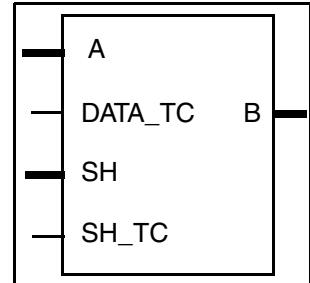
**DW01\_ash**

Arithmetic Shifter

**DW01\_ash**

Arithmetic Shifter

- ❖ Parameterized word length
- ❖ Parameterized shift coefficient width
- ❖ Inferable using a function call

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$ bit(s)	Input	Input data
DATA_TC	1 bit	Input	Data two's complement control 0 = unsigned 1 = signed
SH	$SH\_width$ bit(s)	Input	Shift control
SH_TC	1 bit	Input	Shift two's complement control 0 = unsigned 1 = signed
B	$A\_width$ bit(s)	Output	Output data

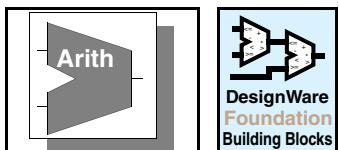
**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 2$	Word length of A and B
SH_width	$\geq 1$	Word length of SH

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation	Function	License Feature Required
mx2	Implement using 2:1 multiplexers only.	none
str	Synthesis model	DesignWare
astr	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*.



## DW\_bin2gray

Binary to Gray Converter

- ❖ Parameterized word length
- ❖ Inferable using a function call



Table 1-1 Pin Description

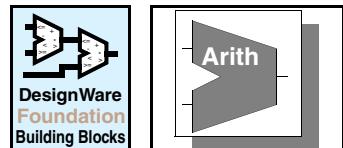
Pin Name	Width	Direction	Function
b	<i>width</i> bit(s)	Input	Binary coded input data
g	<i>width</i> bit(s)	Output	Gray coded output data

Table 1-2 Parameter Description

Parameter	Values	Description
width	$\geq 1$	Input word length

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



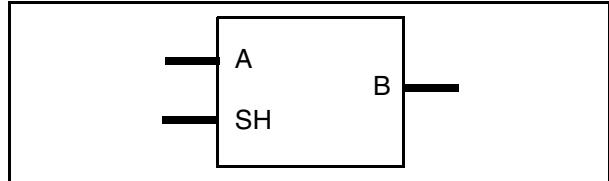
## DW01\_bsh

### Barrel Shifter

## DW01\_bsh

### Barrel Shifter

- ❖ Parameterized data and shift coefficient word lengths
- ❖ Inferable using a function call



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$	Input	Input data
SH	$SH\_width$	Input	Shift control
B	$A\_width$	Output	Shifted data out

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 2$	Word length of A and B
SH_width	$\leq \text{ceil}(\log_2[A\_width])$	Word length of SH

**Table 1-3 Synthesis Implementations<sup>a</sup>**

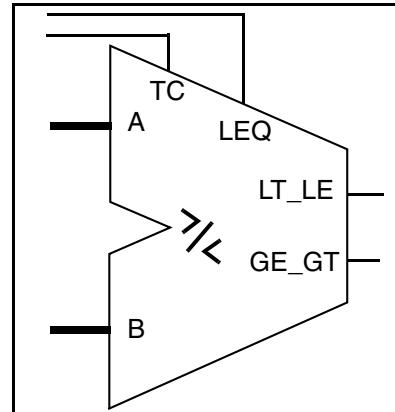
Implementation Name	Function	License Feature Required
str	Synthesis model target for speed	DesignWare
astr	Synthesis model target for area	DesignWare

- 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.

## DW01\_cmp2

### 2-Function Comparator

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's-complement) data operatio



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data
B	<i>width</i> bit(s)	Input	Input data
LEQ	1 bit	Input	Output condition control
TC	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
LT_LT	1 bit	Output	Less-than/less-than-or-equal output condition
GE_GT	1 bit	Output	Greater-than-or-equal/greater-than output condition

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Word length of A and B

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	none
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	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*.

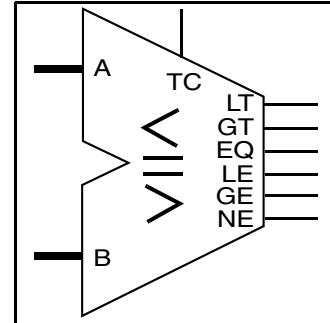
**DW01\_cmp6**

6-Function Comparator

**DW01\_cmp6**

6-Function Comparator

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's-complement) data comparison

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data
B	<i>width</i> bit(s)	Input	Input data
TC	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
LT	1 bit	Output	Less-than output condition
GT	1 bit	Output	Greater-than output condition
EQ	1 bit	Output	Equal output condition
LE	1 bit	Output	Less-than-or-equal output condition
GE	1 bit	Output	Greater-than-or-equal output condition
NE	1 bit	Output	Not equal output condition

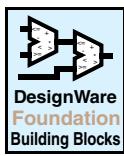
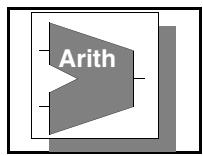
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Word length of A and B

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	none
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

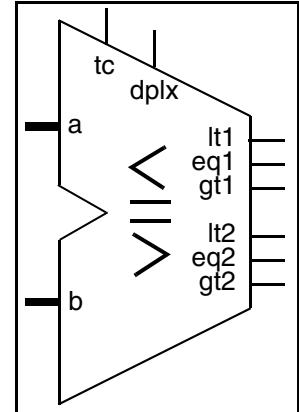
- a. During synthesis, Design Compiler will select the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table. For more details, please refer to the *DesignWare Building Block IP User Guide*.



**DW\_cmp\_dx****DW\_cmp\_dx**

## Duplex Comparator

- ❖ Selectable single full width Compare, or two smaller width Compare operations (duplex)
- ❖ Selectable number system (unsigned or two's complement)
- ❖ Parameterized full word width
- ❖ Parameterized partial word width (allowing for asymmetric partial width operations)
- ❖ Separate flags for Less Than, Equal To, and Greater Than
- ❖ Two sets of flags for duplex operation

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	width bit(s)	Input	Input data
b	width bit(s)	Input	Input data
tc	1 bit	Input	Two's complement control
dplx	1 bit	Input	Duplex mode select (active high)
lt1	1 bit	Output	Part1 : less-than output condition
eq1	1 bit	Output	Part1 : equal output condition
gt1	1 bit	Output	Part1 : greater-than output condition
lt2	1 bit	Output	Full width or part2 : less-than output condition
eq2	1 bit	Output	Full width or part2 : equal output condition
gt2	1 bit	Output	Full width or part2 : greater-than output condition

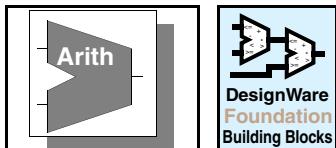
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 4$	Word width of a and b
p1_width	2 to $width - 2$	Word width of part1 of duplex compare

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
bk	Brent-Kung synthesis model	DesignWare

<sup>a</sup>.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*



## DW\_cntr\_gray

Gray Code Counter

- ❖ Gray encoded output
- ❖ Asynchronous and synchronous reset
- ❖ Count enable

Provides minPower benefits with the DesignWare-LP license.

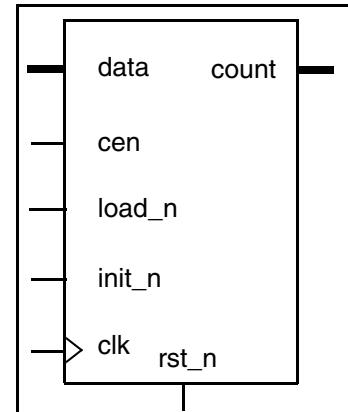


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, asynchronous, active low
init_n	1 bit	Input	Reset, synchronous, active low
load_n	1 bit	Input	Enable data load to counter, active low
data	<i>width</i> bit(s)	Input	Counter load input
cen	1 bit	Input	Count enable, active high
count	<i>width</i> bit(s)	Output	Gray coded counter output

Table 1-2 Parameter Description

Parameter	Values	Description
width	$\geq 1$	Word length of counter

Table 1-3 Synthesis Implementations<sup>a</sup>

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	DesignWare
cla	Carry-lookahead 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*.

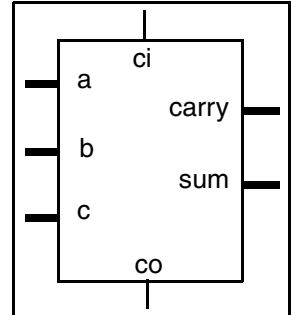
**DW01\_csa**

Carry Save Adder

**DW01\_csa**

Carry Save Adder

- ❖ Parameterized word length
- ❖ Carry-in and carry-out signals

**Table 1-1 Pin Description**

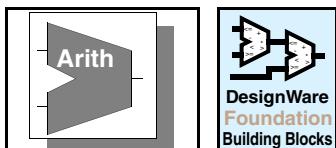
Pin Name	Width	Direction	Function
a	<i>width</i> bit(s)	Input	Input data
b	<i>width</i> bit(s)	Input	Input data
c	<i>width</i> bit(s)	Input	Input data
ci	1 bit	Input	Carry-in
carry	<i>width</i> bit(s)	Output	Carry output data
sum	<i>width</i> bit(s)	Output	Sum output data
co	1 bit	Output	Carry-out

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Word length of a, b, c, sum, and carry

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



## DW01\_dec

### Decrementer

- ❖ Parameterized word length

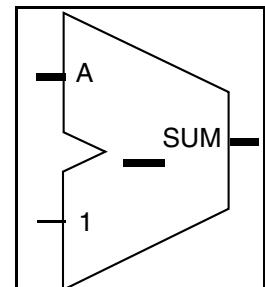


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data
SUM	<i>width</i> bit(s)	Output	Decremented ( $A - 1$ )

Table 1-2 Parameter Description

Parameter	Values	Description
width	$\geq 1$	Word length of A and SUM

Table 1-3 Synthesis Implementations<sup>a</sup>

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	none
cla	Carry-look-ahead synthesis model	none
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	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.

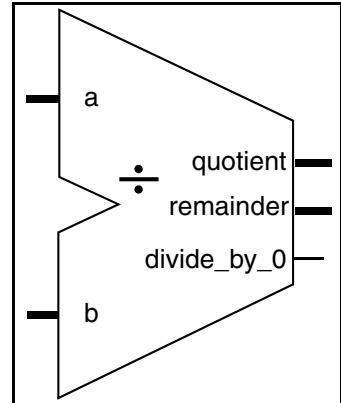
**DW\_div**

Combinational Divider

**DW\_div**

Combinational Divider

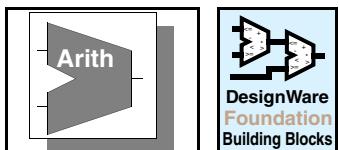
- ❖ Parameterized word lengths
- ❖ Unsigned and signed (two's complement) data operation
- ❖ Remainder or modulus as second output
- ❖ Inferable using a function call
- ❖ Multiple architectures for area/performance trade-offs
- ❖ Optional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	$a\_width$ bit(s)	Input	Dividend
b	$b\_width$ bit(s)	Input	Divisor
quotient	$a\_width$ bit(s)	Output	Quotient
remainder	$b\_width$ bit(s)	Output	Remainder / modulus
divide_by_0	1 bit	Output	Indicates if b equals 0

**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\geq 1$	Word length of a
b_width	$\geq 1$	Word length of b
tc_mode	0 or 1 Default: 0	Two'- complement control
rem_mode	0 or 1 Default: 1	Remainder output control

Table 1-3 Synthesis Implementations<sup>a</sup>

Implementation Name	Function	License Feature
rpl	Restoring ripple-carry divider synthesis model	None
cla	Restoring carry-look-ahead divider synthesis model	DesignWare
cla2	Radix-4 restoring carry-look-ahead divider synthesis model	DesignWare
cla3	Radix-8 restoring carry-look-ahead divider synthesis model	DesignWare
mlt <sup>b</sup>	Multiplicative divider 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 details, refer to the *DesignWare Building Block IP User Guide*.
- b. The 'mlt' implementation is only useful when the divisor is small. So, this implementation is only valid when the *b\_width* parameter (size of the divisor) is no greater than 10.

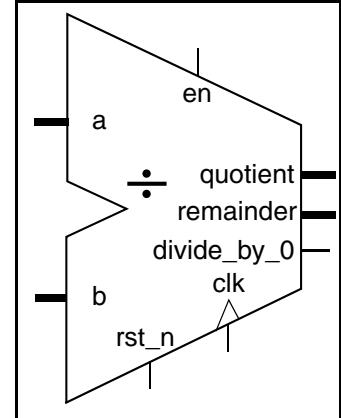
**DW\_div\_pipe**

Stallable Pipelined Divider

**DW\_div\_pipe**

Stallable Pipelined Divider

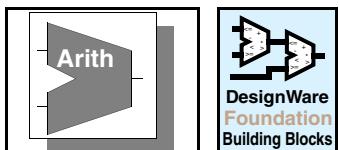
- ❖ Parameterized word length
- ❖ Parameterized unsigned and signed data operation
- ❖ Parameterized number of pipeline stages
- ❖ Parameterized stall mode (stallable or non-stallable)
- ❖ Parameterized reset mode (no reset, asynchronous or synchronous reset)
- ❖ Automatic pipeline retiming
- ❖ Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset, active-low (not used if parameter <i>rst_mode</i> =0)
en	1 bit	Input	Register enable, active high (used only if parameter <i>stall_mode</i> =1) 0 = stall 1 = enable register
a	<i>a_width</i> bit(s)	Input	Dividend
b	<i>b_width</i> bit(s)	Input	Divisor
quotient	<i>a_width</i> bit(s)	Output	Quotient <i>a</i> / <i>b</i>
remainder	<i>b_width</i> bit(s)	Output	Remainder
divide_by_0	1 bit	Output	Indicates if <i>b</i> equals zero

**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\geq 2$ Default: None	Word length of <i>a</i>
b_width	$\geq 2 \leq a\_width$ Default: None	Word length of <i>b</i>
tc_mode	0 or 1 Default: 0	Two's complement control 0 = unsigned 1 = signed

**Table 1-2 Parameter Description (Continued)**

Parameter	Values	Description
rem_mode	0 or 1 Default: 1	Remainder output control 0 = modulus 1 = remainder
num_stages	$\geq 2$ Default: 2	Number of pipeline stages
stall_mode	0 or 1 Default: 1	Stall mode 0 = non-stallable 1 = stallable
rst_mode	0 to 2 Default: 1	Reset mode 0 = no reset 1 = asynchronous reset 2 = synchronous reset

**Table 1-3 Synthesis Implementations**

Implementation Name	Implementation	License Feature Required
str <sup>a</sup>	Pipelined str synthesis model	DesignWare

a. One of pparch or apparch implementation is selected based the constraints of the design.

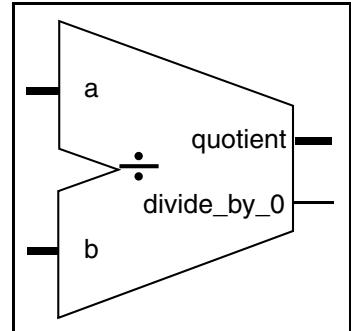
**DW\_div\_sat**

Combinational Divider with Saturation

**DW\_div\_sat**

Combinational Divider with Saturation

- ❖ Parameterized word lengths
- ❖ Parameterized quotient length with saturation
- ❖ Unsigned and signed (two's complement) data operation
- ❖ Multiple architectures for area/performance trade-off

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	$a\_width$ bit(s)	Input	Dividend
b	$b\_width$ bit(s)	Input	Divisor
quotient	$q\_width$ bit(s)	Output	Quotient
divide_by_0	1 bit	Output	Indicates if b equals 0

**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\geq 2$ Default: None	Word length of a
b_width	$\geq 2$ Default: None	Word length of b
q_width	2 to $a\_width$ Default: None	Word length of quotient
tc_mode	0 or 1 Default: 0	Two's- complement control

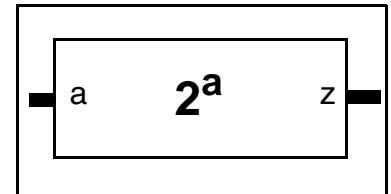
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
cla	Restoring carry-look-ahead divider synthesis model	DesignWare
cla2	Radix-4 restoring carry-look-ahead divider synthesis model	DesignWare
cla3	Radix-8 restoring carry-look-ahead divider synthesis model	DesignWare

## DW\_exp2

### Base-2 Exponential

- ❖ Parameterized word width
- ❖ Supports up to 60 bits of precision



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>op_width</i> bits	Input	Input data in the range (0,1)
z	<i>op_width</i> bits	Output	$2^a$ in the range (1,2)

**Table 1-2 Parameter Description**

Parameter	Values	Description
op_width	2 to 60 <sup>a</sup> Default: 8	Word length of <i>a</i> and <i>z</i>
arch	0 or 2 Default: 2	Implementation selection 0 - area optimized 1 - speed optimized 2- 2007.12 implementation
err_range	1 or 2 Default: 1	Error range of the result compared to the infinitely precise result 1 - 1 ulp 2 - 2 ulps Note: error range is 1 ulp when <i>arch</i> =2

- a. The synthesis model fully covers this range, as does the Verilog simulation model within VCS, but the VHDL simulation model is limited to *op\_width* range 2-38.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Implement using the Datapath Generator technology combined with static DesignWare components	DesignWare
str	Synthesis Model	DesignWare

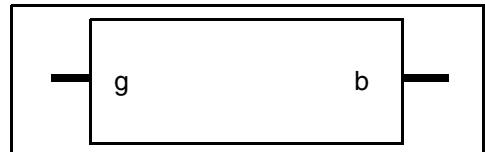
**DW\_gray2bin**

Gray-to-Binary Converter

**DW\_gray2bin**

Gray-to-Binary Converter

- ❖ Parameterized word length
- ❖ Inferable using a function call

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
g	<i>width</i> bit(s)	Input	Gray coded input data
b	<i>width</i> bit(s)	Output	Binary coded output data

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Input word length

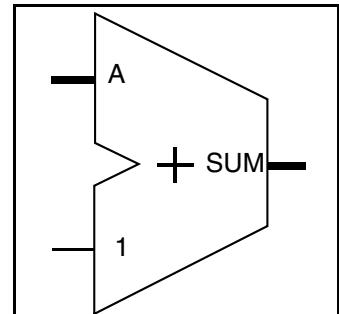
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	DesignWare
cla	Carry-lookahead synthesis model	DesignWare

# DW01\_inc

## Incrementer

- ❖ Parameterized word length



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data
SUM	<i>width</i> bit(s)	Output	Increment ( $A + 1$ )

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Word length of A and SUM

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	none
cla	Carry-look-ahead synthesis model	none
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	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*.

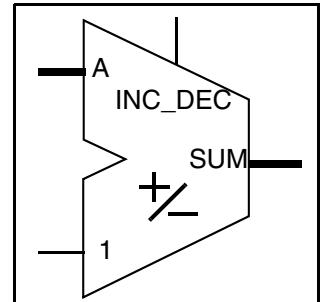
**DW01\_incdec**

Incrementer-Decrementer

**DW01\_incdec**

Incrementer-Decrementer

- ❖ Parameterized word length

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data
INC_DEC	1 bit	Input	Increment control 0 = increment ( $A + 1$ ) 1 = decrement ( $A - 1$ )
SUM	<i>width</i> bit(s)	Output	Increment ( $A + 1$ ) or decrement ( $A - 1$ )

**Table 1-2 Parameter Description**

Parameter	Values	Function
width	$\geq 1$	Word length of A and SUM

**Table 1-3 Synthesis Implementations<sup>a</sup>**

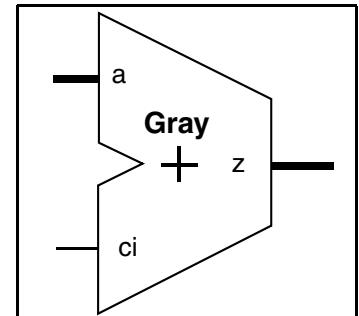
Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	none
cla	Carry look-ahead synthesis model	none
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

- 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*.

## DW\_inc\_gray

Gray Incrementer

- ❖ Parameterized word length
- ❖ Inferable using a function call



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	width bit(s)	Input	Gray coded input data
ci	1 bit	Input	Carry-in
z	width bit(s)	Output	Gray coded output data

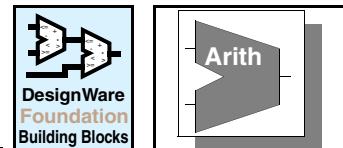
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Input word length

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	DesignWare
cla	Carry-lookahead 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*.

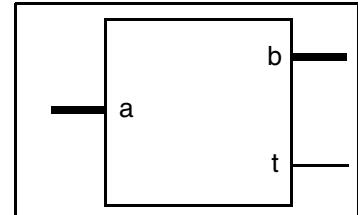
**DW\_inv\_sqrt**

Reciprocal of Square-Root

**DW\_inv\_sqrt**

Reciprocal of Square-Root

- ❖ Parameterized word length
- ❖ Combinational implementation to maximize speed
- ❖ Easy to pipeline for increased throughput

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	$a\_width$ bit(s)	Input	Input fractional bits only
b	$a\_width$ bit(s)	Output	Output data
t	1 bit	Output	sticky bit

**Table 1-2 Parameter Description**

Parameter	Values	Description
$a\_width$	$\geq 2$	Word length of $a$ and $b$
$prec\_control^a$	$\leq(a\_width - 2) / 2$	Controls the number of LS bits that may be removed or added to the internal precision, where 0 implies to keep all the bits.

- a. Starting in the Z2007.03 release, the DW\_inv\_sqrt component has a tighter upper bound for the prec\_control parameter (from  $a\_width-2$  to  $(a\_width-2)/2$ ). If you are using this parameter on an earlier released version of this component, there are two possible consequences when the most recent version is used: (1) the component will not elaborate for the prec\_control value previously used if the value is now out of range, and (2) the component will not have the same numerical behavior as before for the same prec\_control value.  
As a rule of thumb to get the same numerical behavior: a previous value  $x$  should be mapped to a value  $y=\text{floor}(x-(a\_width-2)/2)$ , if the mapped value  $y$  is positive, or 0 otherwise.

**Table 1-3 Synthesis Implementations**

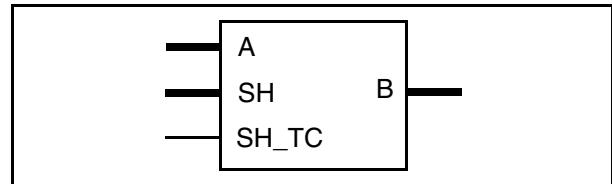
Implementation Name	Function	License Feature Required
rtl	Implement using the Datapath Generator technology combined with static DesignWare components.	DesignWare

## DW\_Ibsh

### Barrel Shifter with Preferred Left Direction

- ❖ Parameterized data and shift coefficient word lengths
- ❖ Capable of rotating in both directions
- ❖ Inferable using a function call

e.



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$	Input	Input data
SH	$SH\_width$	Input	Shift control
SH_TC	1 bit	Input	Shift two's complement control 0 = unsigned 1 = signed
B	$A\_width$	Output	Shifted data out

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A and B
SH_width	$\leq \text{ceil}(\log_2[A\_width])$	Word length of SH

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
str	Synthesis model target for speed	DesignWare
astr	Synthesis model target for area	DesignWare
mx2	Implement using 2:1 multiplexers only. The mx2 implementation is valid only for SH_width values up to and including 31.	none

- a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you can 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*.

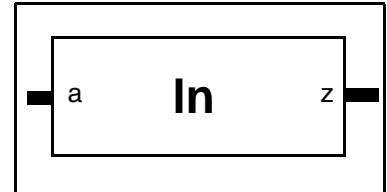
**DW\_In**

Natural Logarithm

**DW\_In**

Natural Logarithm

- ❖ Parameterized word width
- ❖ Supports up to 60 bits of precision

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>op_width</i> bits	Input	Input data in the range [1,2)
z	<i>op_width</i> bits	Output	$\ln(a)$ in the range $[0,\ln(2))$

**Table 1-2 Parameter Description**

Parameter	Values	Description
op_width	2 to 60 bits	Word length of a and z
arch	0 or 1 Default: 0	Implementation selection 0 - area optimized 1 - speed optimized
err_range	1 or 2 Default: 1	Error range of the result compared to the infinitely precise result 1 - 1 ulp 2 - 2 ulps

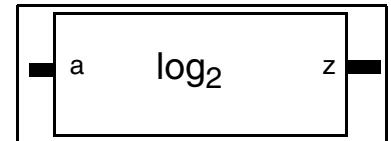
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Implement using the Datapath Generator technology combined with static DesignWare components	DesignWare

## DW\_log2

### Base 2 Logarithm

- ❖ Parameterized word width
- ❖ Supports up to 60 bits of precision



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>op_width</i> bits	Input	Input data in the range (1,2).
z	<i>op_width</i> bits	Output	$\log_2(a)$ in the range (0,1)

**Table 1-2 Parameter Description**

Parameter	Values	Description
op_width	2 to 60	Word length of a and z.
arch	0 to 2 Default: 2	Implementation selection 0 - area optimized 1 - speed optimized 2- 2007.12 implementation
err_range	1 or 2 Default: 1	Error range of the result compared to the infinitely precise result 1 - 1 ulp 2 - 2 ulps Note: error range is 1 ulp when arch=2

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Implement using the Datapath Generator technology combined with static DesignWare components	DesignWare

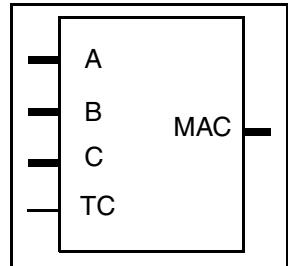
**DW02\_mac**

Multiplier-Accumulator

**DW02\_mac**

Multiplier-Accumulator

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's-complement) data operation
- ❖ Inferable using a function call

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$ bit(s)	Input	Multiplier
B	$B\_width$ bit(s)	Input	Multiplicand
C	$A\_width + B\_width$ bit(s)	Input	Addend
TC	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
MAC	$A\_width + B\_width$ bit(s)	Output	MAC result ( $A \times B + C$ )

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A
B_width	$\geq 1$	Word length of B

**Table 1-3 Synthesis Implementations<sup>a</sup>**

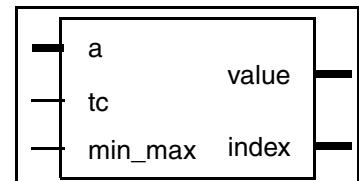
Implementation Name	Function	License Feature Required
pparch	Delay-optimized flexible Booth Wallace	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

- 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*.

# DW\_minmax

## Minimum/Maximum Value

- ❖ Parameterized number of inputs
- ❖ Parameterized word length
- ❖ Unsigned and signed (two's complement) data operation
- ❖ Dynamically selectable mode (minimum or maximum)
- ❖ Additional output gives an index of the minimum or maximum input
- ❖ Inferable using a function call



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	$num\_inputs \times width$ bit(s)	Input	Concatenated input data
tc	1 bit	Input	Two's complement control
min_max	1 bit	Input	Minimum/maximum control 0 = minimum (a) 1 = maximum (a)
value	$width$ bit(s)	Output	Minimum/maximum value
index	$\text{ceil}(\log_2[num\_inputs])$ bit(s)	Output	Index of minimum/maximum input

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Input word length
num_inputs	$\geq 2$ Default: 2	Number of inputs

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
cla	Carry-lookahead tree synthesis model	DesignWare
clas	Carry-lookahead/select tree 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*

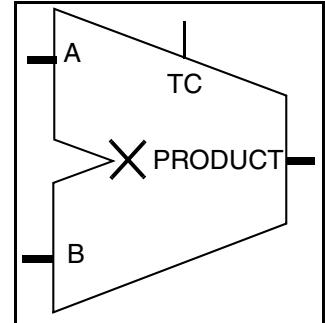
**DW02\_mult**

Multiplier

**DW02\_mult**

Multiplier

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's-complement) data operation

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$ bit(s)	Input	Multiplier
B	$B\_width$ bit(s)	Input	Multiplicand
TC	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
PRODUCT	$A\_width + B\_width$ bit(s)	Output	Product $A \times B$

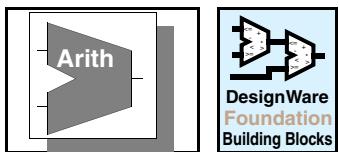
**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A
B_width	$\geq 1$	Word length of B

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
cса <sup>a</sup>	Carry-save array synthesis model	none
pparch <sup>b</sup>	Delay-optimized flexible Booth Wallace	DesignWare
apparch <sup>b</sup>	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

- a. The 'cса' implementation does not require a DesignWare license and is therefore the only implementation available to DesignCompiler Expert with no DesignWare license. When a DesignWare license is available (which is required for Design Compiler Ultra), Design Compiler will not select the 'cса' implementation, although it can manually be selected using the `set_implementation` command.



- b. The 'pparch' (optimized for delay) and 'apparch' (optimized for area) implementations are dynamically generated to best meet your constraints. The 'pparch' and 'apparch' implementations can generate a variety of multiplier architectures including Radix-2 non-Booth, Radix-4 non-Booth, Radix-4 Booth recoded and Radix-8 Booth recoded. Generation of 'pparch' and 'apparch' implementations automatically detects which input would be best suited to be considered the multiplier as opposed to the multiplicand (for Booth recoding). The 'pparch' and 'apparch' implementations are generated making use of any special arithmetic technology cells that are found to be available in your target technology library. The `dc_shell` command, `set_dp_smartgen_options`, can be used to force specific multiplier architectures. For more information on forcing generated arithmetic architectures, use '`man set_dp_smartgen_options`' (in `dc_shell`) to get a listing of the command options.

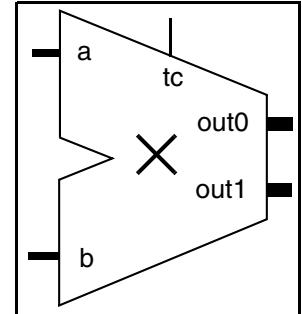
**DW02\_multp**

Partial Product Multiplier

**DW02\_multp**

Partial Product Multiplier

- ❖ Parameterized word lengths
- ❖ Parameterized sign extension of partial product outputs for use in summing products
- ❖ Unsigned and signed (two's-complement) data operation

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	$a\_width$ bit(s)	Input	Multiplier
b	$b\_width$ bit(s)	Input	Multiplicand
tc	1 bit	Input	Two's complement 0 = unsigned 1 = signed
out0	$out\_width$ bit(s)	Output	Partial product of $(a \times b)$
out1	$out\_width$ bit(s)	Output	Partial product of $(a \times b)$

**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\geq 1$	Word length of a
b_width	$\geq 1$	Word length of b
out_width	$\geq a\_width + b\_width + 2$	Word length of out0 and out1
verif_en <sup>a</sup>	0 to 3 Default: 1	Verification Enable Control 0 - out0 and out1 are always the same for a given input pair (a,b) 1 - MSB of out0 is always '0'. Out0 and out1 change with time for the same input pair (a,b) 2 - MSB of out0 or out1 is always '0'. Out0 and out1 change with time for the same input pair (a,b) 3 - no restrictions on MSBs of out0 and out1. Out0 and out1 change with time for the same input pair (a,b)

a. Although the *verif\_en* value can be set for all simulators, CS randomization is only implemented when using Synopsys simulators (VCS, VCS-MX). For more information about *verif\_en*, refer to “[Simulation Using Random Carry-save Representation \(VCS/VCS-MX only\)](#)” on page 3.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
pparch <sup>a</sup>	Delay-optimized flexible parallel-prefix	DesignWare
apparch <sup>a</sup>	Area-optimized flexible parallel-prefix	DesignWare

- a. The 'pparch' (optimized for delay) and 'apparch' (optimized for area) implementations are dynamically generated to best meet your constraints. The 'pparch' and 'apparch' implementations can generate a variety of multiplier architectures including Radix-2 non-Booth, Radix-4 non-Booth, Radix-4 Booth recoded and Radix-8 Booth recoded. Generation of 'pparch' and 'apparch' implementations automatically detects which input would be best suited to be considered the multiplier as opposed to the multiplicand (for Booth recoding). The pparch and apparch implementations are generated making use of any special arithmetic technology cells that are found to be available in your target technology library. The `dc_shell` command, `set_dp_smartgen_options`, can be used to force specific multiplier architectures. For more information on forcing generated arithmetic architectures, use '`man set_dp_smartgen_options`' (in `dc_shell`) to get a listing of the command options.

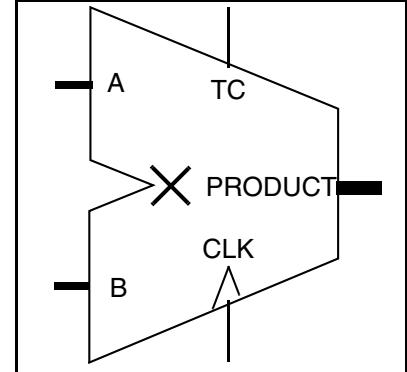
**DW02\_mult\_2\_stage**

Two-Stage Pipelined Multiplier

**DW02\_mult\_2\_stage**

Two-Stage Pipelined Multiplier

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's-complement) data operation
- ❖ Two-stage pipelined architecture
- ❖ Automatic pipeline retiming

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$ bit(s)	Input	Multiplier
B	$B\_width$ bit(s)	Input	Multiplicand
TC	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
CLK	1 bit	Input	Clock
PRODUCT	$A\_width + B\_width$ bit(s)	Output	Product ( $A \times B$ )

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A
B_width	$\geq 1$	Word length of B

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Booth-recoded Wallace-tree synthesis model	DesignWare

## DW02\_mult\_3\_stage

Three-Stage Pipelined Multiplier

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's-complement) data operation
- ❖ Three-stage pipelined architecture
- ❖ Automatic pipeline retiming

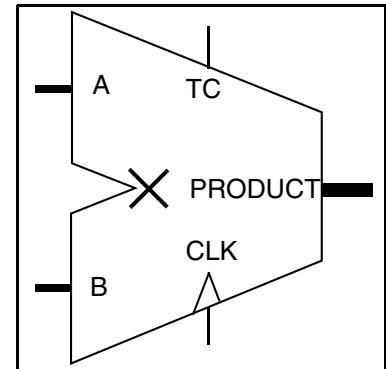


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	$A\_width$ bit(s)	Input	Multiplier
B	$B\_width$ bit(s)	Input	Multiplicand
TC	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
CLK	1 bit	Input	Clock
PRODUCT	$A\_width + B\_width$ bit(s)	Output	Product ( $A \times B$ )

Table 1-2 Parameter Description

Parameter	Values	Description
A_width	$\geq 1$	Word length of A
B_width	$\geq 1$	Word length of B

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Booth-recoded Wallace-tree synthesis model	DesignWare

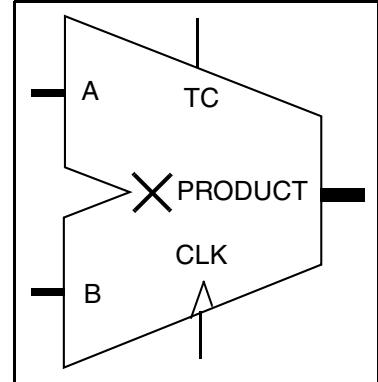
**DW02\_mult\_4\_stage**

Four-Stage Pipelined Multiplier

**DW02\_mult\_4\_stage**

Four-Stage Pipelined Multiplier

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's-complement) data operation
- ❖ Four-stage pipelined architecture
- ❖ Automatic pipeline retiming

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$ bit(s)	Input	Multiplier
B	$B\_width$ bit(s)	Input	Multiplicand
TC	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
CLK	1 bit	Input	Clock
PRODUCT	$A\_width + B\_width$ bit(s)	Output	Product ( $A \times B$ )

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A
B_width	$\geq 1$	Word length of B

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Booth-recoded Wallace-tree synthesis model	DesignWare

## DW02\_mult\_5\_stage

Five-Stage Pipelined Multiplier

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's-complement) data operation
- ❖ Five-stage pipelined architecture
- ❖ Automatic pipeline retiming

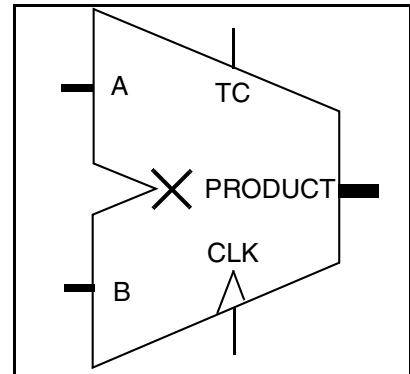


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	$A\_width$ bit(s)	Input	Multiplier
B	$B\_width$ bit(s)	Input	Multiplicand
TC	1 bit	Input	Two's complement 0 = unsigned 1 = signed
CLK	1 bit	Input	Clock
PRODUCT	$A\_width + B\_width$ bit(s)	Output	Product ( $A \times B$ )

Table 1-2 Parameter Description

Parameter	Values	Description
A_width	$\geq 1$	Word length of A
B_width	$\geq 1$	Word length of B

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Booth-recoded Wallace-tree synthesis model	DesignWare

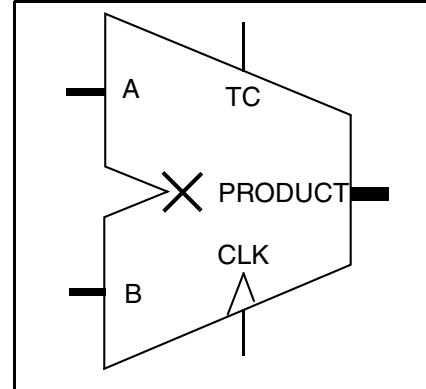
**DW02\_mult\_6\_stage**

Six-Stage Pipelined Multiplier

**DW02\_mult\_6\_stage**

Six-Stage Pipelined Multiplier

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's-complement) data operation
- ❖ Six-stage pipelined architecture
- ❖ Automatic pipeline retiming

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$ bit(s)	Input	Multiplier
B	$B\_width$ bit(s)	Input	Multiplicand
TC	1 bit	Input	Two's complement 0 = unsigned 1 = signed
CLK	1 bit	Input	Clock
PRODUCT	$A\_width + B\_width$ bit(s)	Output	Product ( $A \times B$ )

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A
B_width	$\geq 1$	Word length of B

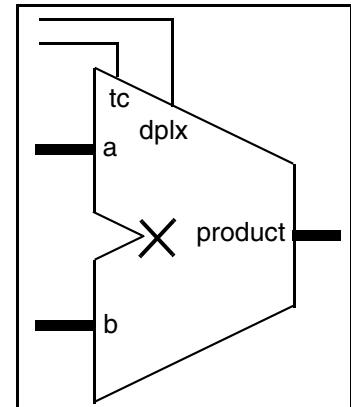
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Booth-recoded Wallace-tree synthesis model	DesignWare

## DW\_mult\_dx

### Duplex Multiplier

- ❖ Selectable single full-width multiplier (simplex) or two parallel smaller-width multiplier (duplex) operations
- ❖ Area and delay are similar to those of the DW02\_mult wallace architecture
- ❖ Selectable number system (unsigned or two's complement)
- ❖ Parameterized full word width
- ❖ Parameterized partial word width (allowing for asymmetric partial width operations)



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	width bit(s)	Input	Input data
b	width bit(s)	Input	Input data
tc	1 bit	Input	Two's complement control
dplx	1 bit	Input	Duplex mode select, active high
product	$width \times 2$ bit(s)	Output	Product(s)

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 4^a$	Word width of a and b
p1_width	2 to $width - 2^b$	Word width of Part1 of duplex multiplier

- Due to the limitation of memory addressing ranges of the computer operating system, there is an upper limit for parameter *width*.
- For the best performance of DW\_mult\_dx, *p1\_width* should be set in the range [ $width/2$ ,  $width - 2$ ].

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
wall	Booth-recoded Wallace-tree synthesis model	DesignWare

**DW\_mult\_pipe**

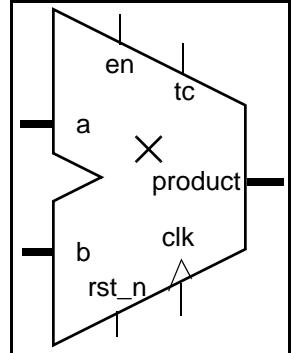
Stallable Pipelined multiplier

**DW\_mult\_pipe**

Stallable Pipelined multiplier

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's complement) pipelined multiplication
- ❖ Parameterized number of pipeline stages
- ❖ Parameterized stall mode (stallable or non-stallable)
- ❖ Parameterized reset mode (no reset, asynchronous or synchronous reset)
- ❖ Automatic pipeline retiming

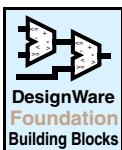
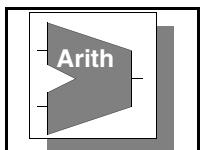
Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset, active-low (not used if parameter <i>rst_mode</i> =0)
en	1 bit	Input	Register enable, active high (used only if parameter <i>stall_mode</i> =1) 0 = stall 1 = enable register
tc	1 bit	Input	Two's complement control: 0 = unsigned 1 = signed
a	<i>a_width</i> bit(s)	Input	Multiplier
b	<i>b_width</i> bit(s)	Input	Multiplicand
product	<i>a_width+b_width</i> bit(s)	Output	Product $a \times b$

**Table 1-2 Parameter Description**

Parameter	Values	Description
<i>a_width</i>	$\geq 1$ Default: None	Word length of <i>a</i>
<i>b_width</i>	$\geq 1$ Default: None	Word length of <i>b</i>
<i>num_stages</i>	$\geq 2$ Default: 2	Number of pipeline stages
<i>stall_mode</i>	0 or 1 Default: 1	Stall mode 0 = non-stallable 1 = stallable

**Table 1-2 Parameter Description**

Parameter	Values	Description
rst_mode	0 to 2 Default: 1	Reset mode 0 = no reset 1 = asynchronous reset 2 = synchronous reset

**Table 1-3 Synthesis Implementations**

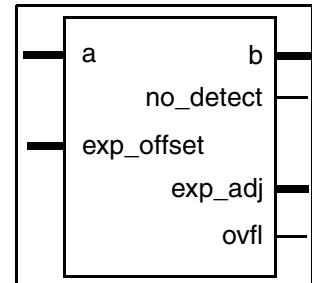
Implementation Name	Implementation	License Feature Required
str <sup>a</sup>	Pipelined str synthesis model	DesignWare

a. Either pparch or apparch implementation is selected based on the constraints of the design.

**DW\_norm****DW\_norm**

Normalization for Fractional Input

- ❖ Parameterized word lengths
- ❖ Parameterized search window
- ❖ Indication of exceptional cases
- ❖ Optional C++ simulation model available in the DWFC Library

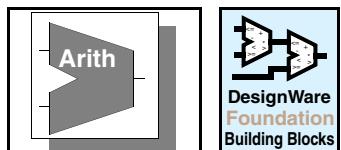
**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
<i>a</i>	<i>a_width</i> bit(s)	Input	Input data
<i>exp_offset</i>	<i>exp_width</i> bit(s)	Input	Offset value for the exponent
<i>no_detect</i>	1 bit	Output	Result of search for the leading bit with value 1 in the search window 0 = bit found 1 = bit not found
<i>ovfl</i>	1 bit	Output	Value provided at output <i>exp_adj</i> is negative or incorrect
<i>b</i>	<i>a_width</i> bit(s)	Output	Normalized output data
<i>exp_adj</i>	<i>exp_width</i> bit(s)	Output	<i>exp_offset</i> combined with the number of bit positions the input <i>a</i> was shifted to the left ( <i>n</i> ). <i>exp_ctr=0</i> → <i>exp_offset+n</i> <i>exp_ctr=1</i> → <i>exp_offset-n</i>

**Table 1-2 Parameter Description**

Parameter	Values	Description
<i>a_width</i>	$\geq 2$ Default: 8	Word length of <i>a</i> and <i>b</i>
<i>srch_wind</i>	2 to <i>a_width</i> Default: 8	Search window for the leading 1 bit (from bit position 0 to <i>a_width</i> -1)
<i>exp_width<sup>a</sup></i>	$\geq \text{ceil}(\log_2(\text{srch\_wind}))$ Default: 4	Word length of <i>exp_offset</i> and <i>exp_adj</i>
<i>exp_ctr</i>	0 or 1 Default: 0	Control over <i>exp_adj</i> computation

- a. **IMPORTANT NOTE:** The lower bound of parameter *exp\_width* was modified starting on G-2012.06-SP1, previous lower bound was 1.

Table 1-3 Synthesis Implementations<sup>a</sup>

Implementation Name	Function	License Feature Required
rtl	Synthesis model for fast architecture	DesignWare
str	Synthesis model for reduced area	DesignWare

a. During synthesis, Design Compiler will select the appropriate architecture for your constraints.

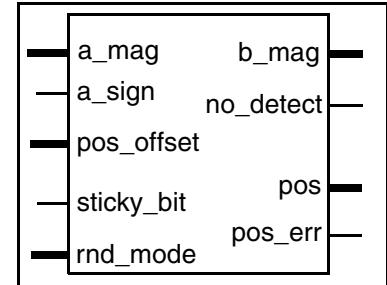
**DW\_norm\_rnd**

Normalization and rounding

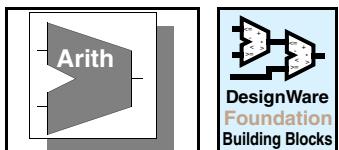
**DW\_norm\_rnd**

Normalization and rounding

- ❖ Parameterized word lengths
- ❖ Parameterized search window
- ❖ Exponent calculation useful for floating-point numbers
- ❖ Implements the most used rounding modes
- ❖ Provides status indications for exceptional cases
- ❖ DesignWare datapath generator is employed for better timing and area
- ❖ Optional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a_mag	$a\_width$ bit(s)	Input	Input data
a_sign	1 bit	Input	0 = positive 1 = negative
pos_offset	$exp\_width$ bit(s)	Input	Offset value for the position of the binary point
sticky_bit	1 bit	Input	Indicates the presence of non-zero bits in fractional bit positions beyond bit $a\_mag_{a\_width-1}$
rnd_mode	3 bits	Input	Rounding mode 000 – Round to nearest even 001 – Round towards zero 010 – Round to plus infinity 011 – Round to minus infinity 100 – Round up 101 – Round away from zero
no_detect	1 bit	Output	Result of MS 1 bit search in the search window. 0 = bit found 1 = bit not found
pos_err	1 bit	Output	Value provided at output pos cannot fit in an $exp\_width$ -bit vector or pos is negative
b_mag	$b\_width$ bit(s)	Output	Normalized and rounded output data
pos	$exp\_width$ bit(s)	Output	pos_offset combined with the number of bit positions that input a_mag was shifted ( $n$ ): $exp\_ctr=0 \rightarrow pos\_offset+n$ $exp\_ctr=1 \rightarrow pos\_offset-n$

**Table 1-2 Parameter Description**

Parameter	Values	Description
$a\_width$	$\geq 2$ Default: 16	Word length of $a\_mag$
$srch\_wind$	2 to $a\_width$ Default: 4	Search window for the MS 1 bit (from left to right, or from bit 0 to $a\_width-1$ )
$exp\_width$	$\geq 1$ Default: 4	Word length of $pos\_offset$ and $pos$
$b\_width$	2 to $a\_width$ Default: 10	Word length of $b\_mag$
$exp\_ctr$	0 or 1 Default: 0	Controls computation of the binary point position ( $pos$ output)

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

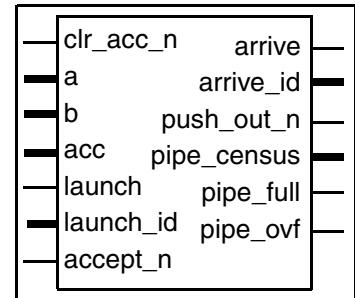
**DW\_piped\_mac**

Pipelined Multiplier-Accumulator

**DW\_piped\_mac**

Pipelined Multiplier-Accumulator

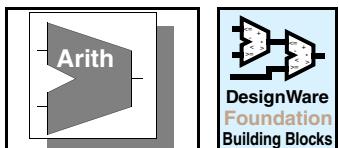
- ❖ Integrated multiply and accumulate
- ❖ Built-in pipeline and power management
- ❖ Parameterized operand widths
- ❖ Parameterized multiply and accumulate output width
- ❖ Parameterized pipeline stages
- ❖ Launch identifier tracking propagation
- ❖ DesignWare datapath generator is employed for better timing and area



Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock source
rst_n	1 bit	Input	Asynchronous reset (active low)
init_n	1 bit	Input	Synchronous reset (active low)
clr_acc_n	1 bit	Input	Clear accumulator results for upcoming product (active low)
a	a_width bit(s)	Input	Multiplier
b	b_width bit(s)	Input	Multiplicand
acc	acc_width bit(s)	Output	Multiply and accumulate result
launch	1 bit	Input	Indicator to begin a new multiply and accumulate
launch_id	id_width bit(s)	Input	Identifier for the corresponding asserted launch
pipe_full	1 bit	Output	Upstream notification that pipeline is full
pipe_ovf	1 bit	Output	Status Flag indicating pipe overflow
accept_n	1 bit	Input	acc result accepted from downstream logic (active low)
arrive	1 bit	Output	acc result is valid
arrive_id	id_width bit(s)	Output	launch_id from the originating launch that produced the 'acc' result
push_out_n	1 bit	Output	Used with external FIFO (optional) to indicate a new acc result has been accepted from the pipeline (active low)
pipe_census	3 bits	Output	Output bus indicates the number of pipe stages currently occupied

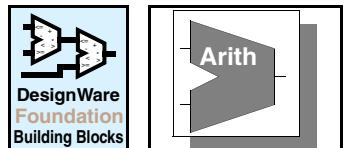
**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\geq 1$ Default: 8	Word length of a
b_width	$\geq 1$ Default: 8	Word length of b
acc_width	$\geq a\_width + b\_width$ Default: 16	Word length of acc
tc	0, 1 Default: 0	Two's complement for internal multiply 0 = unsigned 1 = signed
pipe_reg	0 to 7	Pipeline register insertion Insertion of pipeline register stages. See <a href="#">Table 1-5</a> for settings and their meanings.
id_width	1 to 1024 Default: 8	Launch Identifier width
no_pmt	0, 1 Default: 0	Omit Pipe Manager

Note: † - See section “Omitting Pipe Manager” below.

**Table 1-3 Synthesis Implementations**

Implementation Name	Implementation	License Feature Required
str	Pipelined str synthesis model	DesignWare

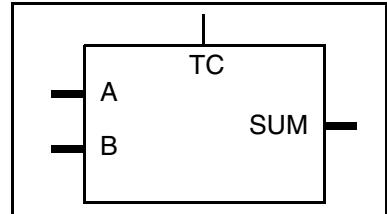
**DW02\_prod\_sum**

Generalized Sum of Products

**DW02\_prod\_sum**

Generalized Sum of Products

- ❖ Parameterized number of inputs
- ❖ Parameterized word length

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width \times num\_inputs$ bit(s)	Input	Concatenated input data
B	$B\_width \times num\_inputs$ bit(s)	Input	Concatenated input data
TC	1 bit	Input	Two's complement 0 = unsigned 1 = signed
SUM	$SUM\_width$ bit(s)	Output	Sum of products

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A
B_width	$\geq 1^a$	Word length of B
num_inputs	$\geq 1$	Number of inputs
SUM_width	$\geq 1$	Word length of SUM

- a. For nbw implementation,  $A\_width+B\_width \leq 36$ . Due to concern of implementation selection run time, a limitation is set for  $A\_width$  and  $B\_width$ .

**Table 1-3 Synthesis Implementations<sup>a</sup>**

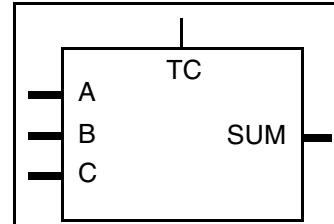
Implementation	Function	License Feature Required
pparch	Delay-optimized flexible Booth Wallace	Designware
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

- 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*.

## DW02\_prod\_sum1

Multiplier-Adder

- ❖ Parameterized word length



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$ bit(s)	Input	Input data
B	$B\_width$ bit(s)	Input	Input data
C	$SUM\_width$ bit(s)	Input	Input data
TC	1 bit	Input	Two's complement 0 = unsigned 1 = signed
SUM	$SUM\_width$ bit(s)	Output	Sum of products

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A
B_width	$\geq 1^a$	Word length of B
SUM_width	$\geq 1$	Word length of C and output SUM

- a. For nbw implementation,  $A\_width+B\_width \leq 36$ . Due to concern of implementation selection run time, a limitation is set for  $A\_width$  and  $B\_width$ .

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	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*.

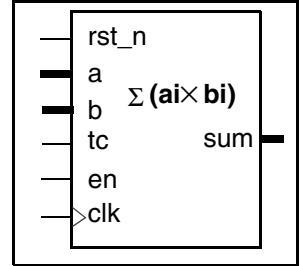
**DW\_prod\_sum\_pipe**

Stallable Pipelined Generalized Sum of Products

**DW\_prod\_sum\_pipe**

Stallable Pipelined Generalized Sum of Products

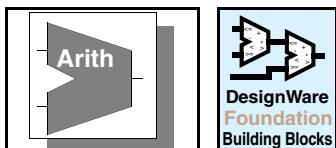
- ❖ Parameterized word length
- ❖ Unsigned and signed (two's complement) data operation
- ❖ Parameterized number of pipeline stages
- ❖ Parameterized stall mode (stallable or non-stallable)
- ❖ Parameterized reset mode (no reset, asynchronous or synchronous reset)
- ❖ Automatic pipeline retiming



Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset, active-low (not used if parameter <i>rst_mode</i> =0)
en	1 bit	Input	Register enable, active high (used only if parameter <i>stall_mode</i> =1) 0 = stall 1 = enable register
tc	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
a	<i>a_width × num_inputs</i> bit(s)	Input	Concatenated input data vector
b	<i>b_width × num_inputs</i> bit(s)	Input	Concatenated input data vector
sum	<i>sum_width</i> bit(s)	Output	Pipelined data summation

**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\geq 1$ Default: None	Word length of a
b_width	$\geq 1$ Default: None	Word length of b
num_inputs	$\geq 1$ Default: 2	Number of inputs
num_stages	$\geq 2$ Default: 2	Number of pipeline stages
stall_mode	0 or 1 Default: 1	Stall mode 0 = non-stallable 1 = stallable
rst_mode	0 to 2 Default: 1	Reset mode 0 = no reset 1 = asynchronous reset 2 = synchronous reset
sum_width	$\geq 48$ Default: None	Word length of sum

**Table 1-3 Synthesis Implementations**

Implementation Name	Implementation	License Feature Required
str <sup>a</sup>	Pipelined str synthesis model	DesignWare

a. Either pparch or apparch implementation is selected based on the constraints of the design.

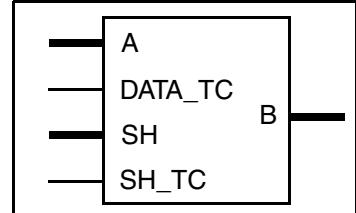
**DW\_rash**

Arithmetic Shifter with Preferred Right Direction

**DW\_rash**

Arithmetic Shifter with Preferred Right Direction

- ❖ Parameterized word length
- ❖ Parameterized shift coefficient width
- ❖ Inferable using a function call

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>A_width</i>	Input	Input data
DATA_TC	1 bit	Input	Data two's complement control 0 = unsigned 1 = signed
SH	<i>SH_width</i>	Input	Shift control
SH_TC	1 bit	Input	Shift two's complement control 0 = unsigned 1 = signed
B	<i>A_width</i>	Output	Output data.

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 2$	Word length of A and B
SH_width	$\geq 1$	Word length of SH

**Table 1-3 Synthesis Implementations<sup>a</sup>**

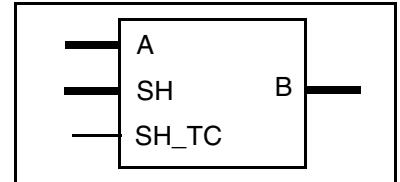
Implementation Name	Function	License Feature Required
str	Synthesis model target for speed	DesignWare
astr	Synthesis model target for area	DesignWare
mx2	Implement using 2:1 multiplexers only. The mx2 implementation is valid only for SH_width values up to and including 31.	none

- a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you can 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*.

## DW\_rbsh

### Barrel Shifter with Preferred Right Direction

- ❖ Parameterized data and shift coefficient word lengths
- ❖ Capable of rotating in both directions
- ❖ Inferable using a function call



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$	Input	Input data
SH	$SH\_width$	Input	Shift control
SH_TC	1 bit	Input	Shift two's complement control 0 = unsigned 1 = signed
B	$A\_width$	Output	Shifted data out

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A and B
SH_width	$\leq \text{ceil}(\log_2[A\_width])$	Word length of SH

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
str	Synthesis model target for speed	DesignWare
astr	Synthesis model target for area	DesignWare
mx2	Implement using 2:1 multiplexers only. The mx2 implementation is valid only for SH_width values up to and including 31.	none

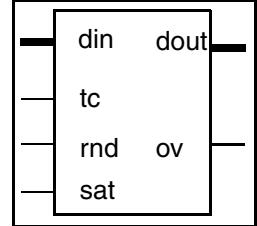
- a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you can force Design Compiler to use one of the architectures described in this table. For more details, please refer to the *DesignWare Building Block IP User Guide*.



## DW01\_satrnd

Arithmetic Saturation and Rounding Logic

- ❖ Parameterized word length
- ❖ Dynamically or statically configurable
- ❖ Arithmetic saturation (clipping) or wrap-around for MSB truncation
- ❖ Round to nearest logic for LSB truncation
- ❖ Signed and unsigned data operation



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
din	<i>width</i> bit(s)	Input	Input data
tc	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
sat	1 bit	Input	Saturation enable 0 = no saturation 1 = enable saturation
rnd	1 bit	Input	Rounding enable 0 = no rounding 1 = enable rounding
ov	1 bit	Output	Overflow status
dout	<i>msb_out - lsb_out + 1</i> bit(s)	Output	Output data

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 2$ Default: 16	Word length of din
msb_out	$width-1 \geq msb\_out > lsb\_out$ Default: 15	dout MSB position after truncation of din MSBs
lsb_out	$msb\_out > lsb\_out \geq 0$ Default: 0	dout LSB position after truncation of din LSBs

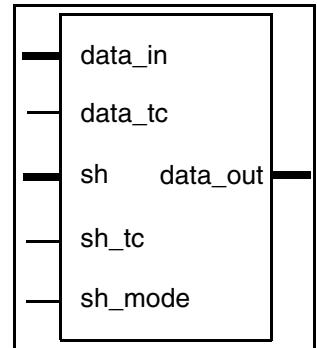
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

# DW\_shifter

## Combined Arithmetic and Barrel Shifter

- ❖ Dynamically selectable arithmetic or barrel shift mode
- ❖ Parameterized input control (inverted and non-inverted logic)
- ❖ Parameterized padded logic value control (for arithmetic shift only)
- ❖ Parameterized data and shift coefficient word lengths
- ❖ Inferable using a function call (support for `inv_mode = 0` only)



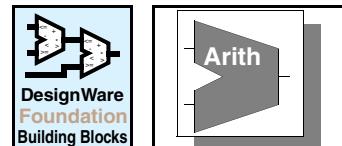
**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data_in	<i>data_width</i> bit(s)	Input	Input data
data_tc	1 bit	Input	Two's complement control on data_in 0 = unsigned data_in 1 = signed data_in
sh	<i>sh_width</i> bit(s)	Input	Shift control
sh_tc	1 bit	Input	Two's complement control on sh 0 = unsigned sh 1 = signed sh
sh_mode	1 bit	Input	Arithmetic or barrel shift mode 0 = barrel shift mode 1 = arithmetic shift mode
data_out	<i>data_width</i> bit(s)	Output	Output data

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	$\geq 2$	Word length of data_in and data_out
sh_width	1 to $(\text{ceil}(\log_2[\text{data\_width}]) + 1)$	Word length of sh
inv_mode	0 to 3 Default: 0	logic mode 0 = normal input, 0 padding in output; 1 = normal input, 1 padding in output; 2 = inverted input <sup>a</sup> , 0 padding in output; 3 = inverted input, 1 padding in output

a. Inverted input refers to sh, sh\_tc, and data\_tc pins only.

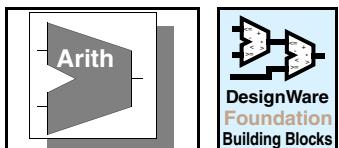
**DW\_shifter**

Combined Arithmetic and Barrel Shifter

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation	Function	License Feature Required
mx2	Implement using 2:1 multiplexers only.	DesignWare
mx2i	Synthesis model	DesignWare
mx4	Synthesis model	DesignWare
mx8	Synthesis model	DesignWare

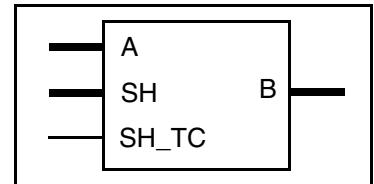
- a. During synthesis, Design Compiler will select the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table. For more details, please refer to the *DesignWare Building Block IP User Guide*.



## DW\_sla

### Arithmetic Shifter with Preferred Left Direction (VHDL Style)

- ❖ Parameterized data and shift coefficient word lengths
- ❖ Uses VHDL semantics for the arithmetic shift operation
- ❖ Capable of shifting in both directions



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>A_width</i>	Input	Input data
SH	<i>SH_width</i>	Input	Shift control
SH_TC	1 bit	Input	Shift two's complement control 0 = unsigned 1 = signed
B	<i>A_width</i>	Output	Shifted data out

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 2$	Word length of A and B
SH_width	$\geq 1$	Word length of SH

**Table 1-3 Synthesis Implementations<sup>a</sup>**

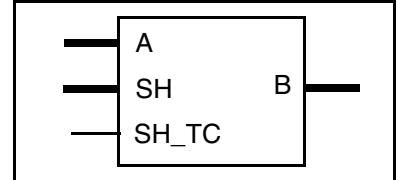
Implementation Name	Function	License Feature Required
str	Synthesis model targeted for speed	DesignWare
astr	Synthesis model targeted for area	DesignWare
mx2	Static implement using 2:1 multiplexers only.	DesignWare

- a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you can 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.

**DW\_sra**

Arithmetic Shifter with Preferred Right Direction (VHDL Style)

- ❖ Parameterized data and shift coefficient word lengths
- ❖ Capable of rotating in both directions
- ❖ Inferable using a function call

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$A\_width$	Input	Input data
SH	$SH\_width$	Input	Shift control
SH_TC	1 bit	Input	Shift two's complement control 0 = unsigned 1 = signed
B	$A\_width$	Output	Shifted data out

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	$\geq 1$	Word length of A and B
SH_width	$\leq \text{ceil}(\log_2[A\_width])$	Word length of SH

**Table 1-3 Synthesis Implementations<sup>a</sup>**

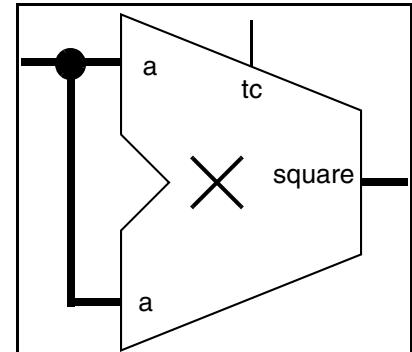
Implementation Name	Function	License Feature Required
str	Synthesis model target for speed	DesignWare
astr	Synthesis model target for area	DesignWare
mx2	Implement using 2:1 multiplexers only. The mx2 implementation is valid only for SH_width values up to and including 31.	none

- a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you can 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*.

## DW\_square

### Integer Squarer

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's complement) data operation
- ❖ Inferable using a function call
- ❖ Optional C++ simulation model available in the DWFC Library



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>width</i> bit(s)	Input	Input data
tc	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
square	$2 \times \text{width}$ bit(s)	Output	Product of $(a \times a)$

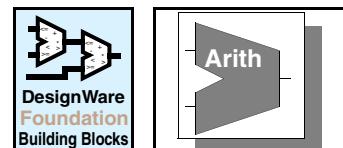
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$ bits Default: 8	Word length of a

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
pparch <sup>a</sup>	Delay-optimized flexible Booth Wallace	DesignWare
apparch <sup>a</sup>	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

- a. The 'pparch' (optimized for delay) and 'apparch' (optimized for area) implementations are dynamically generated to best meet your constraints. The 'pparch' and 'apparch' implementations can generate a variety of multiplier (squarer) architectures including Radix-2 non-Booth, Radix-4 non-Booth, Radix-4 Booth recoded and Radix-8 Booth recoded. The 'pparch' and 'apparch' implementations are generated making use of any special arithmetic technology cells that are found to be available in your target technology library. The `dc_shell` command, `set_dp_smartgen_options`, can be used to force specific multiplier (squarer) architectures. For more information on forcing generated arithmetic architectures, use '`man set_dp_smartgen_options`' (in `dc_shell`) to get a listing of the command options.

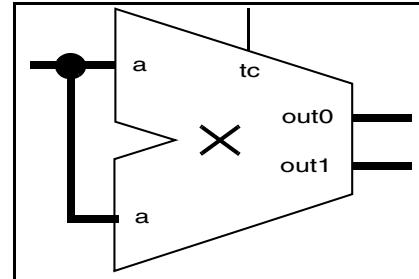
**DW\_squarep**

Partial Product Integer Squarer

**DW\_squarep**

Partial Product Integer Squarer

- ❖ Parameterized word lengths
- ❖ Unsigned and signed (two's-complement) data operation
- ❖ Optional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>width</i> bit(s)	Input	Multiplier
tc	1 bit	Input	Two's complement control 0 = unsigned 1 = signed
out0	<i>width</i> × 2 bit(s)	Output	Partial product of a × a
out1	<i>width</i> × 2 bit(s)	Output	Partial product of a × a

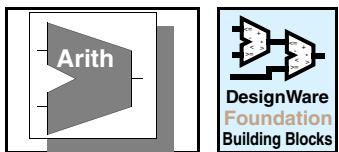
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Word length of signal a
verif_en <sup>a</sup>	0 to 3 Default: 1	Verification Enable Control 0 - out0 and out1 are always the same for a given input 1 - MSB of out0 is always '0'. Out0 and out1 change with time for the same input 2 - MSB of out0 or out1 is always '0'. Out0 and out1 change with time for the same input 3 - no restrictions on MSBs of out0 and out1. Out0 and out1 change with time for the same input

- a. Although the *verif\_en* value can be set for all simulators, CS randomization is only implemented when using Synopsys simulators (VCS, VCS-MX). For more information about *verif\_en*, refer to “[Simulation Using Random Carry-save Representation \(VCS/VCS-MX only\)](#)” on page 3.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
pparch <sup>a</sup>	Delay-optimized flexible parallel-prefix	DesignWare
apparch <sup>b</sup>	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	DesignWare

**DW\_squarep**  
Partial Product Integer Squarer

- a. The 'pparch' (optimized for delay) and 'apparch' (optimized for area) implementations are dynamically generated to best meet your constraints. The 'pparch' and 'apparch' implementations can generate a variety of multiplier (squarer) architectures including Radix-2 non-Booth, Radix-4 non-Booth, Radix-4 Booth recoded and Radix-8 Booth recoded. The 'pparch' and 'apparch' implementations are generated making use of any special arithmetic technology cells that are found to be available in your target technology library. The `dc_shell` command, `set_dp_smartgen_options`, can be used to force specific multiplier (squarer) architectures. For more information on forcing generated arithmetic architectures, use '`man set_dp_smartgen_options`' (in `dc_shell`) to get a listing of the command options.

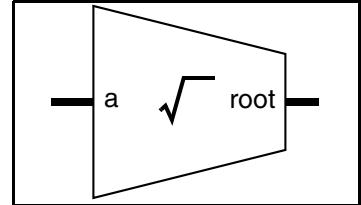
**DW\_sqrt**

Combinational Square Root

**DW\_sqrt**

Combinational Square Root

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's complement) square root computation
- ❖ Inferable using a function call

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>width</i> bit(s)	Input	Radicand
root	int([ <i>width</i> +1]/2) bit(s)	Output	Square root

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 2$	Word length of a
tc_mode	0 or 1 Default: 0	Two's complement control 0 = unsigned 1 = signed

**Table 1-3 Synthesis Implementations<sup>a</sup>**

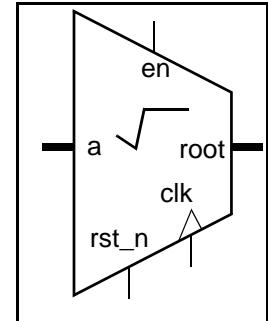
Implementation Name	Function	License Feature Required
rpl	Restoring ripple-carry synthesis model	DesignWare
cla	Restoring carry-lookahead synthesis model	DesignWare

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*

## DW\_sqrt\_pipe

Stallable Pipelined square root

- ❖ Parameterized word length
- ❖ Unsigned and signed (two's complement) data operation
- ❖ Parameterized number of pipeline stages
- ❖ Parameterized stall mode (stallable or non-stallable)
- ❖ Parameterized reset mode (no reset, asynchronous or synchronous reset)
- ❖ Automatic pipeline retiming



Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset, active-low (not used if parameter <i>rst_mode</i> =0)
en	1 bit	Input	Register enable, active high (used only if parameter <i>stall_mode</i> =1) 0 = stall 1 = enable register
a	<i>width</i> bit(s)	Input	Radicand
root	( <i>width</i> +1)/2 bit(s)	Output	Square root

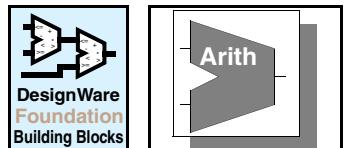
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 2$	Word length of a. Default: None
num_stages	$\geq 2$	Number of pipeline stages. Default: 2
stall_mode	0 or 1 Default: 1	Stall mode 0 = non-stallable, 1 = stallable
rst_mode	0 to 2 Default: 1	Reset mode 0 = no reset, 1 = asynchronous reset, 2 = synchronous reset)
tc_mode	0 or 1 Default: 0	Two's Complement mode 0 = unsigned number, 1 = signed number

**Table 1-3 Synthesis Implementations**

Implementation Name	Implementation	License Feature Required
str <sup>a</sup>	Pipelined str synthesis model	DesignWare

a. One of rpl or cla implementation is selected based the constraints of the design.

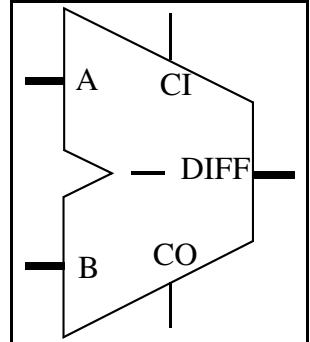
**DW01\_sub**

Subtractor

**DW01\_sub**

Subtractor

- ❖ Parameterized word length
- ❖ Carry-in and carry-out signals

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data
B	<i>width</i> bit(s)	Input	Input data
CI	1 bit	Input	Carry-in
DIFF	<i>width</i> bit(s)	Output	Difference of A –B –CI
CO	1 bit	Output	Carry-out

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Word length of A, B, and DIFF

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	none
cla	Carry-look-ahead synthesis model	none
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	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*.

## DW02\_sum

Vector Adder

- ❖ Parameterized number of inputs
- ❖ Parameterized word length
- ❖ Multiple synthesis implementations
- ❖ Inferable using a function call



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
INPUT	$num\_inputs \times input\_width$ bit(s)	Input	Concatenated input data
SUM	$input\_width$ bit(s)	Output	Sum

**Table 1-2 Parameter Description**

Parameter	Values	Description
num_inputs	$\geq 1$	Number of inputs
input_width	$\geq 1$	Word length of inputs and sum

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	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.

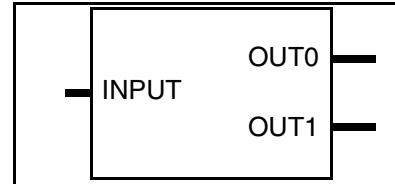
**DW02\_tree**

Wallace Tree Compressor

**DW02\_tree**

Wallace Tree Compressor

- ❖ Parameterized word length

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
INPUT	<i>num_inputs</i> × <i>input_width</i> bit(s)	Input	Input vector
OUT0	<i>input_width</i> bit(s)	Output	Partial sum
OUT1	<i>input_width</i> bit(s)	Output	Partial sum

**Table 1-2 Parameter Description**

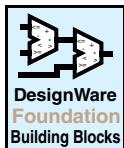
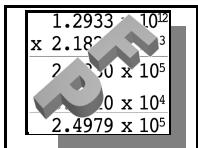
Parameter	Values	Description
num_inputs	≥ 1	Number of inputs
input_width	≥ 1	Word length of OUT0 and OUT1
verif_en <sup>a</sup>	0 or 1 Default: 1	Verification Enable Control 0 - out0 and out1 are always the same for a given input value 1 - out0 and out1 change with time for a given input value

- a. Although the *verif\_en* value can be set for all simulators, CS randomization is only implemented when using Synopsys simulators (VCS, VCS-MX). For more information about *verif\_en*, refer to "[Simulation Using Random Carry-save Representation \(VCS/VCS-MX only\)](#)" on page 3

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
pparch	Delay-optimized flexible parallel-prefix	DesignWare
apparch	Area-optimized flexible architecture that can be optimized for area, for speed, or for area, speed	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*.



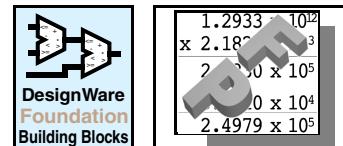
## Datapath – Floating Point Overview

The Floating Point components comprise a library of functions used to synthesize floating point computational circuits in high end ASICs. The functions mainly deal with arithmetic operations in floating point format, format conversions and comparison functions. The main features of this library are as follows:

- ❖ The format of the floating point numbers that determines the precision of the number that it represents is parameterizable. The user can select the precision based on the number of bits in the exponent and significand (or mantissa). The parameters cover all the IEEE formats.
- ❖ Accuracy conforms to the definitions in the IEEE 754 Floating Point standard for the basic arithmetic operations. Improved accuracy is obtained with multi-operand FP components.

Download instructions for the Floating Point components can be found at the following web address:

[http://www.synopsys.com/dw/buildingblock\\_dl.php](http://www.synopsys.com/dw/buildingblock_dl.php)

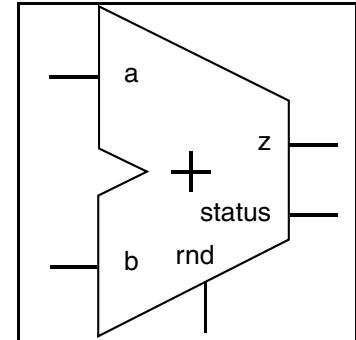
**DW\_fp\_add**

Floating Point Adder

**DW\_fp\_add**

Floating Point Adder

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 253 bits
- ❖ Fully compatible with the IEEE 754 Floating-point standard with proper set of parameters
- ❖ DesignWare datapath generator is employed for better timing and area
- ❖ Optional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

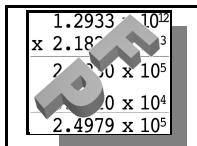
Pin Name	Width	Direction	Function
a	sig_width+exp_width+1 bits	Input	Input data
b	sig_width+exp_width+1 bits	Input	Input data
z	sig_width+exp_width+1 bits	Output	a + b
status	8 bits	Output	Status flags
rnd	3 bits	Input	Rounding mode

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers a, b, and z
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers a, b, and z
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

**Table 1-3 Synthesis Implementations**

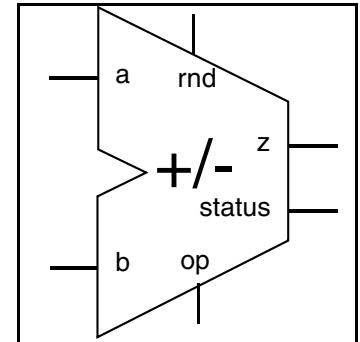
Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



## DW\_fp\_addsub

### Floating Point Adder/Subtractor

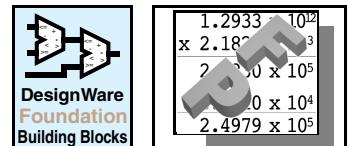
- ❖ The precision is controlled by parameters, and covers formats in the IEEE standard
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 253 bits
- ❖ Fully compatible with the IEEE 754 Floating-point standardDesignWare datapath generator is employed for better power and QoR
- ❖ Optional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>sig_width+exp_width+1</i> bits	Input	Input data
b	<i>sig_width+exp_width+1</i> bits	Input	Input data
op	1 bit	Input	Defines the operation: 0 - addition 1 - subtraction
z	<i>sig_width+exp_width+1</i> bits	Output	$a \text{ op } b$
status	8 bits	Output	Status flags
rnd	3 bits	Input	Rounding mode

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers a, b, and z
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers a, b, and z
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

**DW\_fp\_addsub**

Floating Point Adder/Subtractor

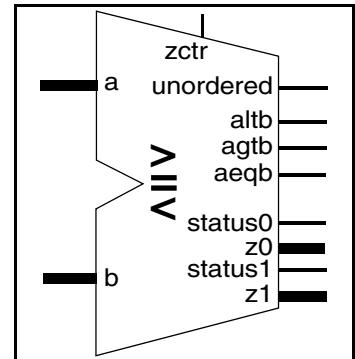
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Area-optimized synthesis model	DesignWare
str	Delay-optimized synthesis model	DesignWare

## DW\_fp\_cmp

### Floating Point Comparator

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Exponents can range from 3 to 31 bits
- ❖ Significand and fractional part of the floating-point number can range from 2 to 256 bits
- ❖ Accuracy conforms to IEEE 754 Floating-point standardOptional C++ simulation model available in the DWFC Library

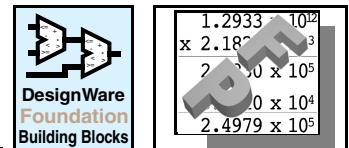


**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>sig_width+exp_width +1</i> bits	Input	Floating-point number
b	<i>sig_width+exp_width +1</i> bits	Input	Floating-point number
altb	1 bit	Output	High when a is less than b
agtb	1 bit	Output	High when a is greater than b
aeqb	1 bit	Output	High when a is equal to b
unordered	1 bit	Output	High when one of the inputs is NaN and <i>ieee_compliance</i> = 1
z0	<i>sig_width+exp_width +1</i> bits	Output	Min(a,b) when zctr=0 or open, otherwise, Max(a,b)
z1	<i>sig_width+exp_width +1</i> bits	Output	Max(a,b) when zctr=0 or open, otherwise, Min(a,b)
status0	8 bits	Output	Status flags corresponding to z0
status1	8 bits	Output	Status flags corresponding to z1
zctr	1 bit	Input	Determines value passed to z0 and z1

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers a, b, z0, and z1.
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers a, b, z0, and z1.

**DW\_fp\_cmp****Table 1-2 Parameter Description**

Parameter	Values	Description
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

## DW\_fp\_div

### Floating Point Divider

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Hardware for denormal numbers of IEEE 754 standard is provided.
- ❖ Fully compatible with the IEEE 754 Floating-point standard with proper set of parameters
- ❖ Faithful rounding with 1 ulp error is supported by parameter
- ❖ DesignWare datapath generator is employed for better timing and area
- ❖ Optional C++ simulation model available in the DWFC Library

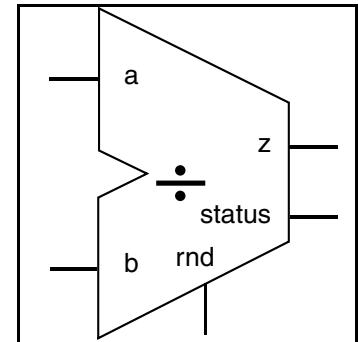
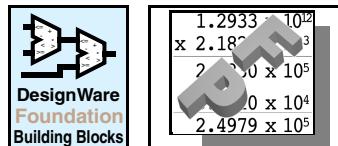


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	exp_width + sig_width + 1 bits	Input	Dividend
b	exp_width + sig_width + 1 bits	Input	Divisor
rnd	3 bits	Input	Rounding mode
z	exp_width + sig_width + 1 bits	Output	Quotient of A/B
status	8 bits	Output	Status flags

Table 1-2 Parameter Description

Parameter	Values	Description
sig_width	4 to 253 bits Default: 23	Word length of fraction field of floating-point numbers a, b, and z
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers a, b, and z
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture includes the use of denormals and NaNs
faithful_round	0 or 1 Default: 0	Select the faithful rounding that admits maximum 1 ulp error. 0 : It support all rounding modes described in <a href="#">Datapath - Floating-point Overview</a> . 1 : results have 1 ulp error (see Table 5).



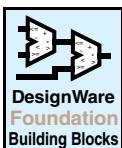
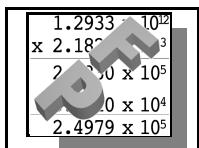
## DW\_fp\_div

### Floating Point Divider

Table 1-3 Synthesis Implementations<sup>a</sup>

Implementation Name	Function	License Feature Required
rtl	Synthesis model (Digit-recurrence Method)	DesignWare
str	Synthesis model (Multiplicative Method)	DesignWare

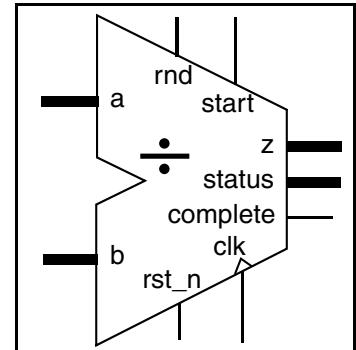
a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you can force Design Compiler to use one of the architectures described in this table. For more details, please refer to the *DesignWare Building Block IP User Guide*.



## DW\_fp\_div\_seq

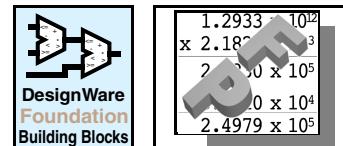
### Floating Point Sequential Divider

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Hardware for denormal numbers of IEEE 754 standard is selectively provided
- ❖ Accuracy conforms to IEEE 754 Floating-point standard
- ❖ Parameterized number of clock cycles
- ❖ Registered or un-registered input and outputs
- ❖ Internal register for the partial pipelining
- ❖ DesignWare datapath generator is employed for better timing and area.
- ❖ Provides minPower benefits with the DesignWare-LP license.
- ❖ Optional C++ simulation model available in the DWFC Library



**Table 1-1 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

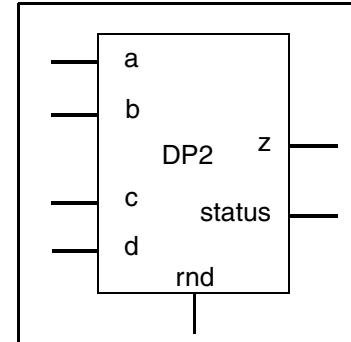
**DW\_fp\_dp2**

2-Term Floating Point Dot-product

**DW\_fp\_dp2**

2-Term Floating Point Dot-product

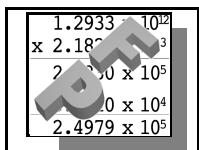
- ❖ The precision is controlled by parameters, and covers formats in the IEEE standard
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 253 bits
- ❖ A parameter controls the use of denormal values
- ❖ More accurate than using a combination of basic FP operators.
- ❖ Provides a variety of rounding modes.
- ❖ DesignWare datapath generator is employed for reduced delay and area.
- ❖ Optional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

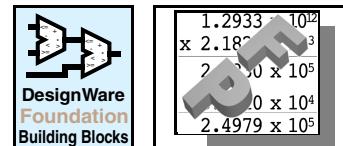
Pin Name	Width	Direction	Function
a	<i>sig_width+exp_width+1</i> bits	Input	FP input data
b	<i>sig_width+exp_width+1</i> bits	Input	FP input data
c	<i>sig_width+exp_width+1</i> bits	Input	FP input data
d	<i>sig_width+exp_width+1</i> bits	Input	FP input data
rnd	3 bits	Input	Rounding mode
z	<i>sig_width+exp_width+1</i> bits	Output	$a * b + c * d$
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers a, b, c, d, and z
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers a, b, c, d, and z
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.
arch_type	0 or 1 Default: 0	Controls the use of an alternative architecture. Default value is 0 (previous architecture).

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

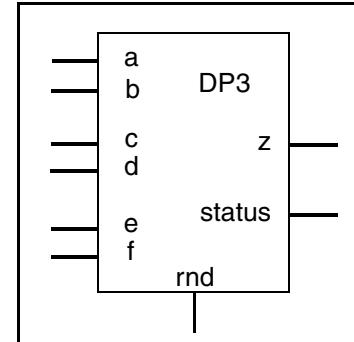
**DW\_fp\_dp3**

3-Term Floating Point Dot-product

**DW\_fp\_dp3**

3-Term Floating Point Dot-product

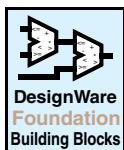
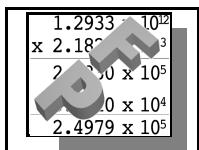
- ❖ The precision is controlled by parameters, and covers formats in the IEEE standard
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 253 bits
- ❖ A parameter controls the use of denormal values
- ❖ More accurate than using a combination of basic FP operators.
- ❖ Provides a variety of rounding modes.
- ❖ DesignWare datapath generator employed for reduced delay and area.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>sig_width+exp_width+1</i> bits	Input	FP input data
b	<i>sig_width+exp_width+1</i> bits	Input	FP input data
c	<i>sig_width+exp_width+1</i> bits	Input	FP input data
d	<i>sig_width+exp_width+1</i> bits	Input	FP input data
e	<i>sig_width+exp_width+1</i> bits	Input	FP input data
f	<i>sig_width+exp_width+1</i> bits	Input	FP input data
rnd	3 bits	Input	Rounding mode
z	<i>sig_width+exp_width+1</i> bits	Output	(a*b)+(c*d)+(e*f)
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits	Word length of fraction field of floating-point numbers a, b, c, d, e, f, and z
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers a, b, c, d, e, f, and z
ieee_compliance	0 or 1	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.
arch_type	0 or 1	Controls the use of an alternative architecture. Default: 0 (previous arch).

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

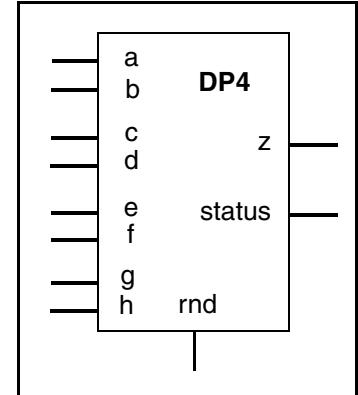
**DW\_fp\_dp4**

4-Term Floating Point Dot-product

**DW\_fp\_dp4**

4-Term Floating Point Dot-product

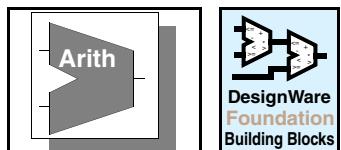
- ❖ The precision is controlled by parameters, and covers formats in the IEEE standard
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 253 bits
- ❖ A parameter controls the use of denormal values
- ❖ More accurate than using a combination of basic FP operators.
- ❖ Provides a variety of rounding modes.
- ❖ DesignWare datapath generator is employed for reduced delay and area.

**Table 1-1 Pin Description**

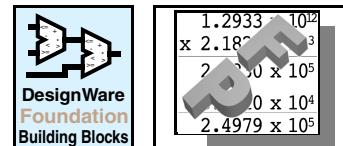
Pin Name	Width	Direction	Function
a	<i>sig_width+exp_width+1</i> bits	Input	FP input data
b	<i>sig_width+exp_width+1</i> bits	Input	FP input data
c	<i>sig_width+exp_width+1</i> bits	Input	FP input data
d	<i>sig_width+exp_width+1</i> bits	Input	FP input data
e	<i>sig_width+exp_width+1</i> bits	Input	FP input data
f	<i>sig_width+exp_width+1</i> bits	Input	FP input data
g	<i>sig_width+exp_width+1</i> bits	Input	FP input data
h	<i>sig_width+exp_width+1</i> bits	Input	FP input data
rnd	3 bits	Input	Rounding mode
z	<i>sig_width+exp_width+1</i> bits	Output	$(a*b)+(c*d)+(e*f)+(g*h)$
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits	Word length of fraction field of floating-point numbers a, b, c, d, e, f, g, h, and z
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers a, b, c, d, e, f, g, h, and z
ieee_compliance	0 or 1	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.
arch_type	0 or 1	Controls the use of an alternative architecture. Default value is 0 (previous architecture).

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

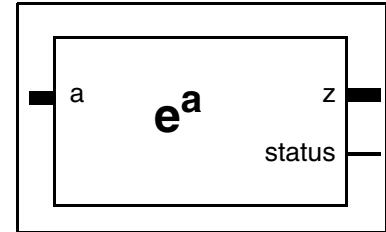
**DW\_fp\_exp**

Floating Point Exponential

**DW\_fp\_exp**

Floating Point Exponential

- ❖ The precision is controlled by parameters, and covers formats in the IEEE Standard 754
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 60 bits
- ❖ A parameter controls the use of denormal values.

**Table 1-1 Pin Description**

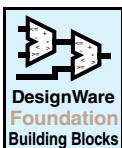
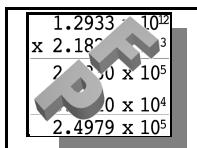
Pin Name	Width	Direction	Function
a	<i>sig_width+exp_width+1</i> bits	Input	Input data
z	<i>sig_width+exp_width+1</i> bits	Output	Exponential = $e^a$
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 60 bits	Word length of fraction field of floating-point numbers a and z.
exp_width	3 to 31	Word length of biased exponent of floating-point numbers a and z.
ieee_compliance	0 or 1 Default: 0	Controls the use of denormals and NaNs: 0 - don't use denormals or NaNs 1 - use denormals and NaNs
arch	0 to 2 Default: 2	Implementation selection. 0 - area optimized 1 - speed optimized 2 - uses 2007.12 sub-components

**Table 1-3 Synthesis Implementations**

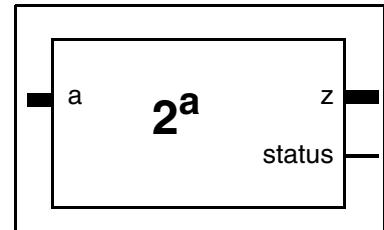
Implementation Name	Function	License Feature Required
rtl	Implement using the Datapath Generator technology combined with static DesignWare components	DesignWare



## DW\_fp\_exp2

### Floating Point Base 2 Exponential

- ❖ The precision is controlled by parameters, and covers formats in the IEEE Standard 754
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 60 bits
- ❖ A parameter controls the use of denormal values.

**Table 1-1 Pin Description**

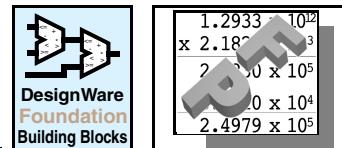
Pin Name	Width	Direction	Function
a	<i>sig_width+exp_width+1</i> bits	Input	Input data
z	<i>sig_width+exp_width+1</i> bits	Output	Base-2 exponential = $2^a$
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 60 bits	Word length of fraction field of floating-point numbers a and z.
exp_width	3 to 31	Word length of biased exponent of floating-point numbers a and z.
ieee_compliance	0 or 1	When 1, the generated architecture is capable of dealing with denormals and NaNs.
arch	0 to 2 Default: 2	Implementation selection 0 - area optimized 1 - speed optimized 2 - implementation released in 2007.12

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Implement using the Datapath Generator technology combined with static DesignWare components	DesignWare

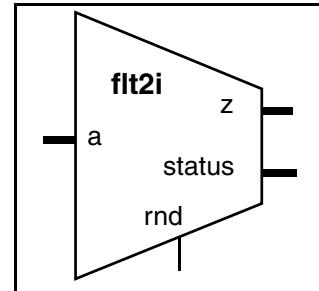
**DW\_fp\_flt2i**

Floating Point-to-Integer Converter

**DW\_fp\_flt2i**

Floating Point-to-Integer Converter

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Accuracy conforms to IEEE 754 Floating-point standard
- ❖ DesignWare datapath generator is employed for better timing and area
- ❖ Optional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

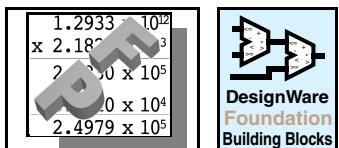
Pin Name	Width	Direction	Function
a	<i>sig_width + exp_width + 1</i> bits	Input	Floating-point number
rnd	3 bits	Input	Rounding mode
z	<i> isize</i> bits	Output	Two's complement integer number
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 Default: 23	Word length of fraction field of floating-point number a
exp_width	3 to 31 Default: 8	Word length of biased exponent of floating-point number a
isize	3 to 512 Default: 32	Word length of converted integer number z
ieee_compliance	0 or 1 Default: 0	When 1, it recognizes NaN, Inf and Denormal inputs.

**Table 1-3 Synthesis Implementations**

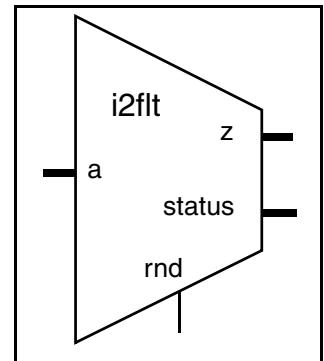
Implementation Name	Function	License Feature Required
rtl	Fast Synthesis model	DesignWare



## DW\_fp\_i2flt

### Integer-to-Floating Point Converter

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Accuracy conforms to IEEE 754 Floating-point standard
- ❖ DesignWare datapath generator is employed for better timing and area
- ❖ Optional C++ simulation model available in the DWFC Library



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	<i>isize</i> bits	Input	Signed/unsigned integer number
rnd	3 bits	Input	Rounding mode
z	<i>sig_width + exp_width + 1</i> bits	Output	Floating-point number
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 Default: 23	Word length of fraction field of floating-point number, z
exp_width	3 to 31 Default: 8	Word length of biased exponent of floating-point number, z
isize	(3+ <i>isign</i> ) ≤ <i>isize</i> ≤ 512 Default: 32	Word length of integer number, a
isign	0 or 1 Default: 1	0 : unsigned integer number, 1 : signed two's complement integer number

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Area-optimized synthesis model	DesignWare
str	Delay-optimized synthesis model	DesignWare

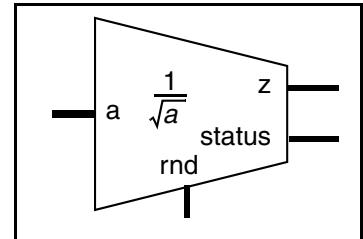
**DW\_fp\_invsqrt**

Floating Point Reciprocal of Square Root

**DW\_fp\_invsqrt**

Floating Point Reciprocal of Square Root

- ❖ The floating-point format is controlled by parameters, and covers formats in the IEEE standard
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 253 bits
- ❖ A parameter controls the use of denormal values
- ❖ Provides a variety of rounding modes
- ❖ Accuracy conforms to IEEE 754 Floating-point standard
- ❖ DesignWare datapath generator is employed for better timing and area

**Table 1-1 Pin Description**

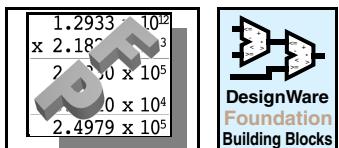
Pin Name	Width	Direction	Function
a	sig_width + exp_width + 1 bits	Input	FP Input data
rnd	3 bits	Input	Rounding mode
z	sig_width + exp_width + 1 bits	Output	FP output data
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits	Word length of fraction field of floating-point numbers $a$ and $z$
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers $a$ and $z$
ieee_compliance	0 or 1	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

**Table 1-3 Synthesis Implementations**

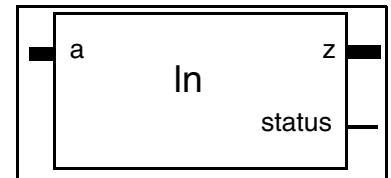
Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



## DW\_fp\_In

### Floating Point Natural Logarithm

- ❖ The precision is controlled by parameters, and covers formats in the IEEE Standard 754
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 60 bits
- ❖ A parameter controls the use of denormal values



**Table 1-1 Pin Description**

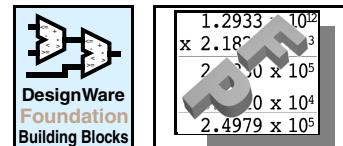
Pin Name	Width	Direction	Function
a	<i>sig_width+exp_width+1</i> bits	Input	Input data
z	<i>sig_width+exp_width+1</i> bits	Output	Logarithm value ln(a)
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 60 bits	Word length of fraction field of floating-point numbers a and z
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers a and z
ieee_compliance	0 or 1	Controls the use of denormals and NaNs: 0 - don't use denormals or NaNs 1 - use denormals and NaNs
extra_prec	0 to $(60-sig\_width)$ Default: 0	Internal extra precision (in bits) used in the computation of the FP output.
arch	0 or 1 Default: 0	Implementation selection. 0 - area optimized 1 - speed optimized

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Implementation using the Datapath Generator technology combined with static DesignWare components	DesignWare

**DW\_fp\_log2**

Floating Point Base-2 Logarithm

**DW\_fp\_log2**

Floating Point Base-2 Logarithm

- ❖ The precision is controlled by parameters, and covers formats in the IEEE Standard 754
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating point number can range from 2 to 60 bits
- ❖ A parameter controls the use of denormal values

**Table 1-1 Pin Description**

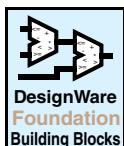
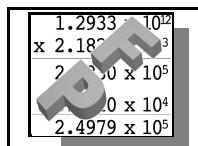
Pin Name	Width	Direction	Function
a	sig_width+exp_width+1 bits	Input	Input data.
z	sig_width+exp_width+1 bits	Output	$\log_2(a)$
status	8-bits	Output	Status flags.

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 60 bits	Word length of fraction field of floating-point numbers <i>a</i> and <i>z</i>
exp_width	3 to 31 bits	Width of input and output data buses
ieee_compliance	0 or 1	When 1 the generated architecture is capable of dealing with denormals and NaNs.
extra_prec	0 to (60-sig_width) Default: 0	Internal extra precision (in bits) used in the computation of the FP output.
arch	0 to 2 Default: 2	Implementation selection. 0 - area optimized 1 - speed optimized 2 - implementation released in 2007.12

**Table 1-3 Synthesis Implementations**

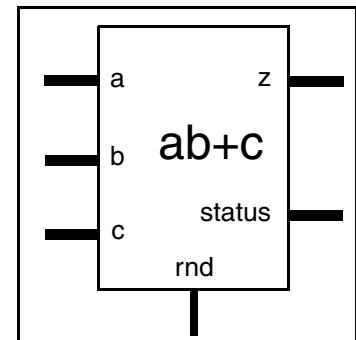
Implementation Name	Function	License Feature Required
rtl	Implementation using the Datapath Generator technology combined with static DesignWare components	DesignWare



## DW\_fp\_mac

### Floating Point Multiply and Add

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Hardware for denormal numbers of IEEE 754 standard is selectively provided.
- ❖ Accuracy conforms to IEEE 754 Floating-point standardOptional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

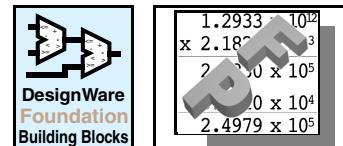
Pin Name	Width	Direction	Function
a	exp_width + sig_width + 1 bits	Input	Multiplier
b	exp_width + sig_width + 1 bits	Input	Multiplicand
c	exp_width + sig_width + 1 bits	Input	Addend
rnd	3 bits	Input	Rounding mode
z	exp_width + sig_width + 1 bits	Output	MAC result ( $a \times b + c$ )
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers a, b, c, and z
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers a, b, c and z
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture includes the use of denormals and NaNs.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Area-optimized synthesis model	DesignWare
str	Delay-optimized synthesis model	DesignWare



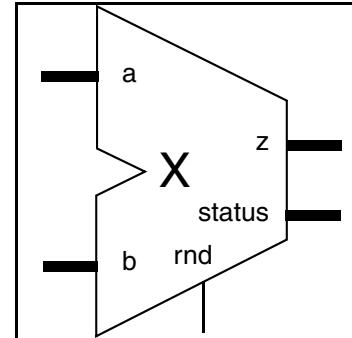
## DW\_fp\_mult

### Floating Point Multiplier

## DW\_fp\_mult

### Floating Point Multiplier

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Hardware for denormal numbers of IEEE 754 standard is selectively provided.
- ❖ Fully compatible with the IEEE 754 Floating-point standard
- ❖ DesignWare datapath generator is employed for better timing and area
- ❖ Optional C++ simulation model available in the DWFC Library



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
a	exp_width + sig_width + 1 bits	Input	Multiplier
b	exp_width + sig_width + 1 bits	Input	Multiplicand
rnd	3 bits	Input	Rounding mode
z	exp_width + sig_width + 1 bits	Output	Product of a X b
status	8 bits	Output	Status flags

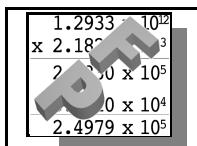
**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers a, b, and z
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers a, b, and z
ieee_compliance	0 or 1 Default: 0	IEEE 754 standard support. 0 : MC (Module Compiler) compatible 1 : IEEE 754 standard compatible, including NaNs and denormal expressions

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare
str <sup>a</sup>	Synthesis model -- delay optimized for non-ieee_compliance	DesignWare

a. Only available when parameter ieee\_compliance is 0.



## DW\_fp\_recip

### Floating Point Reciprocal

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Hardware for denormal numbers of IEEE 754 standard is selectively provided.
- ❖ Both IEEE 754 standard rounding modes and the faithful rounding with 1 ulp error are supported.
- ❖ Accuracy conforms to IEEE 754 Floating-point standard

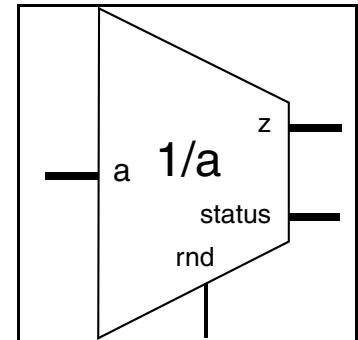


Table 1-1 Pin Description

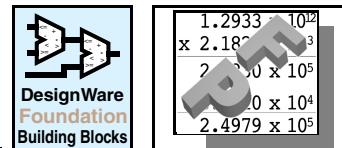
Pin Name	Width	Direction	Function
a	exp_width + sig_width + 1 bits	Input	Divisor
rnd	3 bits	Input	Rounding mode (rnd takes effect only when faithful_round = 0)
z	exp_width + sig_width + 1 bits	Output	Quotient of 1/a
status	8 bits	Output	Status flags

Table 1-2 Parameter Description

Parameter	Values	Description
sig_width	2 to 60 bits	Word length of fraction field of floating-point numbers $a$ , and $z$
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers $a$ , and $z$
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture includes the use of denormals and NaNs.
faithful_round	0 or 1 Default: 0	Select the faithful rounding that admits maximum 1 ulp error. 0 : Support all rounding modes described in <a href="#">Datapath – Floating-point Overview</a> . 1 : Results have 1 ulp error.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

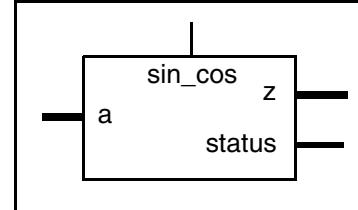
**DW\_fp\_sincos**

Floating Point Sine and Cosine

**DW\_fp\_sincos**

Floating Point Sine and Cosine

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Hardware for denormal numbers of IEEE 754 standard is selectively provided.

**Table 1-1 Pin Description**

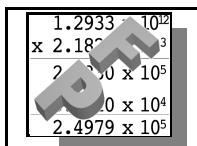
Pin Name	Width	Direction	Function
a	exp_width + sig_width + 1 bits	Input	Input data (radian)
sin_cos	1 bit	Input	Function select: 0 = sine , 1 = cosine
z	exp_width + sig_width + 1 bits	Output	Sine or cosine value
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 33 bits	Word length of fraction field of floating-point numbers a and z
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers a, z
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.
pi_multiple	0 or 1 Default: 1	Angle is multiplied by $\pi$ 0 : z = sin(a) or cos(a) 1 : z = sin( $\pi$ a) or cos( $\pi$ a)
arch	0 or 1 Default: 0	Selection of optimized implementation 0: area-optimized implementation 1: speed-optimized implementation
err_range	1 or 2 Default: 1	Error range selection 1 :   true value - calculated   < 1 ulp 2 :   true value - calculated   < 2 ulp

**Table 1-3 Synthesis Implementations**

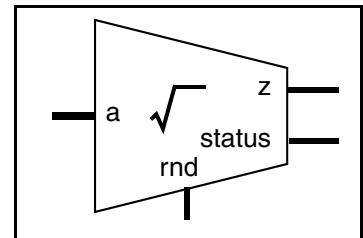
Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



## DW\_fp\_sqrt

### Floating Point Square Root

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Hardware for denormal numbers of IEEE 754 standard is selectively provided.
- ❖ DesignWare datapath generator is employed for better timing and area

**Table 1-1 Pin Description**

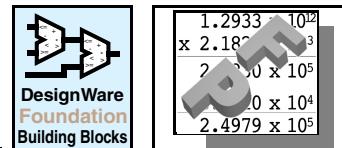
Pin Name	Width	Direction	Function
a	exp_width + sig_width + 1 bits	Input	Input data
rnd	3 bits	Input	Rounding mode
z	exp_width + sig_width + 1 bits	Output	Square root of a
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits	Word length of fraction field of floating-point numbers a and z
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers a and z
ieee_compliance	0 or 1	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



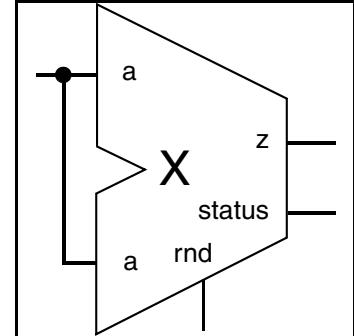
## DW\_fp\_square

### Floating Point Square

## DW\_fp\_square

### Floating Point Square

- ❖ The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- ❖ Hardware for denormal numbers of IEEE 754 standard is selectively provided.
- ❖ Fully compatible with the IEEE 754 Floating-point standard
- ❖ DesignWare datapath generator is employed for better timing and area
- ❖ Optional C++ simulation model available in the DWFC Library



**Table 1-1 Pin Description**

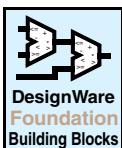
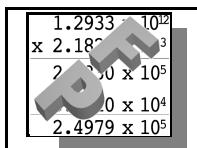
Pin Name	Width	Direction	Function
a	exp_width + sig_width + 1 bits	Input	Input data
rnd	3 bits	Input	Rounding mode
z	exp_width + sig_width + 1 bits	Output	Square of a ( $a^2$ )
status	8 bits	Output	Status flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers a and z
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers a and z
ieee_compliance	0 or 1 Default: 0	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

**Table 1-3 Synthesis Implementations**

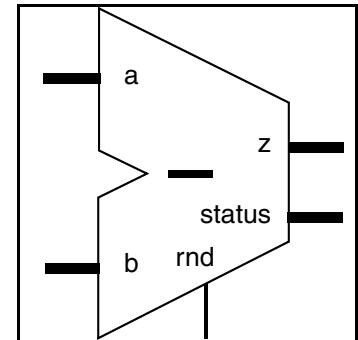
Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



## DW\_fp\_sub

### Floating Point Adder

- ❖ The precision is controlled by parameters, and covers formats in the IEEE standard
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 253 bits
- ❖ Fully compatible with the IEEE 754 Floating-point standard
- ❖ DesignWare datapath generator is employed for better timing and area
- ❖ Optional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

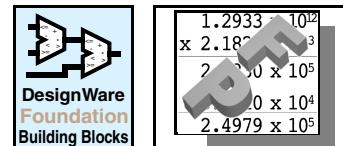
Pin Name	Width	Direction	Function
a	sig_width+exp_width+1 bits	Input	Input data
b	sig_width+exp_width+1 bits	Input	Input data
z	sig_width+exp_width+1 bits	Output	a - b
status	8 bits	Output	Status flags
rnd	3 bits	Input	Rounding mode

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers a, b, and z
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers a, b, and z
ieee_compliance	0 or 1 Default 0	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

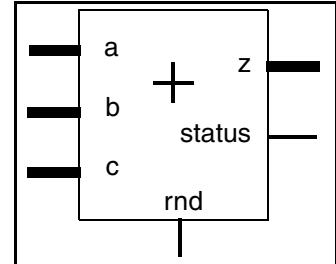
**DW\_fp\_sum3**

3-Input Floating Point Adder

**DW\_fp\_sum3**

3-Input Floating Point Adder

- ❖ The precision is controlled by parameters, and covers formats in the IEEE standard
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of the floating-point number can range from 2 to 253 bits
- ❖ A parameter controls the use of denormal values
- ❖ Faster than an equivalent logic using two FP adders
- ❖ Result is more accurate than using two FP adders
- ❖ DesignWare datapath generator is employed for reduced delay and area

**Table 1-1 Pin Description**

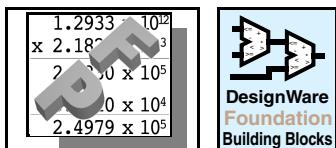
Pin Name	Width	Direction	Function
a	<i>sig_width+exp_width+1</i> bits	Input	Input data
b	<i>sig_width+exp_width+1</i> bits	Input	Input data
c	<i>sig_width+exp_width+1</i> bits	Input	Input data
z	<i>sig_width+exp_width+1</i> bits	Output	$(a + b) + c$
status	8 bits	Output	Status flags
rnd	3 bits	Input	Rounding mode

**Table 1-2 Parameter Description**

Parameter	Values	Description
sig_width	2 to 253 bits	Word length of fraction field of floating-point numbers a, b, c, and z
exp_width	3 to 31 bits	Word length of biased exponent of floating-point numbers a, b, c, and z
ieee_compliance	0 or 1	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs
arch_type	0 or 1	Controls the use of an alternative architecture. Default value is 0 (previous architecture). .

**Table 1-3 Synthesis Implementations**

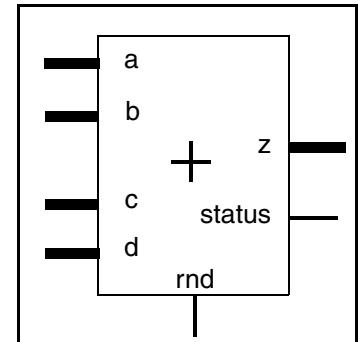
Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



## DW\_fp\_sum4

### 4-Input Floating Point Adder

- ❖ The precision is controlled by parameters, and covers formats in the IEEE standard
- ❖ Exponents can range from 3 to 31 bits
- ❖ Fractional part of floating-point number ranges from 2 to 253 bits
- ❖ A parameter controls the use of denormal values
- ❖ Faster than an equivalent logic using three FP adders for some parameter values
- ❖ Result is always more accurate than using three FP adders



**Table 1-1 Pin Description**

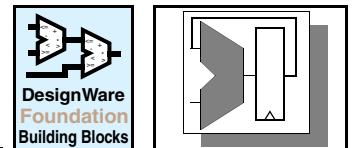
Pin Name	Width	Direction	Function
a	<code>sig_width+exp_width+1</code> bits	Input	Input data
b	<code>sig_width+exp_width+1</code> bits	Input	Input data
c	<code>sig_width+exp_width+1</code> bits	Input	Input data
d	<code>sig_width+exp_width+1</code> bits	Input	Input data
z	<code>sig_width+exp_width+1</code> bits	Output	$((a + b) + c) + d$
status	8 bits	Output	Status flags
rnd	3 bits	Input	Rounding mode

**Table 1-2 Parameter Description**

Parameter	Values	Description
<code>sig_width</code>	2 to 253 bits	Word length of fraction field of floating-point numbers a, b, c, d, and z
<code>exp_width</code>	3 to 31 bits	Word length of biased exponent of floating-point numbers a, b, c, d, and z
<code>ieee_compliance</code>	0 or 1	When 1, the generated architecture is fully compliant with IEEE 754 standard, including the use of denormals and NaNs.
<code>arch_type</code>	0 or 1	Controls the use of an alternative architecture. Default value is 0 (previous architecture).

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



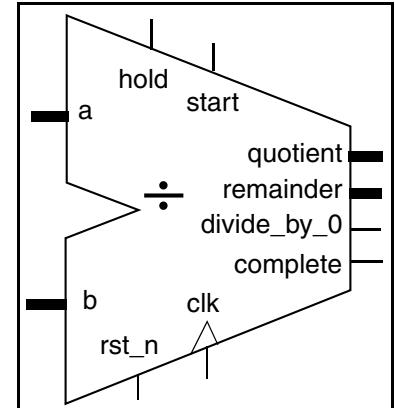
## Datapath – Sequential Overview

This section documents the various Datapath - Sequential IP found in the DesignWare Building Block IP.

## DW\_div\_seq

### Sequential Divider

- ❖ Parameterized word length
- ❖ Parameterized number of clock cycles
- ❖ Unsigned and signed (two's complement) data division
- ❖ Registered or un-registered inputs and outputs
- ❖ C++ simulation model available in the DWFC Library
- ❖ Provides minPower benefits with the DesignWare-LP license.

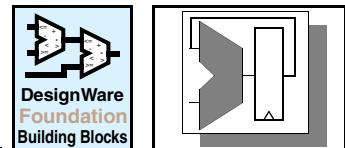


**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, active low
hold	1 bit	Input	Hold current operation (=1)
start	1 bit	Input	Start operation (=1). A new operation is started by setting start=1 for one clock cycle.
a	<i>a_width</i> bit(s)	Input	Dividend
b	<i>b_width</i> bit(s)	Input	Divisor
complete	1 bit	Output	Operation completed (=1)
divide_by_0	1 bit	Output	Indicates if b equals 0
quotient	<i>a_width</i> bit(s)	Output	Quotient
remainder	<i>b_width</i> bit(s)	Output	Remainder

**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\geq 3$ Default 8	Word length of a
b_width	3 to <i>a_width</i> Default: 8	Word length of b
tc_mode	0 or 1 Default: 0	Two's complement control 0 = unsigned 1 = two's complement
num_cyc	3 to <i>a_width</i> Default: 3	User-defined number of clock cycles to produce a valid result. The real number of clock cycles depends on various parameters.

**DW\_div\_seq**

Sequential Divider

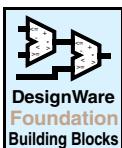
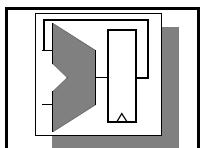
**Table 1-2 Parameter Description (Continued)**

Parameter	Values	Description
rst_mode	0 or 1 Default: 0	Reset mode 0 = asynchronous reset 1 = synchronous reset
input_mode <sup>a</sup>	0 or 1 Default: 1	Registered inputs 0 = no 1 = yes
output_mode	0 or 1 Default: 1	Registered outputs 0 = no 1 = yes
early_start	0 or 1 Default: 0	Computation start 0 = start computation in the second cycle 1 = start computation in the first cycle

- a. When configured with the parameter *input\_mode* set to '0', the inputs *a* and *b* MUST be held constant from the time *start* is asserted until *complete* has gone high to signal completion of the calculation. Conversely, if a configuration with the parameter *input\_mode* set to '1' is used, the *a* and *b* inputs will be captured when *start* is high and otherwise ignored.

**Table 1-3 Synthesis Implementations**

Implementation	Function	License Feature Required
cpa	radix-2 synthesis model	DesignWare
cpa2	radix-4 synthesis model	DesignWare



## DW\_mult\_seq

### Sequential Multiplier

- ❖ Parameterized word length
- ❖ Parameterized number of clock cycles
- ❖ Unsigned and signed (two's complement) data multiplication
- ❖ Registered or un-registered inputs and outputs.
- ❖ Provides minPower benefits with the DesignWare-LP license.

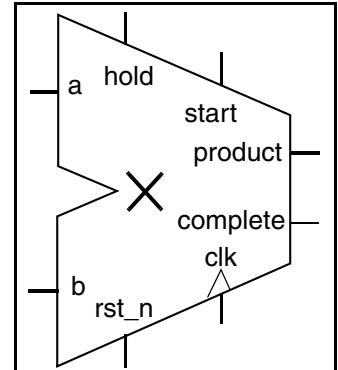
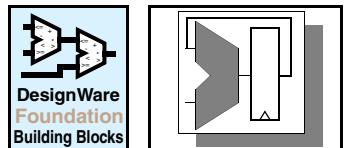


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, active low
hold	1 bit	Input	Hold current operation (=1)
start	1 bit	Input	Start operation (=1). A new operation is started again by making start=1 for one clock cycle.
a	<i>a_width</i> bit(s)	Input	Multiplier
b	<i>b_width</i> bit(s)	Input	Multiplicand
complete	1 bit	Output	Operation completed (=1)
product	<i>a_width + b_width</i> bit(s)	Output	Product $a \times b$

Table 1-2 Parameter Description

Parameter	Values	Description
a_width	$\geq 3$ and $\leq b\_width$	Word length of a
b_width	$\geq 3$	Word length of b
tc_mode	0 or 1 Default: 0	Two's complement control 0 = unsigned 1 = two's complement
num_cyc	$\geq 3$ and $\leq a\_width$ Default: 3	User-defined number of clock cycles to produce a valid result. The real number of clock cycles depends on various parameters.
rst_mode	0 or 1 Default: 0	Reset mode 0 = asynchronous reset 1 = synchronous reset
input_mode <sup>a</sup>	0 or 1 Default: 1	Registered inputs 0 = no 1 = yes



## DW\_mult\_seq

### Sequential Multiplier

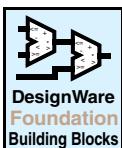
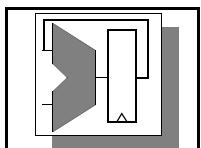
**Table 1-2 Parameter Description (Continued)**

Parameter	Values	Description
output_mode	0 or 1 Default: 1	Registered outputs 0 = no 1 = yes
early_start	0 or 1 Default: 0	Computation start 0 = start computation in the second cycle 1 = start computation in the first cycle

- a. When configured with the parameter *input\_mode* set to '0', the inputs *a* and *b* MUST be held constant from the time *start* is asserted until *complete* has gone high to signal completion of the calculation. Conversely, if a configuration with the parameter *input\_mode* set to '1' is used, the *a* and *b* inputs will be captured when *start* is high and otherwise ignored.

**Table 1-3 Synthesis Implementations**

Implementation	Function	License Feature Required
cpa	Carry-propagate adder synthesis model	DesignWare



## DW\_sqrt\_seq

### Sequential Square Root

- ❖ Parameterized word length
- ❖ Parameterized number of clock cycles
- ❖ Unsigned and signed (two's complement) square roots
- ❖ Registered or un-registered inputs and outputs
- ❖ Provides minPower benefits with the DesignWare-LP license.

Note that data input is taken as absolute value. Two's complement input is converted into unsigned magnitude. Output is unsigned (positive).

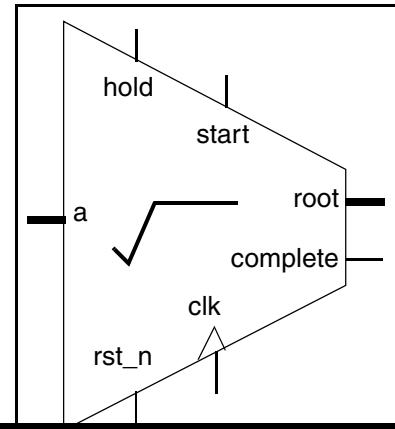
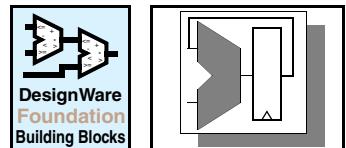


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, active low
hold	1 bit	Input	Hold current operation (=1)
start	1 bit	Input	Start operation (=1). A new operation is started by setting start=1 for one clock cycle.
a	<i>width</i> bit(s)	Input	Radicand
complete	1 bit	Output	Operation completed (=1)
root	( <i>width</i> +1)/2 bit(s)	Output	Square root

Table 1-2 Parameter Description

Parameter	Values	Description
width	$\geq 6$	Word length of a
tc_mode	0 or 1 Default: 0	Two's complement control 0 = unsigned 1 = two's complement
num_cyc <sup>a</sup>	3 to int((width+1)/2) Default: 3	User-defined number of clock cycles to produce a valid result. The real number of clock cycles depends on various parameters.
rst_mode	0 or 1 Default: 0	Reset mode 0 = asynchronous reset 1 = synchronous reset
input_mode <sup>b</sup>	0 or 1 Default: 1	Registered inputs 0 = no 1 = yes



## DW\_sqrt\_seq

### Sequential Square Root

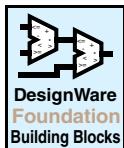
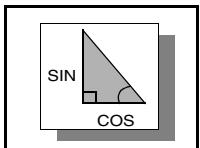
Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
output_mode	0 or 1 Default: 1	Registered outputs 0 = no 1 = yes
early_start	0 or 1 Default: 0	Computation start 0 = start computation in the second cycle 1 = start computation in the first cycle

- a. Note that the num\_cyc specification indicates the actual throughput of the device. That is, if a new input is driven before the num\_cyc number of cycles are complete, the results are undetermined.
- b. When configured with the parameter *input\_mode* set to '0', input 'a' MUST be held constant from the time *start* is asserted until *complete* has gone high to signal completion of the calculation. Conversely, if a configuration with the parameter *input\_mode* set to '1' is used, the 'a' input will be captured when *start* is high and otherwise ignored.

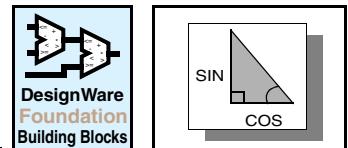
Table 1-3 Synthesis Implementations

Implementation	Function	License Feature Required
cpa	Carry-propagate adder synthesis model	DesignWare



## Datapath – Trigonometric Overview

The trigonometric IP, many of which are inferred, are applicable to ASIC or SoC designs. These IP are high performance trigonometric implementations (based on a fast carry look-ahead architecture).

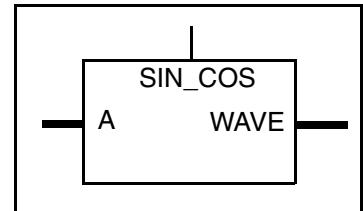
**DW\_sincos**

Combinational Sine - Cosine

**DW\_sincos**

Combinational Sine - Cosine

- ❖ Parameterized word length
- ❖ sine or cosine output by controlling SIN\_COS port.
- ❖ DesignWare Datapath generator is employed for better timing and area.

**Table 1-1 Pin Description**

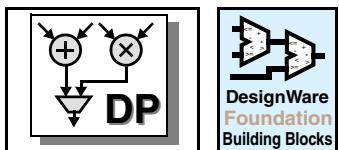
Pin Name	Width	Direction	Function
A	<i>A_width</i> bits	Input	Input data (radian)
SIN_COS	1 bit	Input	Function select: 0 = sine function 1 = cosine function
WAVE	<i>WAVE_width</i> bits	Output	Sine or cosine value

**Table 1-2 Parameter Description**

Parameter	Values	Description
A_width	2 to 34 bits	Word length of A
WAVE_width	2 to 35 bits	Word length of WAVE
arch	0 or 1 Default: 0	Select the implementation 0: area-optimized implementation 1: speed-optimized implementation
err_range	1 or 2 Default: 1	Select the error range 1 :  true value - calculated  < 1 ulp 2 :  true value - calculated  < 2 ulp

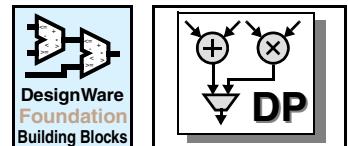
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



# Datapath Functions Overview

This section contains Datapath Functions. These functions improve the quality of synthesized datapaths.



## DWF\_dp\_absval

Absolute Value Function

# DWF\_dp\_absval

## Absolute Value Function

The DWF\_dp\_absval function returns the absolute value (magnitude) of argument *a*. Argument *a* and the return value are signed (two's complement).

The signed return value can be converted to an unsigned to prevent overflow. Overflow occurs for the value  $\text{DWF\_dp\_absval}(-2^{\text{width}-1}) = 2^{\text{width}-1}$ , which cannot be represented as a signed number of *width* bits.

**Table 1-1 Function Names**

Function Name	Description
DWF_dp_absval <sup>a</sup>	VHDL/Verilog signed (two's complement) absolute value

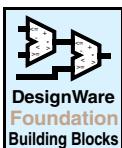
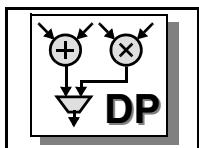
- a. A similar function DWF\_absval exists for DesignWare Building Blocks component DW01\_absval. However, function DWF\_dp\_absval is likely to give better QoR, especially if used inside a larger datapath context.

**Table 1-2 Argument Description**

Argument Name	Type	Width	Description
<i>a</i>	Vector	width	Input data
DWF_dp_absval	Vector	width	Return value

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
width	$\geq 1$	Word length of input <i>a</i> and return value



## DWF\_dp\_blend

### Graphics Alpha Blend Functions

The DWF\_dp\_blend function blend two input pixels (arguments *x* and *y*) with a factor (argument *alpha*) and return the output pixel..

**Table 1-1 DWF\_dp\_blend Function Names**

Function Name	Description
DWF_dp_blend	VHDL/Verilog basic blend (best QoR, result has error of 1 LSB)
DWF_dp_blend_exact	VHDL/Verilog exact blend (worst QoR, exact result)
DWF_dp_blend_exact2 <sup>a</sup>	VHDL/Verilog exact2 blend (good QoR, exact result, alpha1 input)
DWF_dp_blend2	VHDL/Verilog basic blend without rounding/truncation
DWF_dp_blend2_exact2 <sup>b</sup>	VHDL/Verilog exact2 blend without rounding/truncation

a. Compatible with the MCL function `gfxBlend` from Module Compiler (MC).

b. Compatible with the MCL function `gfxBlend2` from Module Compiler (MC).

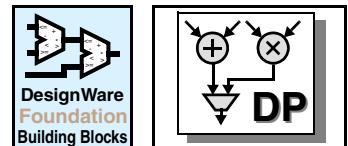
**Table 1-2 Argument Descriptions**

Argument Name	Type	Width / Values	Description
<i>x</i>	Vector	width	Input pixel
<i>y</i>	Vector	width	Input pixel
<i>alpha</i>	Vector	alpha_width	Blend factor
<i>alpha1</i> <sup>a</sup>	Bit	1	Blend factor 1 indicator
DWF_dp_blend DWF_dp_blend_exact DWF_dp_blend_exact2	Vector	width	Return value
DWF_dp_blend2 DWF_dp_blend2_exact2	Vector	width+alpha_width	Return value

a. Only for functions DWF\_dp\_blend\_exact2 and DWF\_dp\_blend2\_exact2.

**Table 1-3 Parameter Descriptions (Verilog)**

Parameter	Values	Description
width	$\geq 2$	Word length of inputs <i>x</i> and <i>y</i>
alpha_width	$\geq 2$	Word length of input <i>alpha</i>

**DWF\_dp\_count\_ones**

Count Ones

## DWF\_dp\_count\_ones

Count Ones

The DWF\_dp\_count\_ones functions count the number of bits with value 1 in argument vector a. Operation for unsigned and signed arguments is identical and the functions always return an unsigned value.

**Table 1-1 Function Names**

Function Name	Description
DWF_dp_count_ones	VHDL unsigned count-ones
DWF_dp_count_ones	VHDL signed (two's complement) count-ones
DWF_dp_count_ones	VHDL std_logic_vector count-ones
DWF_dp_count_ones	Verilog unsigned count-ones
DWF_dp_count_ones	Verilog signed (two's complement) count-ones

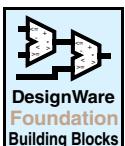
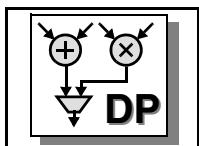
**Table 1-2 Argument Description**

Argument Name	Type	Width / Values	Description
a	Vector	width	Input data
DWF_dp_count_ones	Vector	ceil(log <sub>2</sub> [width+1])	Return value

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
width	≥ 1	Word length of input a

DW\_dp\_count\_ones\_function.inc



## DWF\_dp\_mult\_comb

### Combined Unsigned/Signed Multiply

The DWF\_dp\_mult\_comb function performs combined (switchable) unsigned/signed multiplication of the two arguments a and b. Argument a (b) is interpreted as signed if argument a\_tc (b\_tc) is 1, otherwise as unsigned. The result must be interpreted as signed if argument a\_tc or b\_tc (or both) is 1 (= signed multiplication), otherwise as unsigned (= unsigned multiplication).

**Table 1-1 Function Names**

Function Name	Description
DWF_dp_mult_comb	VHDL combined multiply (std_logic_vector/unsigned/signed arguments)
DWF_dp_mult_comb	Verilog combined multiply
DWF_dp_mult_comb_tc	Verilog combined multiply (signed arguments)

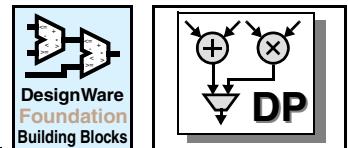
**Table 1-2 Argument Description**

Argument Name	Type	Width / Values	Description
a	Vector	$a\_width$	Input multiplier
a_tc	Bit	1	Two's complement control for multiplier 0 = unsigned, 1 = signed
b	Vector	$b\_width$	Input multiplicand
b_tc	Bit	1	Two's complement control for multiplicand 0 = unsigned, 1 = signed
DWF_dp_mult_comb	Vector	$a\_width+b\_width$	Returned value

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
a_width	$\geq 1$	Word length of input a
b_width	$\geq 1$	Word length of input b

DW\_dp\_mult\_comb\_function.inc

**DWF\_dp\_mult\_comb\_sat**

Combined Unsigned/Signed Multiply and Saturate

**DWF\_dp\_mult\_comb\_sat**

Combined Unsigned/Signed Multiply and Saturate

The DWF\_dp\_mult\_comb\_sat function performs combined (switchable) unsigned/signed multiplication of the two arguments *a* and *b*, truncates the upper bits of the result to the width specified by argument *p\_width* and returns a saturated value if an overflow (or underflow) occurs. Argument *a* (*b*) is interpreted as signed if argument *a\_tc* (*b\_tc*) is 1, otherwise as unsigned. The result must be interpreted as signed if argument **a\_tc** or **b\_tc** (or both) is 1 (= signed multiplication), otherwise as unsigned (= unsigned multiplication). A dedicated overflow detection (needed for saturation) is used to improve QoR of the multiplier.

**Table 1-1 Function Names**

Function Name	Description
DWF_dp_mult_comb_sat	VHDL combined multiply and saturate (std_logic_vector/unsigned/signed arguments)
DWF_dp_mult_comb_sat	Verilog combined multiply and saturate
DWF_dp_mult_comb_sat_tc	Verilog combined multiply and saturate (signed arguments)

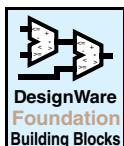
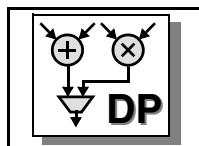
**Table 1-2 Argument Description**

Argument Name	Type	Width / Values	Description
<i>a</i>	Vector	Input	<i>a_width</i>
<i>a_tc</i>	Bit	Input	1
<i>b</i>	Vector	Input	<i>b_width</i>
<i>b_tc</i>	Bit	Input	1
<i>p_width</i>	Integer	Input	$\geq 2$
DWF_dp_mult_comb_sat	Vector	Output	<i>p_width</i>

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
<i>a_width</i>	$\geq 2$	Word length of input <i>a</i>
<i>b_width</i>	$\geq 2$	Word length of input <i>b</i>
<i>p_width</i>	$\geq 2$	Word length of returned value

**Verilog Include File:** DW\_dp\_mult\_comb\_sat\_function.inc



## DWF\_dp\_mult\_ovldet

### Multiply and Overflow Detection

The DWF\_dp\_mult\_ovldet procedure multiplies the two arguments *a* and *b*, truncates the upper bits of the result to the width specified by argument *p\_width* and returns the truncated value and an overflow flag that indicates whether an overflow (or underflow) occurred. A dedicated overflow detection is used to improve the QoR of the multiplier.

**Table 1-1 Function Names**

Function Name	Description
DWF_dp_mult_ovldet	VHDL unsigned multiply and overflow detection
DWF_dp_mult_ovldet	VHDL signed (two's complement) multiply and overflow detection
DWF_dp_mult_ovldet_uns	Verilog unsigned multiply and overflow detection
DWF_dp_mult_ovldet_tc	Verilog signed (two's complement) multiply and overflow detection

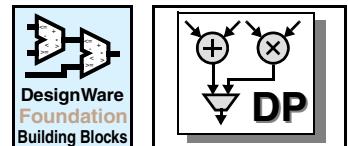
**Table 1-2 Argument Description**

Argument Name	Type	Width / Values	Description
<i>a</i>	Vector	Input	<i>a_width</i>
<i>b</i>	Vector	Input	<i>b_width</i>
<i>p</i>	Vector	Output	<i>p_width</i>
ovfl	Bit	Output	1

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
<i>a_width</i>	$\geq 2$	Word length of input <i>a</i>
<i>b_width</i>	$\geq 2$	Word length of input <i>b</i>
<i>p_width</i>	$\geq 2$	Word length of output product

**Verilog Include File:** DW\_dp\_mult\_ovldet\_function.inc

**DWF\_dp\_mult\_sat**

Multiply and Saturate Functions

## DWF\_dp\_mult\_sat

### Multiply and Saturate Functions

The DWF\_dp\_mult\_sat function multiplies the two arguments *a* and *b*, truncates the upper bits of the result to the width specified by argument *p\_width* and returns a saturated value if an overflow (or underflow) occurs. A dedicated overflow detection is used to improve QoR of the multiplier.

**Table 1-1 Function Names**

Function Name	Description
DWF_dp_mult_sat	VHDL unsigned multiply and saturate
DWF_dp_mult_sat	VHDL signed (two's complement) multiply and saturate
DWF_dp_mult_sat_uns	Verilog unsigned multiply and saturate
DWF_dp_mult_sat_tc	Verilog signed (two's complement) multiply and saturate

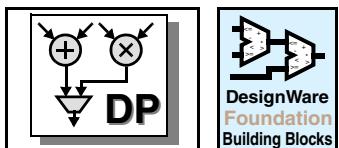
**Table 1-2 Argument Descriptions**

Argument Name	Type	Width / Values	Description
<i>a</i>	Vector	<i>a_width</i>	Input multiplier
<i>b</i>	Vector	<i>b_width</i>	Input multiplicand
<i>p_width</i>	Integer	$\geq 2$	Word length of returned value (VHDL only, constant)
DWF_dp_mult_sat	Vector	<i>p_width</i>	Returned value

**Table 1-3 Parameter Descriptions (Verilog)**

Parameter	Values	Description
<i>a_width</i>	$\geq 2$	Word length of input <i>a</i>
<i>b_width</i>	$\geq 2$	Word length of input <i>b</i>
<i>p_width</i>	$\geq 2$	Word length of returned value

**Verilog Include File:** DW\_dp\_mult\_sat\_function.inc



# DWF\_dp\_rnd

## Arithmetic Rounding Functions

The DWF\_dp\_rnd functions truncate the lower bits of argument  $a$  below the bit position specified by argument  $lsb$  and return a rounded value according to the rounding mode specified by argument  $mode$ . Argument  $a$  and the return value are both either signed (two's complement) or unsigned.

**Table 1-1 Function Names**

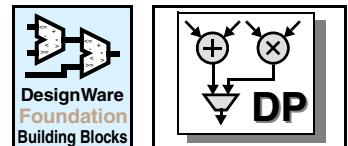
Function Name	Description
DWF_dp_rnd	VHDL unsigned rounding
DWF_dp_rnd	VHDL signed (two's complement) rounding
DWF_dp_rnd_uns	Verilog unsigned rounding
DWF_dp_rnd_tc	Verilog signed (two's complement) rounding

**Table 1-2 Argument Description**

Argument Name	Type	Width / Values	Description
$a$	Vector	width	Input data
$lsb$	Integer	$\geq 1, \leq width-1$	LSB index of return value (VHDL only, constant)
$mode$	Vector	4	Rounding mode
DWF_dp_rnd	Vector	$width-lsb$	Return value

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
width	$\geq 1$	Word length of input $a$
lsb	$\geq 1, \leq width-1$	LSB index of return value

**DWF\_dp\_rndsat**

Arithmetic Rounding and Saturation Functions

# **DWF\_dp\_rndsat**

Arithmetic Rounding and Saturation Functions

The DWF\_dp\_rndsat functions truncate the lower bits of argument *a* below the bit position specified by argument *lsb*, round according to the rounding mode specified by argument *mode*, truncate the upper bits above the bit position specified by argument *msb* and return a saturated value if an overflow (or underflow) occurs. Argument *a* and the return value are both either signed (two's complement) or unsigned.

**Table 1-1 Function Names**

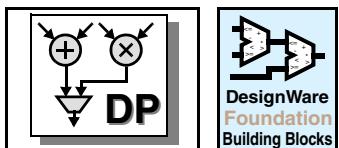
Function Name	Description
DWF_dp_rndsat	VHDL unsigned rounding and saturation
DWF_dp_rndsat	VHDL signed (two's complement) rounding and saturation
DWF_dp_rndsat_uns	Verilog unsigned rounding and saturation
DWF_dp_rndsat_tc	Verilog signed (two's complement) rounding and saturation

**Table 1-2 Argument Description**

Argument Name	Type	Width / Values	Description
<i>a</i>	Vector	width	Input data
<i>msb</i>	Integer	$> \text{lsb}, \leq \text{width}-1$	MSB index of return value (VHDL only, constant)
<i>lsb</i>	Integer	$\geq 1, < \text{msb}$	LSB index of return value (VHDL only, constant)
<i>mode</i>	Vector	4	Rounding mode (values in DWF_dp_rnd: <a href="#">Table 1-4</a> )
DWF_dp_rndsat	Vector	<i>msb-lsb+1</i>	Return value

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
<i>width</i>	$\geq 1$	Word length of input <i>a</i>
<i>msb</i>	$> \text{lsb}, \leq \text{width}-1$	MSB index of return value
<i>lsb</i>	$\geq 1, < \text{msb}$	LSB index of return value



## DWF\_dp\_sat

### Arithmetic Saturation Functions

The DWF\_dp\_sat functions truncate the upper bits of argument *a* to the width specified by argument *size* and returns a saturated value if an overflow (or underflow) occurs. Argument *a* and the return value are both either signed (two's complement) or unsigned.

The DWF\_dp\_sati functions have the same functionality, except that all output bits are inverted.

**Table 1-1 Function Names**

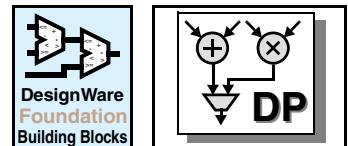
Function Name	Description
DWF_dp_sat	VHDL unsigned saturate
DWF_dp_sati	VHDL signed (two's complement) saturate
DWF_dp_sat_uns	Verilog unsigned saturate
DWF_dp_sat_tc	Verilog signed (two's complement) saturate
DWF_dp_sati	VHDL unsigned saturate, inverted output
DWF_dp_sati	VHDL signed (two's complement) saturate, inverted output
DWF_dp_sati_uns	Verilog unsigned saturate, inverted output
DWF_dp_sati_tc	Verilog signed (two's complement) saturate, inverted output

**Table 1-2 Argument Description**

Argument Name	Type	Width / Values	Description
<i>a</i>	Vector	width	Input data
<i>size</i>	Integer	> 1, < width	Word length of return value (VHDL only, constant)
DWF_dp_sat	Vector	<i>size</i>	Return value
DWF_dp_sati			

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
width	$\geq 1$	Word length of input <i>a</i>
size	$> 1, < \text{width}$	Word length of return value

**DWF\_dp\_sign\_select**

Sign Selection / Conditional Two's Complement Functions

## DWF\_dp\_sign\_select

Sign Selection / Conditional Two's Complement Functions

The DWF\_dp\_sign\_select functions return the positive or negative (two's complement) value of argument *a* controlled by sign argument *s*. Argument *a* and the return value are both either signed (two's complement) or unsigned.

A signed return value has overflow for the value  $\text{DWF\_dp\_sign\_select}(-2^{\text{width}-1}, 1) = 2^{\text{width}-1}$ , which cannot be represented as a signed number of width bits. The complement of an unsigned number always results in underflow (that is, a negative number cannot be represented as an unsigned), but the unsigned DWF\_dp\_sign\_select can be meaningful for the conditional addition/subtraction in a larger unsigned expression.

**Table 1-1 Function Names**

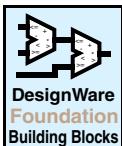
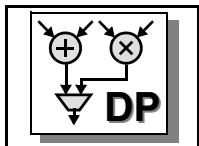
Function Name	Description
DWF_dp_sign_select	VHDL unsigned sign select
DWF_dp_sign_select	VHDL signed (two's complement) sign select
DWF_dp_sign_select_uns	Verilog unsigned sign select
DWF_dp_sign_select_tc	Verilog signed (two's complement) sign select

**Table 1-2 Argument Description**

Argument Name	Type	Width	Description
<i>a</i>	Vector	width	Input data
<i>s</i>	Bit	1	Sign / complement control
DWF_dp_sign_select	Vector	width	Return value

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
width	$\geq 1$	Word length of input <i>a</i> and return value



## DWF\_dp\_simd\_addc

### SIMD Add with Carries

The DWF\_dp\_simd\_addc procedures implement a configurable SIMD adder with input and output carries. They allow you to either add arguments *a* and *b* as full-width vectors (for example, one 32-bit addition) or to add smaller partitions of *a* and *b* using multiple parallel adders (for example, two 16-bit additions or four 8-bit additions), each with separate input carries *cin* and output carries *cout*. The argument *no\_confs* specifies the number of possible configurations, and argument *conf* dynamically selects one configuration. Configuration with number *conf* has  $2^{\text{conf}}$  partitions of size  $\text{width}/2^{\text{conf}}$ . Arguments *a* and *b* and the output *s* are either unsigned or signed (two's complement).

**Table 1-1 Procedure Names**

Function Name	Description
DWF_dp_simd_addc	VHDL unsigned SIMD add with carries
DWF_dp_simd_addc	VHDL signed (two's complement) SIMD add with carries
DWF_dp_simd_addc_uns	Verilog unsigned SIMD add with carries
DWF_dp_simd_addc_tc	Verilog signed (two's complement) SIMD add with carries

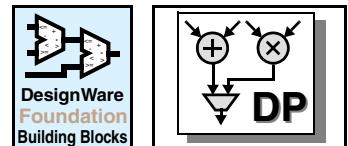
**Table 1-2 Argument Description**

Argument Name	Type	Direction	Width / Values	Description
<i>a</i>	Vector <sup>a</sup>	Input	width	Input addend
<i>b</i>	Vector <sup>a</sup>	Input	width	Input addend
<i>cin</i>	Vector <sup>b</sup>	Input	$2^{\text{no\_confs}-1}$	Input carries
<i>no_confs</i>	Integer	Input	$\geq 2$	Number of configurations (VHDL only, constant)
<i>conf</i>	Vector <sup>b</sup>	Input	$\text{ceil}(\log_2[\text{no\_confs}])$	Configuration selection: $2^{\text{conf}}$ partitions of size $\text{width}/2^{\text{conf}}$
<i>s</i>	Vector <sup>a</sup>	Output	width	Output sum
<i>cout</i>	Vector <sup>b</sup>	Output	$2^{\text{no\_confs}-1}$	Output carries

- a. unsigned or signed in VHDL
- b. std\_logic\_vector in VHDL

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
<i>width</i>	$\geq 2$ , must be a multiple of $2^{\text{no\_confs}-1}$	Word length
<i>no_confs</i>	$\geq 2$	Number of configurations

**DWF\_dp\_simd\_add**

SIMD Add

## DWF\_dp\_simd\_add

SIMD Add

The DWF\_dp\_simd\_add functions implement a configurable SIMD adder. They allow you to either add arguments *a* and *b* as full-width vectors (for example, one 32-bit addition) or to add smaller partitions of *a* and *b* using multiple parallel adders (for example, two 16-bit additions or four 8-bit additions). Argument *no\_confs* specifies the number of possible configurations and argument *conf* dynamically selects one configuration. Configuration with number *conf* has  $2^{\text{conf}}$  partitions of size  $\text{width}/2^{\text{conf}}$ . Arguments *a*, *b*, and the return value are all either unsigned or signed (two's complement).

**Table 1-1 Function Names**

Function Name	Description
DWF_dp_simd_add	VHDL unsigned SIMD add
DWF_dp_simd_add	VHDL signed (two's complement) SIMD add
DWF_dp_simd_add_uns	Verilog unsigned SIMD add
DWF_dp_simd_add_tc	Verilog signed (two's complement) SIMD add

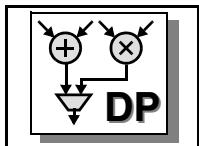
**Table 1-2 Argument Description**

Argument Name	Type	Width / Values	Description
<i>a</i>	Vector <sup>a</sup>	width	Input addend
<i>b</i>	Vector <sup>a</sup>	width	Input addend
<i>no_confs</i>	Integer	$\geq 2$	Number of configurations (VHDL only, constant)
<i>conf</i>	Vector <sup>b</sup>	$\text{ceil}(\log_2[\text{no\_confs}])$	Configuration selection: $2^{\text{conf}}$ partitions of size $\text{width}/2^{\text{conf}}$
DWF_dp_simd_add	Vector <sup>a</sup>	width	Returned value

- a. unsigned or signed in VHDL
- b. std\_logic\_vector in VHDL

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
width	$\geq 2$ , must be a multiple of $2^{\text{no\_confs}-1}$	Word length
no_confs	$\geq 2$	Number of configurations



## DWF\_dp\_simd\_mult

### SIMD Multiply

The DWF\_dp\_simd\_mult functions implement a configurable SIMD multiplier. They allow you to either multiply arguments *a* and *b* as full-width vectors (for example, one 32-bit multiplication) or to multiply smaller partitions of *a* and *b* using multiple parallel multipliers (for example, two 16-bit multiplications or four 8-bit multiplications). The argument *no\_confs* specifies the number of possible configurations, and argument *conf* dynamically selects one of the configurations. Configuration with number *conf* has  $2^{\text{conf}}$  partitions of size  $\text{width}/2^{\text{conf}}$ . Arguments *a* and *b* and the return value are all either unsigned or signed (two's complement).

**Table 1-1 Function Names**

Function Name	Description
DWF_dp_simd_mult	VHDL unsigned SIMD multiply
DWF_dp_simd_mult	VHDL signed (two's complement) SIMD multiply
DWF_dp_simd_mult_uns	Verilog unsigned SIMD multiply
DWF_dp_simd_mult_tc	Verilog signed (two's complement) SIMD multiply

**Table 1-2 Argument Description**

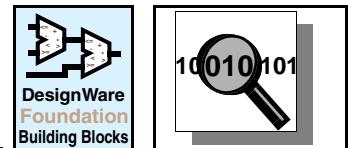
Argument Name	Type	Width / Values	Description
<i>a</i>	Vector <sup>a</sup>	width	Input multiplier
<i>b</i>	Vector <sup>a</sup>	width	Input multiplicand
<i>no_confs</i>	Integer	$\geq 2$	Number of configurations (VHDL only, constant)
<i>conf</i>	Vector <sup>b</sup>	$\text{ceil}(\log_2[\text{no\_confs}])$	Configuration selection: $2^{\text{conf}}$ partitions of size $\text{width}/2^{\text{conf}}$
DWF_dp_simd_mult	Vector <sup>a</sup>	width	Return value

a. unsigned or signed in VHDL

b. std\_logic\_vector in VHDL

**Table 1-3 Parameter Description (Verilog)**

Parameter	Values	Description
width	$\geq 2$ , must be a multiple of $2^{\text{no\_confs}-1}$	Word length
no_confs	$\geq 2$	Number of configurations



## Data Integrity

This section documents the various DesignWare Building Block IP data integrity components.



## DW\_crc\_p

Universal Parallel (Combinational) CRC Generator/Checker

- ❖ Parameterized arbitrary polynomial (up to 64-bit)
- ❖ Parameterized data width (up to 512 bits)
- ❖ Parameterized initial CRC value (all ones or all zeroes)
- ❖ Parameterized inversion of generated CRC
- ❖ Parameterized bit and byte ordering

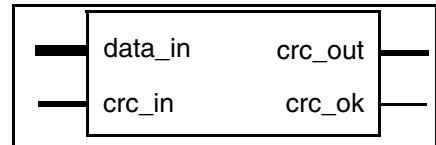


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
data_in	<i>data_width</i> bit(s)	Input	Input data used for both generating and checking for valid CRC
crc_in	<i>poly_size</i> bit(s)	Input	Input CRC value used to check a record (not used when generating CRC from data_in)
crc_ok	1 bit	Output	Indicates a correct residual CRC value, active high
crc_out	<i>poly_size</i> bit(s)	Output	Provides the CRC check bits to be appended to the input data to form a valid record (data_in and crc_in)

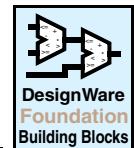
Table 1-2 Parameter Description

Parameter	Values	Description
data_width	1 to 512 Default: 16	Width of data_in (i.e. the amount of data that CRC will be calculated upon)
poly_size	2 to 64 Default: 16	Size of the CRC polynomial and thus the width of crc_in and crc_out
crc_cfg	0 to 7 Default: 7	CRC initialization and insertion configuration
bit_order	0 to 3 Default: 3	Bit and byte order configuration
poly_coef0	1 to 65535 <sup>a</sup> Default: 4129 <sup>b</sup>	Polynomial coefficients 0 through 15
poly_coef1	0 to 65535 Default: 0	Polynomial coefficients 16 through 31
poly_coef2	0 to 65535 Default: 0	Polynomial coefficients 32 through 47
poly_coef3	0 to 65535 Default: 0	Polynomial coefficients 48 through 63

a. *poly\_coef0* must be an odd number (since all primitive polynomials include the coefficient 1, which is equivalent to  $X^0$ ).

b. CCITT-CRC16 polynomial is  $X^{16} + X^{12} + X^5 + 1$ , thus

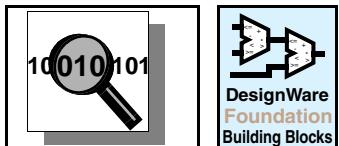
$$poly\_coef0 = 212 + 25 + 1 = 4129.$$

**DW\_crc\_p**

Universal Parallel (Combinational) CRC Generator/Checker

**Table 1-3    Synthesis Implementations**

Implementation Name	Implementation	License Feature Required
str	Synthesis model	DesignWare



## DW\_crc\_s

Universal Synchronous (Clocked) CRC Generator/Checker

- ❖ Parameterized arbitrary polynomial (up to 64-bit)
- ❖ Parameterized data width (up to polynomial size)
- ❖ Parameterized register initialization (all ones or all zeroes)
- ❖ Parameterized inverted insertion of generated CRC
- ❖ Parameterized bit and byte ordering
- ❖ Loadable CRC value for use in context switching of interspersed blocks

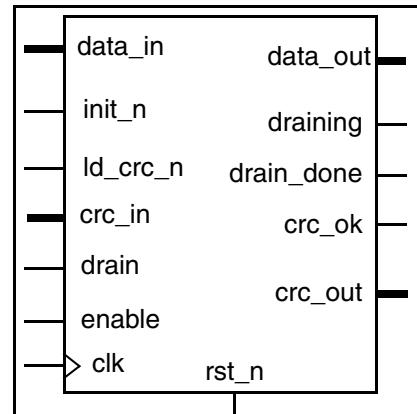
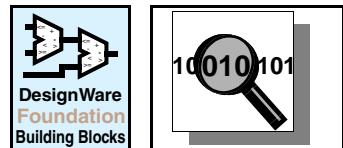


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock input
rst_n	1 bit	Input	Asynchronous reset input, active low
init_n	1 bit	Input	Synchronous initialization control input, active low
enable	1 bit	Input	Enable control input for all operations (other than reset and initialization), active high
drain	1 bit	Input	Drains control input, active high
ld_crc_n	1 bit	Input	Synchronous CRC register load control input, active low
data_in	<i>data_width</i> bit(s)	Input	Input data
crc_in	<i>poly_size</i> bit(s)	Input	Input CRC value (to be loaded into the CRC register as commanded by the <i>ld_crc_n</i> control input)
draining	1 bit	Output	Indicates that the CRC register is draining (inserting the CRC into the data stream)
drain_done	1 bit	Output	Indicates that the CRC register has finished draining
crc_ok	1 bit	Output	Indicates a correct residual CRC value, active high
data_out	<i>data_width</i> bit(s)	Output	Output data
crc_out	<i>poly_size</i> bit(s)	Output	Provides constant monitoring of the CRC register

**DW\_crc\_s**

Universal Synchronous (Clocked) CRC Generator/Checker

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to <i>poly_size</i> Default: 16	Width of data_in and data_out (also the number of bits per clock) The <i>poly_size</i> value must be an integer multiple of <i>data_width</i> .
poly_size	2 to 64 Default: 16	Size of the CRC polynomial The <i>poly_size</i> value must be an integer multiple of <i>data_width</i> .
crc_cfg	0 to 7 Default: 7	CRC initialization and insertion configuration
bit_order	0 to 3 Default: 3	Bit and byte order configuration
poly_coef0	1 to 65535 <sup>a</sup> Default: 4129 <sup>b</sup>	Polynomial coefficients 0 through 15
poly_coef1	0 to 65535 Default: 0	Polynomial coefficients 16 through 31
poly_coef2	0 to 65535 Default: 0	Polynomial coefficients 32 through 47
poly_coef3	0 to 65535 Default: 0	Polynomial coefficients 48 through 63

- a. The *poly\_coef0* value must be an odd number (since all primitive polynomials include the coefficient 1, which is equivalent to  $X^0$ ).
- b. CCITT-CRC16 polynomial is  $X^{16} + X^{12} + X^5 + 1$ , thus  $\text{poly\_coef0} = 2^{12} + 2^5 + 1 = 4129$ .

**Table 1-3 Synthesis Implementations**

Implementation Name	Implementation	License Feature Required
str	Synthesis model	DesignWare



## DW\_ecc

### Error Checking and Correction

- ❖ Parameterized word width
- ❖ Generates check bits for new data written, and corrects corrupt data for read and read-modify-write cycles
- ❖ Supports scrubbing
- ❖ Flags to indicate if an error was detected, and if the error is not correctable
- ❖ Flow-through architecture for speed and flexibility
- ❖ Error syndrome output for error logging

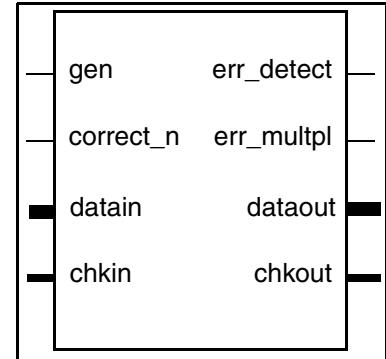
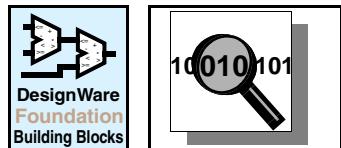


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
gen	1 bit	Input	Suppresses correction in write mode ( <code>gen = 1</code> ) and generates check bits. Enables correction when in read mode ( <code>gen = 0</code> ) and <code>correct_n</code> is asserted (low).
correct_n	1 bit	Input	Enables correction of correctable words, active low
datain	<i>width</i> bits	Input	Input data word to check (check mode), or data from which check bits are generated (generate mode)
chkin	<i>chkbits</i> bits	Input	Check bits input for error analysis on read
err_detect	1 bit	Output	Indicates that an error has been detected, active high. Location of error is specified by the error syndrome.
err_multpl	1 bit	Output	Indicates that the error detected is a multiple-bit error and, therefore, uncorrectable
dataout	<i>width</i> bits	Output	Output data. May be corrected if an error is detected and <code>correct_n</code> is asserted.
chkout	<i>chkbits</i> bits	Output	When <code>gen = 1</code> , <code>chkout</code> contains the check bits generated from <code>datain</code> . When <code>gen = 0</code> and <code>synd_sel = 0</code> , <code>chkout</code> is the corrected or uncorrected data from <code>chkin</code> . When <code>gen = 0</code> and <code>synd_sel = 1</code> , <code>chkout</code> is the error syndrome value

Table 1-2 Parameter Description

Parameter	Values	Description
width	8 to 8178	Width of input and output data buses
chkbits	5 to 14	Width of check bits input and output buses, calculated from <code>width</code>
synd_sel	0 or 1	Selects function of <code>chkout</code> when <code>gen = 0</code> . If <code>synd_sel = 0</code> and <code>gen = 0</code> , then <code>chkout</code> is the corrected or uncorrected data from <code>chkin</code> . If <code>synd_sel = 1</code> and <code>gen = 0</code> , then <code>chkout</code> is the error syndrome value

**DW\_ecc**

## Error Checking and Correction

**Table 1-3    Synthesis Implementations**

Implementation Name	Function	License Feature Required
str <sup>a</sup>	Synthesis model	DesignWare
rtl <sup>b</sup>	Synthesis model	DesignWare

- a. For Design Compiler versions prior to 2009.06-SP1, the "str" implementation is used when *chkbits* is set to a value of '10' or less; "str" is not available for 2009.06-SP1 or later versions.
- b. As of Design Compiler version 2009.06-SP1, "rtl" is the only synthesis implementation available and it supports all word sizes. Prior to 2009.06-SP1, the "rtl" implementation is used when *chkbits* is set to a value of '11' or greater.



## DW04\_par\_gen

Parity Generator and Checker

- ❖ Generates parity for given input data
- ❖ Supports even and odd parity, selectable via a parameter
- ❖ Supports variable word widths
- ❖ Inferable using a function call



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
datain	<i>width</i> bit(s)	Input	Input data word to check or generate parity
parity	1 bit	Output	Generated parity

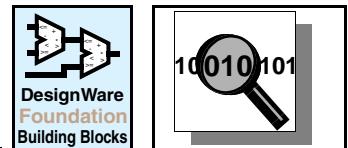
**Table 1-2 Parameter Description**

Parameter	Values <sup>a</sup>	Description
width	1 to 256	Defines the width of the input bus
par_type	0 or 1	Defines the type of parity

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



## Data Integrity – Coding Group Overview

The Coding Group consists of a set of IP that encode and/or decode data for use in data communications and data storage applications. Currently the 8B/10B coding scheme (used in standard data communication and networking protocols such as Gigabit Ethernet and Fiber Channel) is embodied in the Coding Group IP.



## DW\_8b10b\_dec

### 8b10b Decoder

- ❖ Configurable data width
- ❖ Configurable simplified Special Character indicator flags (for protocols requiring only the K28.5 special character)
- ❖ Synchronous initialization of Running Disparity with design specified value
- ❖ All outputs registered
- ❖ Provides minPower benefits with the DesignWare-LP license.

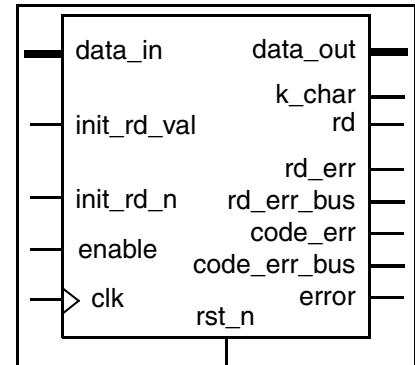
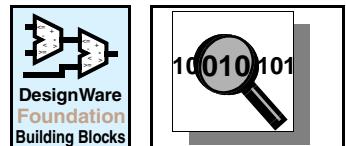


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock input
rst_n	1 bit	Input	Asynchronous reset input, active low
init_rd_n	1 bit	Input	Synchronous initialization control input, active low
init_rd_val	1 bit	Input	Value of initial Running Disparity
data_in	bytes × 10 bit(s)	Input	Input 8b/10b data for decoding
error	1 bit	Output	Active high, error flag indicating the presence of any type of error (running disparity or coding) in the information currently decoded on data_out
rd	1 bit	Output	Current Running Disparity (after decoding data presented at data_in to data_out)
k_char	bytes bit(s)	Output	Special Character indicators (one indicator per decoded byte)
data_out	bytes × 8 bit(s)	Output	Decoded output data
rd_err	1 bit	Output	Active high, error flag indicating the presence of one or more Running Disparity errors in the information currently decoded on data_out
code_err	1 bit	Output	Active high, error flag indicating the presence of a coding error in at least one byte of information currently decoded on data_out
enable	1 bit	Input	Enables register clocking
rd_err_bus	bytes bit(s)	Output	This bus indicates which bytes of the incoming bus have Running Disparity errors (one bit per incoming data_in byte with bit 0 corresponding to an error seen in the least significant 10 bits of input data)
code_err_bus	bytes bit(s)	Output	This bus indicates which bytes of the incoming bus have Coding errors (one bit per incoming data_in byte with bit 0 corresponding to an error seen in the least significant 10 bits of input data).


**DW\_8b10b\_dec**  
 8b10b Decoder
**Table 1-2 Parameter Description**

Parameter	Values	Description
bytes	1 to 16 Default: 2	Number of bytes to encode
k28_5_only	0 or 1 Default: 0	Special Character subset control parameter 0 - for all special characters decoded, 1 - for only K28.5 decoded [when k_char = HIGH implies K28.5, all other special characters indicate an error]
en_mode	0 or 1 Default: 0	Enable control 0 - the enable input port is not connected (backward compatible with older components) 1 - when enable=0 the decoder is stalled
init_mode	0 or 1 Default: 0	Initialization mode for running disparity 0 - during active init_rd_n input, delay init_rd_val one clock cycle before applying it to data_in input in calculating data_out (backward compatible with older components) 1 - during active init_rd_n input, directly apply init_rd_val to data_in input (with no clock cycle delay) in calculating data_out
rst_mode	0 or 1 Default: 0	Reset mode: 0 = asynchronous reset 1 = synchronous reset

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

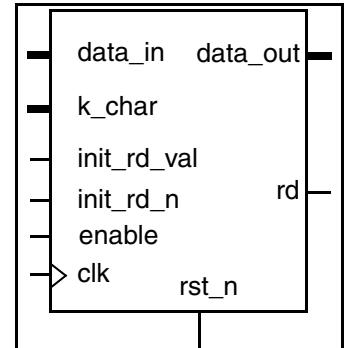


## DW\_8b10b\_enc

### 8b10b Encoder

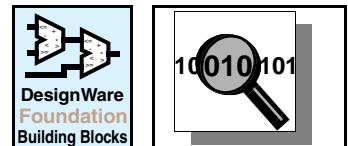
- ❖ Configurable data width
- ❖ Configurable simplified Special Character control (for protocols requiring only the K28.5 special character)
- ❖ Synchronous initialization of Running Disparity with design specified value
- ❖ All outputs registered

Provides minPower benefits with the DesignWare-LP license.



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Asynchronous reset, active low
init_rd_n	1 bit	Input	Synchronous initialization, active low
init_rd_val	1 bit	Input	Value of initial Running Disparity
k_char	bytes bit(s)	Input	Special character controls (one control per byte to encode)
data_in	bytes × 8 bit(s)	Input	Input data for encoding
rd	1 bit	Output	Current Running Disparity (before encoding data presented at data_in)
data_out	bytes × 10 bit(s)	Output	8b10b encoded data
enable	1 bit	Input	Enables register clocking



## DW\_8b10b\_enc

### 8b10b Encoder

**Table 1-2 Parameter Description**

Parameter	Value	Description
bytes	1 to 16 Default: 2	Number of bytes to encode
k28_5_only	0 or 1 Default: 0	Special character subset control parameter 0 for all special characters available, 1 for only K28.5 available [when k_char = HIGH, regardless of the value on data_in]
en_mode	0 or 1 Default: 0	Enable control 0 - the enable input port is not connected (backward compatible with older components) 1 - when enable=0 the encoder is stalled
init_mode	0 or 1 Default: 0	Initialization mode for running disparity 0 - during active init_rd_n input, delay init_rd_val one clock cycle before applying it to data_in input in calculating data_out (backward compatible with older components). 1 - during active init_rd_n input, directly apply init_rd_val to data_in input (with no clock cycle delay) in calculating data_out.
rst_mode	0 or 1 Default: 0	Reset mode: 0 = asynchronous reset 1 = synchronous reset

**Table 1-3 Synthesis Implementations**

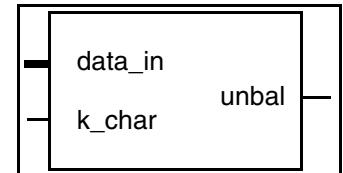
Implementation	Function	License Feature Required
rtl	Synthesis model	DesignWare



## DW\_8b10b\_unbal

### 8b10b Coding Balance Predictor

- ❖ Independent of Running Disparity
- ❖ Higher speed than a full encoder
- ❖ Predicts balance for both data and special characters

**Table 1-1 Pin Description**

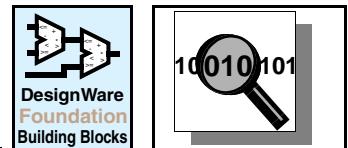
Pin Name	Width	Direction	Function
k_char	1 bit	Input	Special character control input (LOW for data characters, HIGH for special characters)
data_in	8 bits	Input	Input for 8-bit data character to be encoded
unbal	1 bit	Output	Unbalanced code character indicator (LOW for balanced, HIGH for unbalanced)

**Table 1-2 Parameter Description**

Parameter	Values	Description
k28_5_mode	0 or 1 Default: 0	Special Character subset control parameter 0 for all special characters available, 1 for only K28.5 available [when k_char = HIGH, regardless of the value on data_in]

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



# Digital Signal Processing (DSP)

This section documents the DSP components of the DesignWare Building Block IP. Currently, there are two digital FIR filter components designed for applications requiring programmable coefficients for either high-speed or area-efficient filtering.



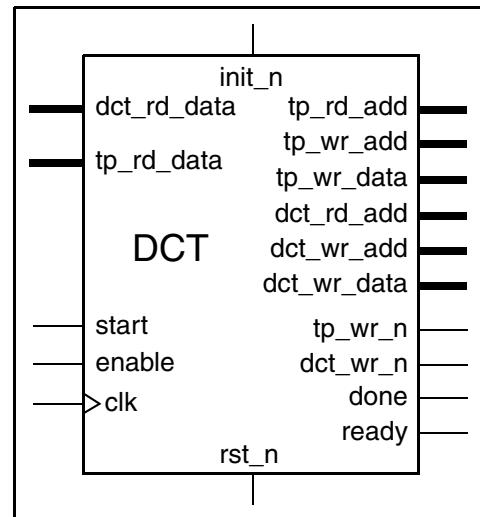
## DW\_dct\_2d

### Two Dimensional Discrete Cosine Transform

- ❖ Parameterized input data width and block size
- ❖ Parameterized coefficients

### Applications

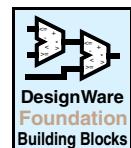
- ❖ Linear DCT for audio such as Dolby Digital
- ❖ NxN matrix for jpeg mjpe/mpeg and DV video
- ❖ Dolby AC2 and AC3: 1-D DCT
- ❖ JPEG (still images): 2-D DCT spatial compression
- ❖ MPEG1 and MPEG2: 2-D DCT plus motion compensation
- ❖ H.261 and H.263: moving image compression for video conferencing and video telephony



DW\_dct\_2d is only valid for A-2007.12-SP1 and later releases.

**Table 1-1 Pin Description**

Pin Name	Direction	Function
clk	input	Primary clock
rst_n	input	Asynchronous reset
init_n	input	Synchronous reset
enable	input	Enable run
start	input	Start transform
dct_rd_data	input	Read data input port
tp_rd_data	input	Read data from transpose RAM
tp_rd_add	output	Read address for transpose RAM
tp_wr_add	output	Write address for transpose RAM
tp_wr_data	output	Data out to transpose RAM
tp_wr_n	output	Write to transpose RAM (active low)
dct_wr_n	output	Write signal out (active low)
done	output	Read data input done
ready	output	First transformed data ready
dct_wr_data	output	Transformed data out

**DW\_dct\_2d**

## Two Dimensional Discrete Cosine Transform

**Table 1-2 Parameter Description**

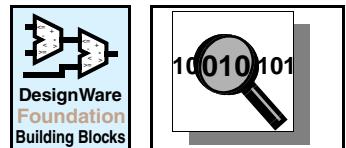
Parameter	Value	Description
n	4 - 16 Default: 8	Size of 2 dimensional matrix  n is the size of the two dimensional matrix to be transformed. It must be even and between 4 and 16, so 4, 6, 8, 10, 12, 14, and 16. The coefficients are generated for each size of n, using the formulas listed above. The scaling factor required for each coefficient must be applied to the coefficient before inserting it in the module.
bpp	4 - 32 Default: 8	Number of bits per pixel
reg_out	0, 1 Default: 0	Register outputs
tc_mode	0, 1 Default: 0	Input data type: two's complement  Determines the data type of the input data, for forward dct mode only. If set to 0, the data is assumed to be non negative, and set to 1, the data is considered two's complement. Intermediate data and output data is always in two's complement mode.
rt_mode	0, 1 Default: 1	0:round data 1:truncate data  Determines the output rounding or truncate. 0 means round the data to the nearest up. 1 truncates the data at the output width determined by the formula. NOTE: two's complement data is output always, and truncation will cause errors or weighting to negative numbers.
idct_mode	0, 1 Default: 0	0:dct(forward) 1:idct(inverse)  Determines the direction of data throughput, and the size of the various ports. If 0, the data is assumed to be pixel or sample data and the forward DCT algorithm is used, the input is bpp, and the output is n/2+bpp. If set to 1, the input is considered to be dct coefficient data, and the input size is determined by the above formula, and the output data is sized at bpp.
co_a	16 bits Default: 23170	Coefficient A  Coefficients are the various dct coefficients as calculated by the DCT formula below.
co_b	16 bits Default: 32138	Coefficient B
co_c	16 bits Default: 30274	Coefficient C
co_d	16 bits Default: 27245	Coefficient D
co_e	16 bits Default: 18205	Coefficient E
co_f	16 bits Default: 12541	Coefficient F

**Table 1-2 Parameter Description**

Parameter	Value	Description
co_g	16 bits Default: 6393	Coefficient G
co_h	16 bits Default: 35355	Coefficient H
co_i	16 bits Default: 49039	Coefficient I
co_j	16 bits Default: 46194	Coefficient J
co_k	16 bits Default: 41573	Coefficient K
co_l	16 bits Default: 27779	Coefficient L
co_m	16 bits Default: 19134	Coefficient M
co_n	16 bits Default: 9755	Coefficient N
co_o	16 bits Default: 35355	Coefficient O
co_p	16 bits Default: 49039	Coefficient P

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

**DW\_decode\_en**

Binary Decoder with Enable

**DW\_decode\_en**

Binary Decoder with Enable

- ❖ Parameterized word length
- ❖ Integrates enable control

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
en	1 bit	Input	Enable input (active high)
a	<i>width</i>	Input	Binary input data
b	$2^{width}$	Output	Decoded output data

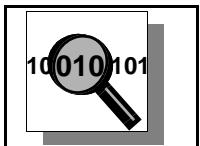
**Table 1-2 Parameter Description**

Parameter	Values	Description
width <sup>a</sup>	1 to 16	Word length of input a is <i>width</i> . Word length of output b is $2^{width}$

- a. The *width* parameter value causes the size of output B to grow exponentially. Therefore, a *width* value greater than 12 will result in abnormally long compile times.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare
cgen	Datapath generator-based Implementation	DesignWare



## DW\_fir

### High-Speed Digital FIR Filter

- ❖ High-speed transposed canonical FIR filter architecture
- ❖ Parameterized coefficient, data, and accumulator word lengths
- ❖ Parameterized filter order
- ❖ Serially loadable coefficients
- ❖ Cascadable architecture for easy partitioning
- ❖ DesignWare datapath generator is employed for better timing and area

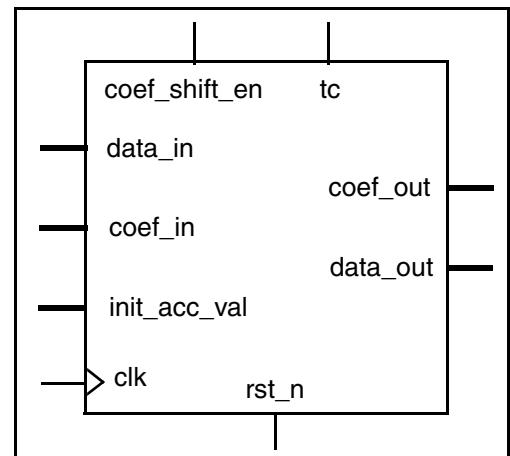
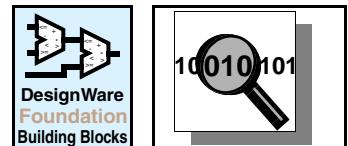


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock. All internal registers are sensitive on the positive edge of clk and all setup and hold times are with respect to this edge of clk.
rst_n	1 bit	Input	Asynchronous reset, active low. Clears all coefficient and data values.
coef_shift_en	1 bit	Input	Enable coefficient shift loading at coef_in, active high.
tc	1 bit	Input	Defines data_in and coef_in values as two's complement or unsigned. If low, the data_in and coef_in values are unsigned; if high, they are two's complement.
data_in	data_in_width bit(s)	Input	Input data.
coef_in	coef_width bit(s)	Input	Serial coefficient coef_shift_en port. This port is enabled when the coef_shift_en pin is set high. A rising edge of clk loads the coefficient data at coef_in into the first internal coefficient register and shifts all other coefficients in the internal registers one location to the right.
init_acc_val	data_out_width bit(s)	Input	Initial accumulated sum value. If unused, this pin is tied to low ("000...000"), that is, when the FIR filter is implemented with a single DW_fir component. When several DW_fir components are cascaded, the data_out of the previous stage is connected to the init_acc_val port of the next.
data_out	data_out_width bit(s)	Output	Accumulated sum of products of the FIR filter.
coef_out	coef_width bit(s)	Output	Serial coefficient output port. When the coef_shift_en pin is high and coefficients are being loaded serially, the coefficient data in the last internal coefficient register is output through the coef_out port.

**DW\_fir**

High-Speed Digital FIR Filter

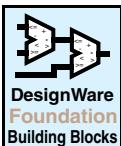
**Table 1-2 Parameter Description**

Parameter	Values	Description
data_in_width	$\geq 1$	Input data word length
coef_width	$\geq 1$	Coefficient word length
data_out_width <sup>a</sup>	$\geq 1$	Accumulator word length
order	2 to 256	FIR filter order

- a. The parameter `data_out_width` is normally set to a value of  $coef\_width + data\_in\_width + margin$ . The value  $coef\_width + data\_in\_width$  accounts for the internal coefficient multiplications. An appropriate margin must be included if the filter coefficients have a gain or are cascaded. The value  $margin \leq \log_2(order)$ .

**Table 1-3 - Synthesis Implementations**

Implementation Name	Function	License Required
str	Structural synthesis model	DesignWare



## DW\_fir\_seq

### Sequential Digital FIR Filter

- ❖ Area-efficient multi-cycle implementation
- ❖ Parameterized coefficient, data, and accumulator word lengths
- ❖ Parameterized filter order
- ❖ Serially loadable coefficients
- ❖ Cascadable architecture for easy partitioning
- ❖ DesignWare datapath generator is employed for better timing and area

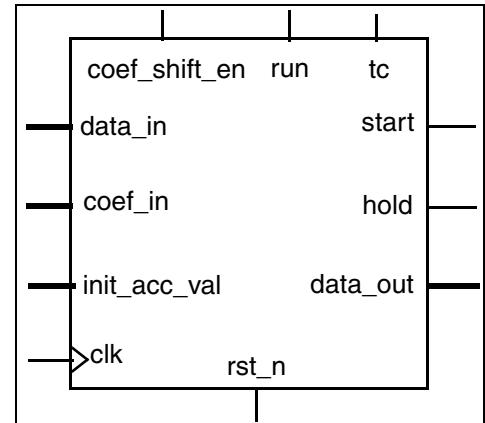
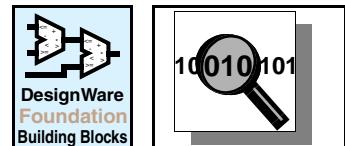


Table 1-1 - Pin Description

Pin Name	Size	Direction	Function
clk	1 bit	Input	Clock. All internal registers are sensitive to the positive edge of clk.
rst_n	1 bit	Input	Asynchronous reset, active low.
coef_shift_en	1 bit	Input	Enable coefficient shift loading at coef_in, active high. This signal is synchronous to the positive edge of clk.
tc	1 bit	Input	Defines data_in and coef_in values as two's complement or unsigned. When low, the data_in and coef_in values are unsigned. When high, the data_in and coef_in values are two's complement.
run	1 bit	Input	Handshake signal that initiates the processing of a data sample on the data_in port. This signal is synchronous to the positive edge of clk.
data_in	<i>data_in_width</i> bit(s)	Input	Input data.
coef_in	<i>coef_width</i> bit(s)	Input	Serial coefficient load port. This port is enabled when the coef_shift_en pin is set high. A rising edge of clk loads the coefficient data at coef_in into the first internal coefficient register and shifts all other coefficients in the internal registers one location to the right.
init_acc_val	<i>data_out_width</i> bit(s)	Input	Initial accumulated value for the convolution sum of products. Normally, set to zero ("000...000").
start	1 bit	Output	Handshake signal generated by synchronizing the run input with clk. It acknowledges the run signal and indicates the start of processing of a data_in sample.

**DW\_fir\_seq**

Sequential Digital FIR Filter

**Table 1-1 - Pin Description**

Pin Name	Size	Direction	Function
hold	1 bit	Output	Handshake signal that indicates processing has been completed for the current data_in sample and the filter is ready to process the next sample.
data_out	<i>data_out_width</i> bit(s)	Output	The accumulated sum of products from the FIR convolution plus the init_acc_val input $\text{init\_acc\_val}(n-1) + \sum_{i=0}^{\text{order}-1} \text{data\_in}(n-i-1) \text{ coef}(i)$

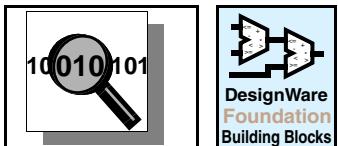
**Table 1-2 Parameter Description**

Parameter	Values	Description
data_in_width	$\geq 1$	Input data word length
coef_width	$\geq 1$	Coefficient word length
data_out_width <sup>a</sup>	$\geq 1$	Accumulator word length
order	2 to 256	FIR filter order

- a. The parameter data\_out\_width is normally set to a value of *coef\_width + data\_in\_width + margin*. The value *coef\_width+data\_in\_width* accounts for the internal coefficient multiplications. An appropriate margin must be included if the filter coefficients have a gain or are cascaded. The value *margin*  $\leq \log_2(\text{order})$ .

**Table 1-3 - Synthesis Implementations**

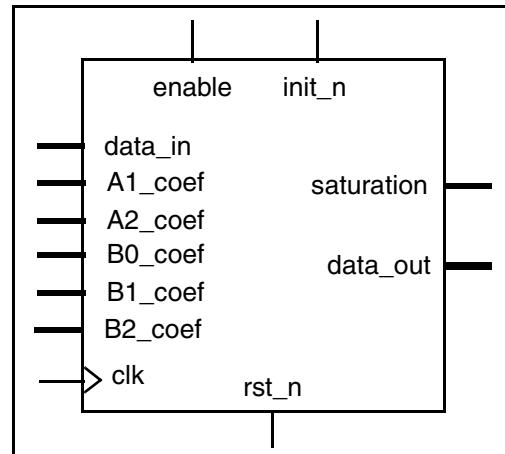
Implementation Name	Function	License Required
str	Structural synthesis model	DesignWare



## DW\_iir\_dc

### High-Speed Digital IIR Filter with Dynamic Coefficients

- ❖ High-speed transposed-form multiplier architecture
- ❖ Variable coefficient values
- ❖ Parameterized coefficient widths
- ❖ DesignWare datapath generator is employed for better timing and area

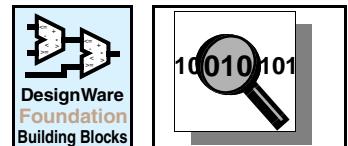


**Table 1-1 Signal Description**

Name	Width	I/O	Description
clk	1 bit	In	Clock signal. All registers are sensitive on the positive edge of clk and all setup and hold times are with respect to this edge of clk.
rst_n	1 bit	In	Asynchronous reset, active-low. Clears all registers.
init_n	1 bit	In	Synchronous, active-low signal to clear all registers.
enable	1 bit	In	Active-high signal to enable all registers.
A1_coef	<i>max_coef_width</i> bit(s)	In	Two's complement value of coefficient A1.
A2_coef	<i>max_coef_width</i> bit(s)	In	Two's complement value of coefficient A2.
B0_coef	<i>max_coef_width</i> bit(s)	In	Two's complement value of coefficient B0.
B1_coef	<i>max_coef_width</i> bit(s)	In	Two's complement value of coefficient B1.
B2_coef	<i>max_coef_width</i> bit(s)	In	Two's complement value of coefficient B2.
data_in	<i>data_in_width</i> bit(s)	In	Input data.
data_out	<i>data_out_width</i> bit(s)	Out	Accumulated sum of products of the IIR filter.
saturation	1 bit	Out	Used to indicate the output data or feedback data is in saturation.

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_in_width	$\geq 2$ , Default = 8	Input data word length

**DW\_iir\_dc**

High-Speed Digital IIR Filter with Dynamic Coefficients

**Table 1-2 Parameter Description (Continued)**

Parameter	Values	Description
data_out_width	$\geq 2$ , Default = 16	Width of output data. This parameter should also satisfy the following equation: $\text{data\_out\_width} \leq \text{maximum}(\text{feedback\_width}, \text{data\_in\_width} + \text{frac\_data\_out\_width}) + \text{max\_coef\_width} + 3 - \text{frac\_coef\_width}$
frac_data_out_width	0 to <i>data_out_width</i> -1 Default = 4	Width of fraction portion of data_out.
feedback_width	$\geq 2$ , Default = 12	Width of feedback_data. (feedback_data is internal to the <model>.)
max_coef_width	$\geq 2$ , Default = 8	Maximum coefficient word length
frac_coef_width	0 to <i>max_coef_width</i> -1 Default = 4	Width of the fraction portion of the coefficients
saturation_mode	0 or 1, Default = 0	Controls the mode of operation of the saturation output
out_reg	0 or 1, Default = 1	Controls whether data_out and saturation are registered

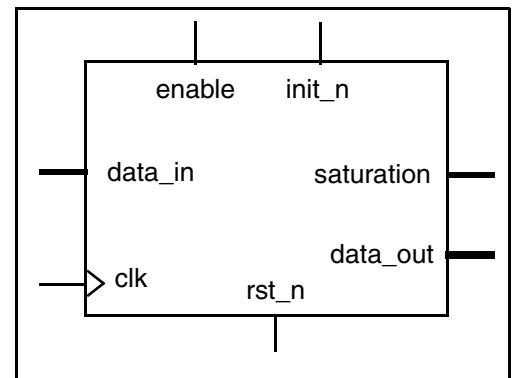
**Table 1-3 - Synthesis Implementations**

Implementation Name	Function	License Required
mult	Structural synthesis model	DesignWare

## DW\_iir\_sc

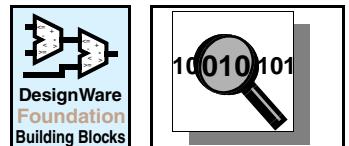
### High-Speed Digital IIR Filter with Static Coefficients

- ❖ High-speed direct-form vector sum architecture
- ❖ High-speed transposed-form multiplier architecture
- ❖ Parameterized input, output, and feedback data widths
- ❖ Parameterized coefficient values and widths
- ❖ Parameterized fraction widths and saturation mode
- ❖ DesignWare datapath generator is employed for better timing and area



**Table 1-1 Signal Description**

Name	Width	I/O	Description
clk	1 bit	Input	Clock signal. All internal registers are sensitive on the positive edge of clk and all setup and hold times are with respect to this edge of clk.
rst_n	1 bit	Input	Synchronous reset, active-low. Clears all registers
init_n	1 bit	Input	Synchronous, active-low signal to clear all registers
enable	1 bit	Input	Active-high signal to enable all registers
data_in	<i>data_in_width</i> bit(s)	Input	Input data.
data_out	<i>data_out_width</i> bit(s)	Output	Accumulated sum of products of the IIR filter.
saturation	1 bit	Output	Used to indicate the output data or feedback data is in saturation.

**DW\_iir\_sc**

High-Speed Digital IIR Filter with Static Coefficients

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_in_width	$\geq 2$ Default = 4	Input data word length
data_out_width	$\geq 2$ Default = 6	Width of output data. This parameter should also satisfy the following equation: $\text{data\_out\_width} \leq \text{maximum}(\text{feedback\_width}, \text{data\_in\_width} + \text{frac\_data\_out\_width}) + \text{max\_coef\_width} + 3 - \text{frac\_coef\_width}$
frac_data_out_width	0 to $\text{data\_out\_width} - 1$ Default = 0	Width of fraction portion of data_out
feedback_width	$\geq 2$ Default = 8	Width of feedback_data (feedback_data is internal to the <model>)
max_coef_width	$\geq 2$ to 31 Default = 4	Maximum coefficient word length
frac_coef_width	0 to $\text{max\_coef\_width} - 1$ Default = 0	Width of the fraction portion of the coefficients
saturation_mode	0 or 1 Default = 1	Controls the mode of operation of the saturation output
out_reg	0 or 1 Default = 1	Controls whether data_out and saturation are registered
A1_coef	<i>range</i> Default = -2	Constant coefficient value A1 $\text{range} = -2^{\text{max\_coef\_width}-1} \text{ to } 2^{\text{max\_coef\_width}-1} - 1$
A2_coef	<i>range</i> Default = 3	Constant coefficient value A2 $\text{range} = -2^{\text{max\_coef\_width}-1} \text{ to } 2^{\text{max\_coef\_width}-1} - 1$
B0_coef	<i>range</i> Default = 5	Constant coefficient value B0 $\text{range} = -2^{\text{max\_coef\_width}-1} \text{ to } 2^{\text{max\_coef\_width}-1} - 1$
B1_coef	<i>range</i> Default = -6	Constant coefficient value B1 $\text{range} = -2^{\text{max\_coef\_width}-1} \text{ to } 2^{\text{max\_coef\_width}-1} - 1$
B2_coef	<i>range</i> Default = -2	Constant coefficient value B2 $\text{range} = -2^{\text{max\_coef\_width}-1} \text{ to } 2^{\text{max\_coef\_width}-1} - 1$

**Table 1-3 - Synthesis Implementations**

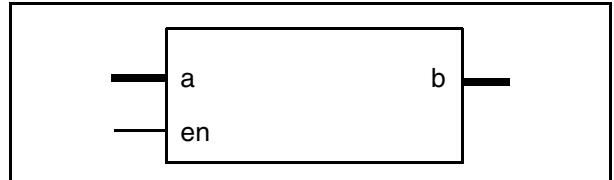
Implementation Name	Function	License Required
mult	Multiplier synthesis model	DesignWare
vsum	Vector sum synthesis model	DesignWare



## DW\_thermdec

Binary Thermometer Decoder with Enable

- ❖ Parameterized word length
- ❖ Integrates enable control



### Description

DW\_thermdec decodes a binary value present at input port *a* and sets contiguous bits of the output port *b*, depending on the enable input (*en*). A thermometer decoder with input *width* = *n* bits has  $2^n$  bits at the output, where each output with index  $j \leq n$  becomes active when *en* = 1 (enable input) and input *a* = *i*. The output bits are active high. When *en* = 0 none of the output bits are active.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
en	1 bit	Input	Enable input (active high)
a	<i>width</i>	Input	Binary input data
b	$2^{\text{width}}$	Output	Decoded output data

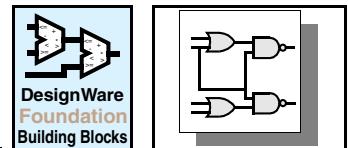
**Table 1-2 Parameter Description**

Parameter	Values	Description
width <sup>a</sup>	1 to 16	Word length of input a is <i>width</i> . Word length of output b is $2^{\text{width}}$

- a. The *width* parameter value causes the size of output *b* to grow exponentially. Therefore, a *width* value greater than 12 will result in abnormally long compile times.

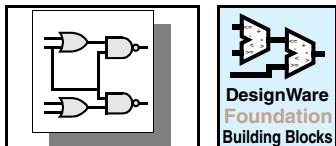
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
cgen	Datapath generator-based Implementation	DesignWare



## Logic – Combinational Overview

The combinational components consist of high-performance logical components. Most components in this category have multiple architectures for each function (architecturally optimized for either performance or area) to provide you with the best architecture for your design goals. All components have a parameterized word length.



# DW01\_binenc

## Binary Encoder

- ❖ Parameterized word length
- ❖ Inferable using a function call



Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	$A\_width$	Input	Input data
ADDR	$ADDR\_width$	Output	Binary encoded output data

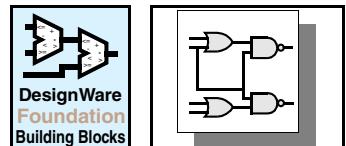
Table 1-2 Parameter Description

Parameter	Values	Description
$A\_width$	$\geq 1$	Word length of input A
$ADDR\_width$	$\geq \text{ceil}(\log_2[A\_width])$	Word length of output ADDR

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare
cla <sup>a</sup>	Synthesis model	DesignWare
aot	Synthesis model	DesignWare

a. The 'cla' implementation is only available when  $A\_width \leq 12$ .

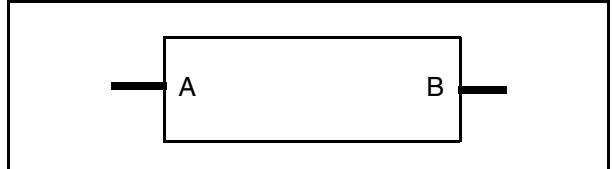
**DW01\_decode**

Decoder

**DW01\_decode**

Decoder

- ❖ Parameterized word length
- ❖ Inferable using a function call

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
A	$width$	Input	Binary input data
B	$2^{width}$	Output	Decoded output data

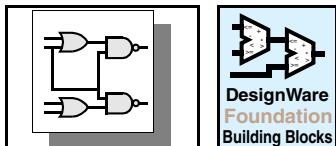
**Table 1-2 Parameter Description**

Parameter	Values	Description
width <sup>a</sup>	$\geq 1$	Word length of input A is $width$ . Word length of output B is $2^{width}$

- a. The width parameter value causes the size of output B to grow exponentially. Therefore, a width value greater than 12 will result in abnormally long compile times.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	none

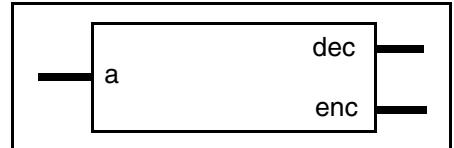


## DW\_lod

### Leading One's Detector

- ❖ Parameterized word length
- ❖ Inferable using a function call

### Description



DW\_lod contains two outputs, dec and enc. The dec output is a decoded one-hot value of the a input vector assuming there is at least one 0 on a. The output enc represents the number of 1s found (from the most significant bit) before the first occurrence of a 0 from the input port a. All lower order bits (to the right) from the first occurrence of the 0 on the a input port are “don't care.” If no 0 is found and only 1s are present, the resulting value of enc is all 1s and of dec is all 0s.

The output port enc width is automatically derived from the input port width parameter, a\_width, and is defined as  $\text{ceil}(\log_2[\text{a\_width}])+1$  as listed in [Table 1-1](#). Output port dec has the same width as the a input.

**Table 1-1 Pin Description**

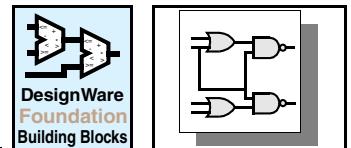
Pin Name	Width	Direction	Function
a	a_width	Input	Input vector
dec	a_width	Output	One-hot decode of a - All 0s if a is all 1s.
enc	$\text{ceil}(\log_2[\text{a\_width}])+1$	Output	Number of leading 1s in a before first 0. All 1s if a is all 1s.

**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\geq 1$	Vector width of input a

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
cla	Synthesis model	DesignWare
rtl	Synthesis model	DesignWare

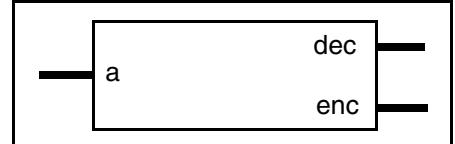
**DW\_Isd**

Leading Signs Detector

**DW\_Isd**

Leading Signs Detector

- ❖ Parameterized word length
- ❖ Inferable using a function call
- ❖ Optional C++ simulation model available in the DWFC Library

**Table 1-1 Pin Description**

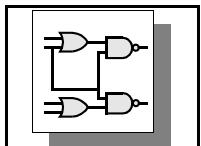
Pin Name	Width	Direction	Function
a	a_width	Input	Input vector
dec	a_width	Output	One-hot decode of a.
enc	$\text{ceil}(\log_2[\text{a\_width}])$	Output	Number of leading sign bits in a before the least significant sign bit.

**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\leq 1$ Default: 8	Vector width of input a

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



## DW\_Iza

### Leading Zero's Anticipator

- ❖ Parameterized word length
- ❖ Inferable through function call



### Description

The DW\_Iza takes as its input the normalized and swapped unsigned values to the subtraction. The inputs are designed to have  $a \geq b$  and of the same width, and the output will be the number of most significant zeros *anticipated* to be in the result. This anticipated value may be off by one bit less than the true required shift. The shift count generated by this component, if in error, will be one less than the actual number of shifts required by the addition.

**Table 1-1 Pin Description**

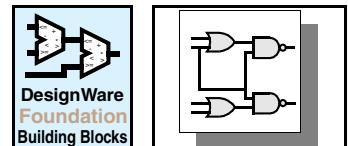
Pin Name	Width	Direction	Function
a	width	Input	a input
b	width	Input	b input
count	$\text{ceil}(\log_2[\text{width}])$	Output	Count of leading zeros

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$2 \leq \text{width} \leq 256$	Width of a and b inputs

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

**DW\_lzd**

Leading Zero's Detector

**DW\_lzd**

Leading Zero's Detector

- ❖ Parameterized word length
- ❖ Inferable using a function call
- ❖ Optional C++ simulation model available in the DWFC Library

**Description**

DW\_lzd contains two outputs, dec and enc. The dec output is a decoded one-hot value of the a input vector assuming there is at least one 1 on a. The output enc represents the number of 0s found (from the most significant bit) before the first occurrence of a 1 from the input port a. All lower order bits (to the right) from the first occurrence of the "1" on the a input port are "don't care." If no 1 is found and only 0s are present, the resulting value of enc is all 1s and of dec is all 0s.

The output port enc width is automatically derived from the input port width parameter, *a\_width*, and is defined as  $\text{ceil}(\log_2[a\_width]) + 1$  as listed in [Table 1-1](#). Output port dec has the same width as the a input.

**Table 1-1 Pin Description**

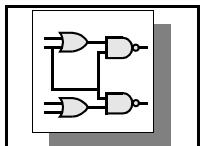
Pin Name	Width	Direction	Function
a	<i>a_width</i>	Input	Input vector
dec	<i>a_width</i>	Output	One-hot decode of a - All 0s if a is all 0s.
enc	$\text{ceil}(\log_2[a\_width]) + 1$	Output	Number of leading 0s in a before first 1. All 1s if a is all 0s.

**Table 1-2 Parameter Description**

Parameter	Values	Description
<i>a_width</i>	$\geq 1$ Default: 8	Vector width of input a

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
cla	Synthesis model	DesignWare
rtl	Synthesis model	DesignWare



## DW01\_mux\_any

Universal Multiplexer

- ❖ Parameterized word lengths
- ❖ Saves coding time by eliminating the need to code muxes explicitly
- ❖ Increases design abstraction
- ❖ Uses 8-to-1 muxes where possible

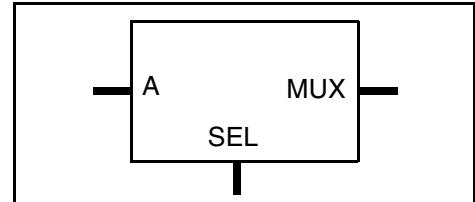


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	$A\_width$	Input	Data input bus
SEL	$SEL\_width$	Input	Select input
MUX	$MUX\_width$	Output	Multiplexed data out

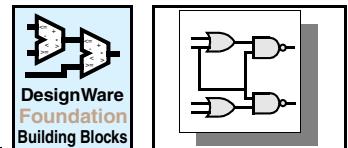
Table 1-2 Parameter Description

Parameter	Values	Description
$A\_width$	$\geq 1$	Word length of A
$SEL\_width$	$\geq 1$	Word length of SEL
$MUX\_width$	$\geq 1$	$A((SEL + 1) \times MUX\_width - 1)$ downto $SEL * MUX\_width$
bal_str <sup>a</sup>	0 or 1 Default: 0	Controls the symmetric structure of the 'str' implementation. 1 = use symmetric structure with 'str' implementation.

a. Prior to release 2007.12-SP2, this parameter is ignored.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



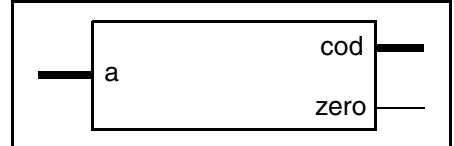
## DW\_pricod

Priority Coder

# DW\_pricod

Priority Coder

- ❖ Parameterized word length
- ❖ Inferable using a function call



## Description

The cod output of DW\_pricod is a coded one-hot value of the a input vector with a 1 at the most significant (left-most) non-zero bit position of a. All lower order bits (to the right) from the first occurrence of a 1 on the a input port are “don’t care.” The zero output indicates whether all bits of input a are 0. If no 1 is found and only 0s are present, the resulting value of cod is all 0s and the value of zero is 1.

**Table 1-1 Pin Description**

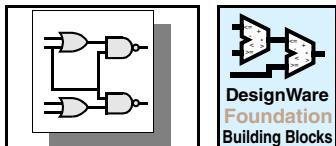
Pin Name	Width	Direction	Function
a	a_width	Input	Input vector
cod	a_width	Output	One-hot coded value of a.
zero	1	Output	All-zero flag (= 1 if all bits of a are 0)

**Table 1-2 Parameter Description**

Parameter	Values	Description
a_width	$\geq 1$	Vector width of input a

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare
cla	Synthesis model	DesignWare



## DW01\_prienc

### Priority Encoder

- ❖ Parameterized word length
- ❖ Inferable using a function call



Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	$A\_width$	Input	Input data
INDEX	$INDEX\_width$	Output	Binary encoded output data

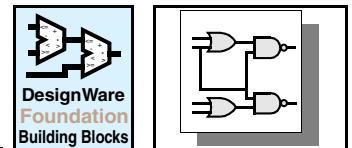
Table 1-2 Parameter Description

Parameter	Values	Description
A_width	$\geq 1$	Word length of input A
INDEX_width	$\geq \text{ceil}(\log_2[A\_width])$	Word length of output INDEX

Table 1-3 Synthesis Implementations

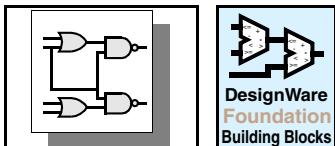
Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare
cla <sup>a</sup>	Synthesis model	DesignWare
aot	Synthesis model	DesignWare

a. The 'cla' implementation is only available when  $A\_width \leq 512$ .



## Logic – Sequential Overview

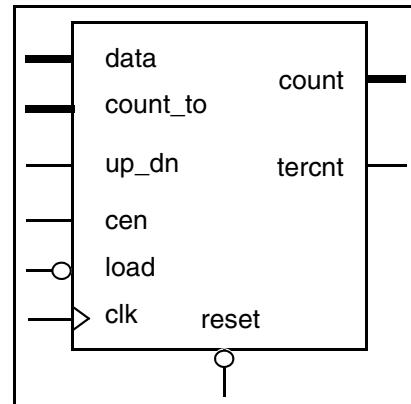
The sequential components consist of high-performance counters, many with either dynamic or static count-to flags. Components in this category have multiple architectures for each function (architecturally optimized for either performance or area) to provide you with the best architecture for your design goals. All components have a parameterized word length.



## DW03\_bictr\_dcnto

Up/Down Binary Counter with Dynamic Count-to Flag

- ❖ Parameterized word length
- ❖ Terminal count flag for count-to comparison
- ❖ Pin-programmable count-to value
- ❖ Up/down count control
- ❖ Asynchronous reset
- ❖ Synchronous counter load
- ❖ Synchronous count enable
- ❖ Provides minPower benefits with the DesignWare-LP license.



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data	<i>width</i>	Input	Counter load input
count_to	<i>width</i>	Input	Count compare 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	<i>width</i>	Output	Output count bus
tercnt	1	Output	Terminal count flag, active high

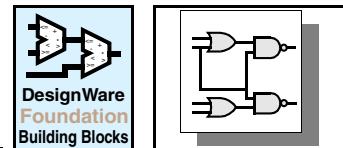
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	$\geq 1$	Width of data input bus

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare
lpwr <sup>a</sup>	Low Power Synthesis model	DesignWare-LP

a. Effectiveness of low power design depends on the use of the `-gate_clock` option to `compile_ultra` command

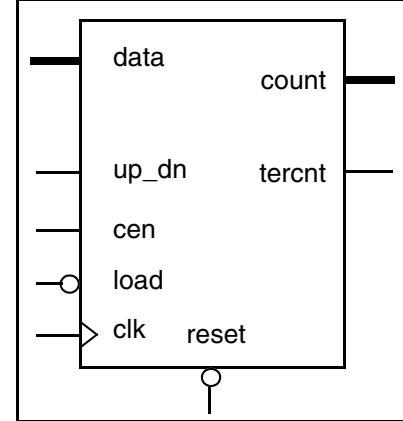
**DW03\_bictr\_scnto**

Up/Down Binary Counter with Static Count-to Flag

**DW03\_bictr\_scnto**

Up/Down Binary Counter with Static Count-to Flag

- ❖ Parameterized word length
- ❖ Parameterized count-to value
- ❖ Up/down count control
- ❖ Asynchronous reset
- ❖ Loadable count register
- ❖ Terminal count flag
- ❖ Counter enable
- ❖ Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data	<i>width</i>	Input	Counter load input
up_dn	1 bit	Input	High for count up and low for count down
load	1 bit	Input	Enable data load to counter, active low
cen	1 bit	Input	Count enable, active high
clk	1 bit	Input	Clock
reset	1 bit	Input	Counter reset, active low
count	<i>width</i>	Output	Output count bus
tercnt	1 bit	Output	Terminal count flag

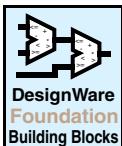
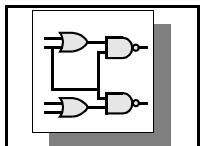
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	1 to 30	Width of data and count
count_to	1 to $2^{width-1}$	Count-to value

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare
lpwr <sup>a</sup>	Low Power Synthesis model	DesignWare-LP

a. Effectiveness of low power design depends on the use of the `-gate_clock` option to `compile_ultra` command



## DW03\_bictr\_decode

Up/Down Binary Counter with Output Decode

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

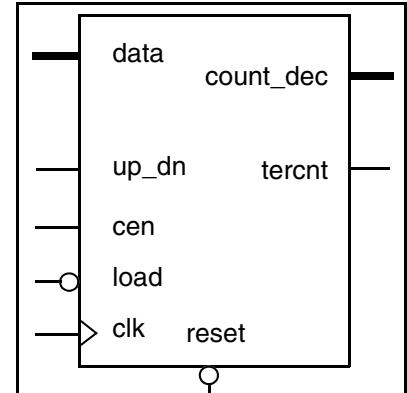


Table 1-1 Pin Description

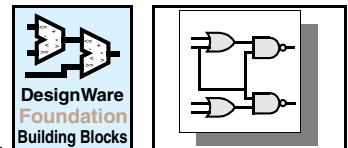
Pin Name	Width	Direction	Function
data	<i>width</i>	Input	Counter load input
up_dn	1	Input	High for count up and low for count down
load	1	Input	Enable data load to counter, active low
cen	1	Input	Count enable, active high
clk	1	Input	Clock
reset	1	Input	Counter reset, active low
count_dec	$2^{width}$	Output	Binary decoded count value
tercnt	1	Output	Terminal count flag

Table 1-2 Parameter Description

Parameter	Values	Function
width	$\geq 1$	Width of data input bus

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

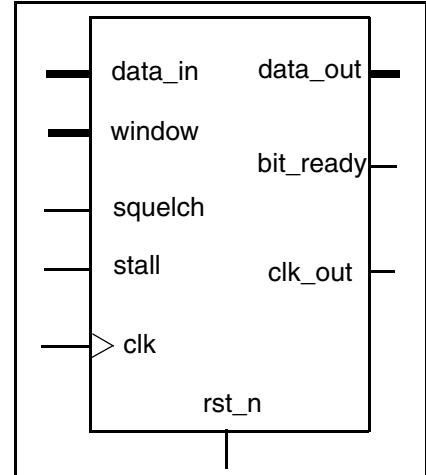
**DW\_dpll\_sd**

Digital Phase Locked Loop

**DW\_dpll\_sd**

Digital Phase Locked Loop

- ❖ Parameterizable divisor (ratio of reference clock to baud rate)
- ❖ Multichannel data recovery (recovery of channels that accompany the locked channel)
- ❖ Stall input for power saving mode and/or prescaler (allowing one DW\_dpll\_sd to recover data at multiple rates)
- ❖ Squelch input for ignoring phase information when channel data is unknown or unconnected
- ❖ Sampling window control to aid data recovery under harsh conditions
- ❖ Parameterizable gain to meet a variety of application needs
- ❖ Parameterizable filter (controls phase correction reactivity from minor phase errors)
- ❖ Provides minPower benefits with the DesignWare-LP license.

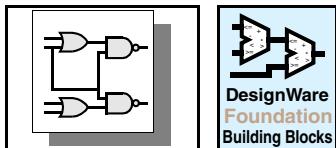
**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Reference clock
rst_n	1 bit	Input	Asynchronous reset, active low
stall	1 bit	Input	Stalls everything except synchronizer, active high
squelch	1 bit	Input	Turns off phase detection. When high no phase correction is carried out leaving DPLL free running, active high
window	ceil(log2(windows))	Input	Sampling window selector <sup>a</sup>
data_in	width bit(s)	Input	Serial input data stream
clk_out	1 bit	Output	Recovered Clock
bit_ready	1 bit	Output	Output data ready flag
data_out	width bit(s)	Output	Recovered output data stream

a. The minimum value must be 1.

**Table 1-2 Parameter Description**

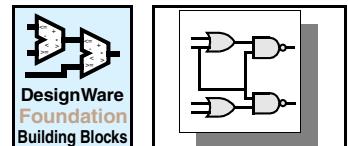
Parameter	Values	Description
width	1 to 16 Default: 1	Number of input serial channels
divisor	4 to 256 Default: 4	Determines the number of samples per input clock cycle

**Table 1-2 Parameter Description (Continued)**

Parameter	Values	Description
gain	1 to 2 Default: 1	Phase correction factor for the absolute value of clock phase error greater than  1  1 = 50% phase correction 2 = 100% phase correction
filter	0 to 8 Default: 2	Phase correction control for +/- 1 clock phase error region. 0 = no correction 1 = always correct For integer N > 1, correct after N samples at a current phase (such as, N consecutive samples at +1 or N consecutive samples at -1)
windows	1 to (divisor+1)/2 Default: 1	Number of sampling windows for the input serial data stream

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

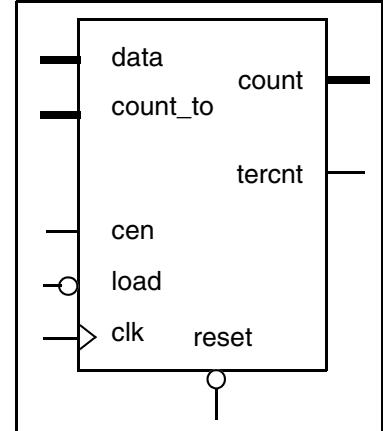
**DW03\_Lfsr\_dcnto**

LFSR Counter with Dynamic Count-to Flag

**DW03\_Lfsr\_dcnto**

LFSR Counter with Dynamic Count-to Flag

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

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data	<i>width</i> bit(s)	Input	Input data
count_to	<i>width</i> 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	<i>width</i> bit(s)	Output	Output count bus
tercnt	1 bit	Output	Output terminal count

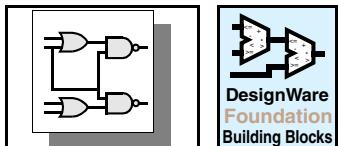
**Table 1-2 Parameter Description**

Parameter	Legal Range <sup>a</sup>	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**

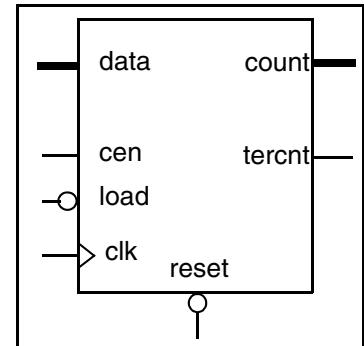
Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



## DW03\_Lfsr\_scnto

LFSR Counter with Static Count-to Flag

- ❖ Parameterized count-to value to indicate when the counter reaches a specified value
- ❖ Parameterized word length
- ❖ High speed, area-efficient
- ❖ Asynchronous reset
- ❖ Terminal count flag



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data	<i>width</i> bit(s)	Input	Input data
load	1 bit	Input	Input load, active low
cen	1 bit	Input	Input count enable
clk	1 bit	Input	Clock
reset	1	Input	Asynchronous reset, active low
count	<i>width</i> bit(s)	Output	Output count bus
tercnt	1 bit	Output	Output terminal count

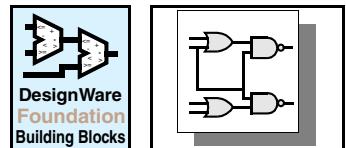
**Table 1-2 Parameter Description**

Parameter	Values <sup>a</sup>	Function
width	2 to 50	Word length of counter
count_to	1 to $2^{width-2}$	count_to bus

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

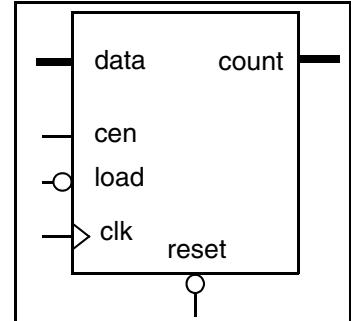
**DW03\_Lfsr\_load**

LFSR Counter with Loadable Input

**DW03\_Lfsr\_load**

LFSR Counter with Loadable Input

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

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data	<i>width</i> 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	<i>width</i> bit(s)	Output	Output count bus

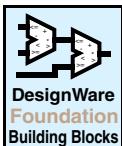
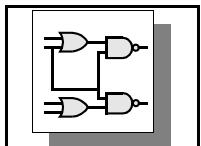
**Table 1-2 Parameter Description**

Parameter	Values <sup>a</sup>	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



## DW03\_Lfsr\_updn

LFSR Up/Down Counter

- ❖ High speed, area-efficient
- ❖ Pseudorandom sequence generator
- ❖ Up/down count control
- ❖ Asynchronous reset
- ❖ Terminal count flag

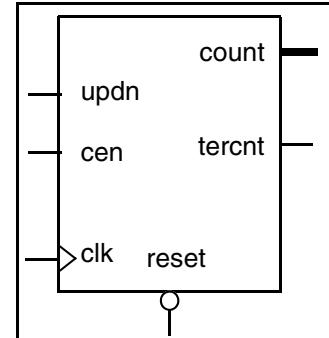


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
updn	1 bit	Input	Input high for count up and low for count down
cen	1 bit	Input	Input count enable
clk	1 bit	Input	Clock
reset	1 bit	Input	Asynchronous reset, active low
count	<i>width</i> bit(s)	Output	Output count bus
tercnt	1 bit	Output	Output terminal count

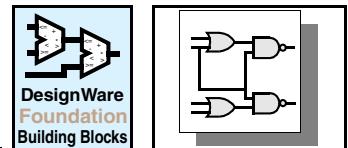
Table 1-2 Parameter Description

Parameter	Values <sup>a</sup>	Description
width	2 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



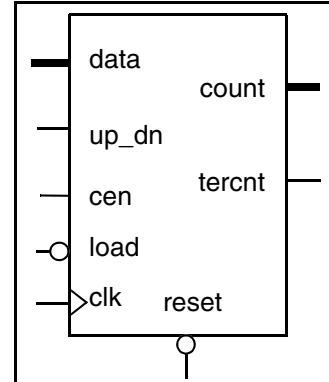
## DW03\_updn\_ctr

### Up/Down Counter

## DW03\_updn\_ctr

### Up/Down Counter

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



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data	<i>width</i> bit(s)	Input	Input data bus
up_dn	1 bit	Input	Count up ( <i>up_dn</i> = 1) or count down ( <i>up_dn</i> = 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	<i>width</i> bit(s)	Output	Output count bus
tercnt	1 bit	Output	Terminal count flag

**Table 1-2 Parameter Description**

Parameter	Value	Function
width	$\geq 1$	Width of count output bus

**Table 1-3 Synthesis Implementations<sup>a</sup>**

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

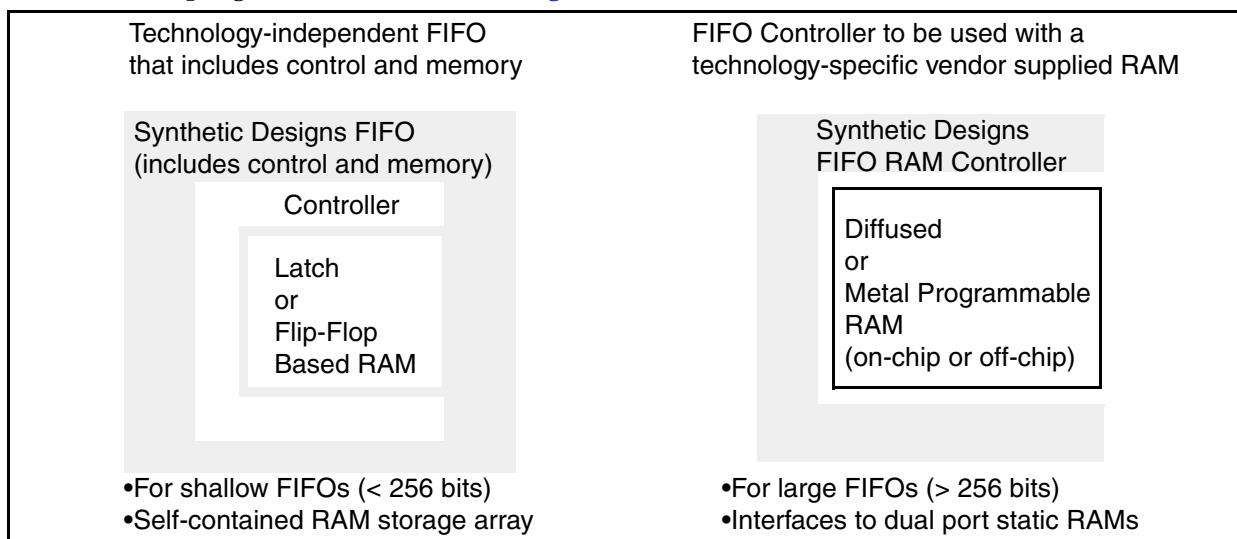
a. During synthesis, Design Compiler will select the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table. For more details, please refer to the *DesignWare Building Block IP User Guide*.

## Memory – FIFO Overview

The FIFOs in this category address a broad array of design requirements. FIFOs, which include dual-port RAM memory arrays, are offered for both synchronous and asynchronous interfaces. The memory arrays are offered in two configurations: latch-based to minimize area, and D flip-flop-based to maximize testability. These two configurations also offer flexibility when working under design constraints, such as a requirement that no latches be employed. Flip-flop-based designs employ no clock gating to minimize skew and maximize performance. All FIFOs employ a FIFO RAM controller architecture in which there is no extended “fall-through” time required before reading contents just written.

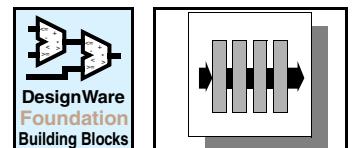
Also offered are FIFO Controllers without the RAM array. They consist of control and flag logic and an interface to common ASIC dual port RAMs. Choosing between the two is typically based on the required size of the FIFO. For shallow FIFOs (less than 256 bits), synchronous or asynchronous FIFOs are available which include both memory and control in a single macro. These macros can be programmed via word width, depth, and level (almost-full flag) parameters.

For larger applications (greater than 256 bits), you can use the asynchronous FIFO Controller with a diffused or metal programmable RAM. See [Figure 1-1](#).



**Figure 1-1 Memory: FIFOs and FIFO Controllers**

All FIFOs and Controllers support full, empty, and programmable flag logic. Programmable flag logic may be statically or dynamically programmed. When statically programmed, the threshold comparison value is hardwired at synthesis compile time. When dynamically programmed, it may be changed during FIFO operation.

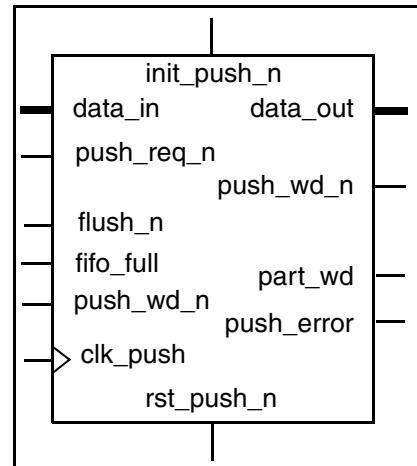
**DW\_asymdata\_inbuf**

Asymmetric Data Input Buffer

**DW\_asymdata\_inbuf**

Asymmetric Data Input Buffer

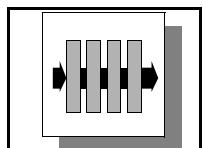
- ❖ Parameterized asymmetric input and output data widths ( $in\_width < out\_width$  with integer multiple relationship)
- ❖ Parameterized byte (sub-word) ordering within a word
- ❖ Parameterized *flush\_value*
- ❖ Word integrity flag
- ❖ Parameterized error status mode
- ❖ Registered push error (overflow)

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk_push	1	Input	Clock source
rst_push_n	1	Input	Asynchronous reset (active low)
init_push_n	1	Input	Synchronous reset (active low)
push_req_n	1	Input	Push request (active low)
data_in	<i>in_width</i>	Input	Input data (word)
flush_n	1	Input	Flush the partial word (active low)
fifo_full	1	Input	Full indication connected RAM/FIFO
push_wd_n	1	Output	Ready to write full data word (active low)
data_out	<i>out_width</i>	Output	Output data (sub-word)
inbuf_full	1	Output	Input registers all contain active data_in sub-words
part_wd	1	Output	Partial word pushed flag
push_error	1	Output	Overrun of RAM/FIFO (includes input registers)

**Table 1-2 Parameter Description**

Parameter	Values	Description
in_width	1 to 256 Default: 8	Width of data_in Must be less than <i>out_width</i> by an integer multiple
out_width	1 to 256 Default: 16	Width of data_out Must be greater than <i>in_width</i> by an integer multiple
err_mode	0 or 1 Default: 0	Error flag behavior mode 0 = sticky push_error flag (hold on first occurrence, clears only on reset) 1 = dynamic push_error flag (reports every occurrence of error)

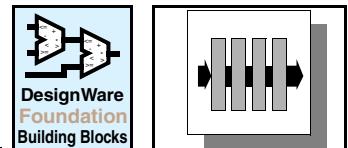


**Table 1-2 Parameter Description**

Parameter	Values	Description
byte_order	0 or 1 Default: 0	Sub-word ordering into Word 0 = the first byte (or sub-word) is in MSB of word 1 = the first byte (or sub-word) is in LSB of word
flush_value	0 or 1 Default: 0	Fixed flushing value 0 = fill empty bits of partial word with 0's upon flush 1 = fill empty bits of partial word with 1's upon flush

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

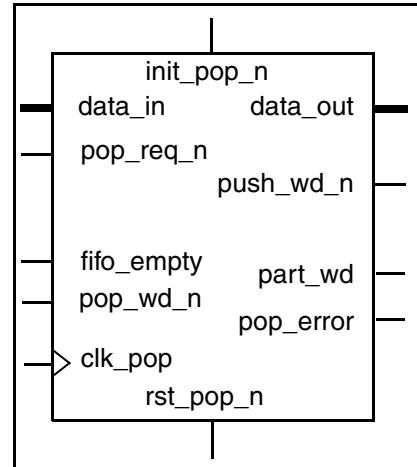
**DW\_asymdata\_outbuf**

Asymmetric Data Output Buffer

**DW\_asymdata\_outbuf**

Asymmetric Data Output Buffer

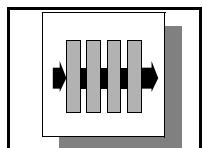
- ❖ Parameterized asymmetric input and output data widths ( $in\_width > out\_width$  with integer multiple relationship)
- ❖ Parameterized byte (sub-word) ordering within a word
- ❖ Parameterized flush value
- ❖ Word integrity flag
- ❖ Parameterized error status mode
- ❖ Registered push error (overflow)

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk_pop	1	Input	Clock source
rst_pop_n	1	Input	Asynchronous reset (active low)
init_pop_n	1	Input	Synchronous reset (active low)
pop_req_n	1	Input	Pop request (active low)
data_in	<i>in_width</i>	Input	Input data (sub-word)
fifo_empty	1	Input	Empty indication connected RAM/FIFO
pop_wd_n	1	Output	Full data word transferred (active low)
data_out	<i>out_width</i>	Output	Output data (sub-word)
part_wd	1	Output	Partial word popped flag
pop_error	1	Output	Underrun of RAM/FIFO

**Table 1-2 Parameter Description**

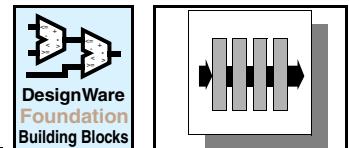
Parameter	Values	Description
in_width	1 to 256 Default: 16	Width of data_in Must be greater than <i>out_width</i> by an integer multiple
out_width	1 to 256 Default: 8	Width of data_out Must be less than <i>in_width</i> by an integer multiple
err_mode	0 or 1 Default: 0	Error flag behavior mode 0 = sticky pop_error flag (hold on first occurrence, clears only on reset) 1 = dynamic pop_error flag (reports every occurrence of error)

**Table 1-2 Parameter Description**

Parameter	Values	Description
byte_order	0 or 1 Default: 0	Sub-word ordering into Word 0 = the first byte (or sub-word) is in MSB of word 1 = the first byte (or sub-word) is in LSB of word

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

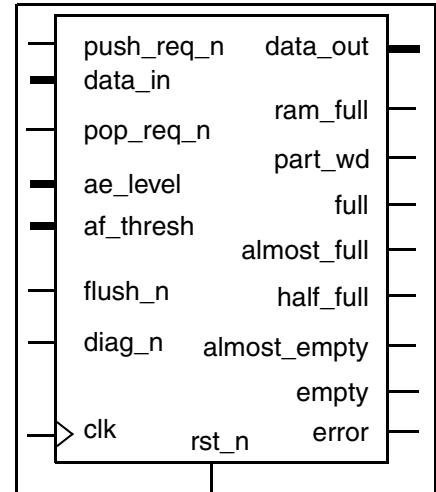
**DW\_asymfifo\_s1\_df**

Asymmetric I/O Synchronous (Single Clock) FIFO with Dynamic Flag

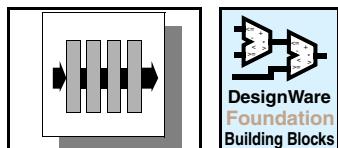
**DW\_asymfifo\_s1\_df**

Asymmetric I/O Synchronous (Single Clock) FIFO with Dynamic Flag

- ❖ Fully registered synchronous flag output ports
- ❖ D flip-flop-based memory array for high testability
- ❖ All operations execute in a single clock cycle
- ❖ FIFO empty, half full, and full flags
- ❖ Parameterized asymmetric input and output bit widths (must be integer-multiple relationship)
- ❖ Word integrity flag for  $data\_in\_width < data\_out\_width$
- ❖ Flushing out partial word for  $data\_in\_width < data\_out\_width$
- ❖ Parameterized byte (or subword) order within a word
- ❖ FIFO error flag indicating underflow, overflow, and pointer corruption
- ❖ Parameterized word depth
- ❖ Dynamically programmable almost full and almost empty flags
- ❖ Parameterized reset mode (synchronous or asynchronous, memory array initialized or not)

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset input, active low (asynchronous if <code>rst_mode = 0</code> , synchronous if <code>rst_mode = 1</code> )
push_req_n	1 bit	Input	FIFO push request, active low
flush_n	1 bit	Input	Flushes the partial word into memory (fills in 0's) (for $data\_in\_width < data\_out\_width$ only)
pop_req_n	1 bit	Input	FIFO pop request, active low
diag_n	1 bit	Input	Diagnostic control, active low (for <code>err_mode = 0</code> , NC for other <code>err_mode</code> values)
data_in	$data\_in\_width$ bit(s)	Input	FIFO data to push
ae_level	$\text{ceil}(\log_2[depth])$ bit(s)	Input	Almost empty level (the number of words in the FIFO at or below which the <code>almost_empty</code> flag is active)
af_thresh	$\text{ceil}(\log_2[depth])$ bit(s)	Input	Almost full threshold (the number of words stored in the FIFO at or above which the <code>almost_full</code> flag is active)
empty	1 bit	Output	FIFO empty output, active high
almost_empty	1 bit	Output	FIFO almost empty output, active high, asserted when FIFO level $\leq ae\_level$
half_full	1 bit	Output	FIFO half full output, active high

**Table 1-1 Pin Description (Continued)**

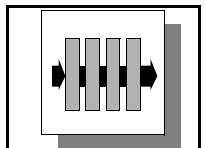
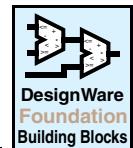
Pin Name	Width	Direction	Function
almost_full	1 bit	Output	FIFO almost full output, active high, asserted when FIFO level $\geq$ (af_thresh)
full	1 bit	Output	FIFO full output, active high
ram_full	1 bit	Output	RAM full output, active high
error	1 bit	Output	FIFO error output, active high
part_wd	1 bit	Output	Partial word, active high (for $data\_in\_width < data\_out\_width$ only; otherwise, tied low)
data_out	$data\_out\_width$ bit(s)	Output	FIFO data to pop

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_in_width	1 to 256	Width of the data_in bus. data_in_width must be in an integer-multiple relationship with data_out_width. That is, either $data\_in\_width = K \times data\_out\_width$ , or $data\_out\_width = K \times data\_in\_width$ , where K is an integer.
data_out_width	1 to 256	Width of the data_out bus. data_out_width must be in an integer-multiple relationship with data_in_width. That is, either $data\_in\_width = K \times data\_out\_width$ , or $data\_out\_width = K \times data\_in\_width$ , where K is an integer.
depth	2 to 256	Number of memory elements used in the FIFO (addr_width = ceil[log <sub>2</sub> (depth)])
err_mode	0 to 2 Default: 1	Error mode 0 = underflow/overflow with pointer latched checking, 1 = underflow/overflow latched checking, 2 = underflow/overflow unlatched checking.
rst_mode	0 to 3 Default: 1	Reset mode 0 = asynchronous reset including memory, 1 = synchronous reset including memory, 2 = asynchronous reset excluding memory, 3 = synchronous reset excluding memory.
byte_order	0 or 1 Default: 0	Order of send/receive bytes or subword [subword - 8 bits - subword] within a word 0 = first byte is in most significant bits position; 1 = first byte is in the least significant bits position [valid for $data\_in\_width \neq data\_out\_width$ ].

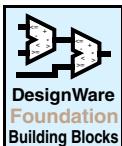
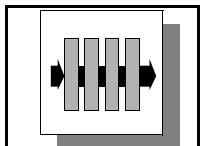
**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cl1	Partial carry look-ahead model	DesignWare
cl2	Full carry look-ahead model	DesignWare

**DW\_asymfifo\_s1\_df**

Asymmetric I/O Synchronous (Single Clock) FIFO with Dynamic Flag

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*



## DW\_asymfifo\_s1\_sf

Asymmetric I/O Synchronous (Single Clock) FIFO with Static Flags

- ❖ Fully registered synchronous flag output ports
- ❖ D flip-flop-based memory array for high testability
- ❖ All operations execute in a single clock cycle
- ❖ FIFO empty, half full, and full flags
- ❖ Parameterized asymmetric input and output bit widths (must be integer-multiple relationship)
- ❖ Word integrity flag for  $data\_in\_width < data\_out\_width$
- ❖ Flushing out partial word for  $data\_in\_width < data\_out\_width$
- ❖ Parameterized byte (or subword) order within a word
- ❖ FIFO error flag indicating underflow, overflow, and pointer corruption
- ❖ Parameterized word depth
- ❖ Parameterized almost full and almost empty flags
- ❖ Parameterized reset mode (synchronous or asynchronous, memory array initialized or not)

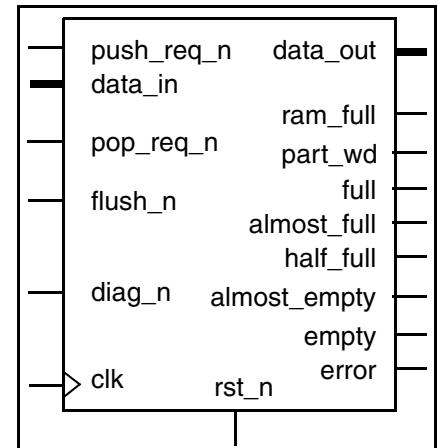
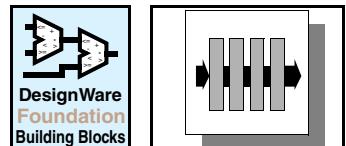


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset input, active low asynchronous if $rst\_mode = 0$ , synchronous if $rst\_mode = 1$
push_req_n	1 bit	Input	FIFO push request, active low
flush_n	1 bit	Input	Flushes the partial word into memory (fills in 0's) (for $data\_in\_width < data\_out\_width$ only)
pop_req_n	1 bit	Input	FIFO pop request, active low
diag_n	1 bit	Input	Diagnostic control, active low (for $err\_mode = 0$ , NC for other $err\_mode$ values)
data_in	$data\_in\_width$ bit(s)	Input	FIFO data to push
empty	1 bit	Output	FIFO empty output, active high
almost_empty	1 bit	Output	FIFO almost empty output, active high, asserted when FIFO level $\leq ae\_level$
half_full	1 bit	Output	FIFO half full output, active high
almost_full	1 bit	Output	FIFO almost full output, active high, asserted when FIFO level $\geq (depth - af\_level)$
full	1 bit	Output	FIFO full output, active high

**DW\_asymfifo\_s1\_sf**

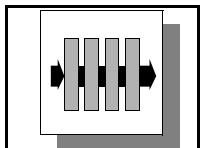
Asymmetric I/O Synchronous (Single Clock) FIFO with Static Flags

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
ram_full	1 bit	Output	RAM full output, active high
error	1 bit	Output	FIFO error output, active high
part_wd	1 bit	Output	Partial word, active high (for $\text{data\_in\_width} < \text{data\_out\_width}$ only; otherwise, tied low)
data_out	$\text{data\_out\_width}$ bit(s)	Output	FIFO data to pop

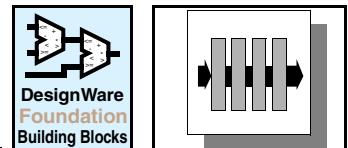
**Table 1-2 Parameter Description**

Parameter	Values	Description
data_in_width	1 to 256	Width of the data_in bus. data_in_width must be in an integer-multiple relationship with data_out_width. That is, either $\text{data\_in\_width} = K \times \text{data\_out\_width}$ , or $\text{data\_out\_width} = K \times \text{data\_in\_width}$ , where $K$ is an integer.
data_out_width	1 to 256	Width of the data_out bus. data_out_width must be in an integer-multiple relationship with data_in_width. That is, either $\text{data\_in\_width} = K \times \text{data\_out\_width}$ , or $\text{data\_out\_width} = K \times \text{data\_in\_width}$ , where $K$ is an integer.
depth	2 to 256	Number of memory elements used in the FIFO ( $\text{addr\_width} = \text{ceil}[\log_2(\text{depth})]$ )
ae_level	1 to $\text{depth} - 1$	Almost empty level (the number of words in the FIFO at or below which the almost_empty flag is active)
af_level	1 to $\text{depth} - 1$	Almost full level (the number of empty memory locations in the FIFO at which the almost_full flag is active).
err_mode	0 to 2 Default: 1	Error mode 0 = underflow/overflow with pointer latched checking, 1 = underflow/overflow latched checking, 2 = underflow/overflow unlatched checking).
rst_mode	0 to 3 Default: 1	Reset mode 0 = asynchronous reset including memory, 1 = synchronous reset including memory, 2 = asynchronous reset excluding memory, 3 = synchronous reset excluding memory).
byte_order	0 or 1 Default: 0	Order of send/receive bytes or subword [subword < 8 bits > subword] within a word 0 = first byte is in most significant bits position; 1 = first byte is in the least significant bits position [valid for $\text{data\_in\_width} \neq \text{data\_out\_width}$ ].

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cl1	Partial carry look-ahead model	DesignWare
cl2	Full carry look-ahead 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*.

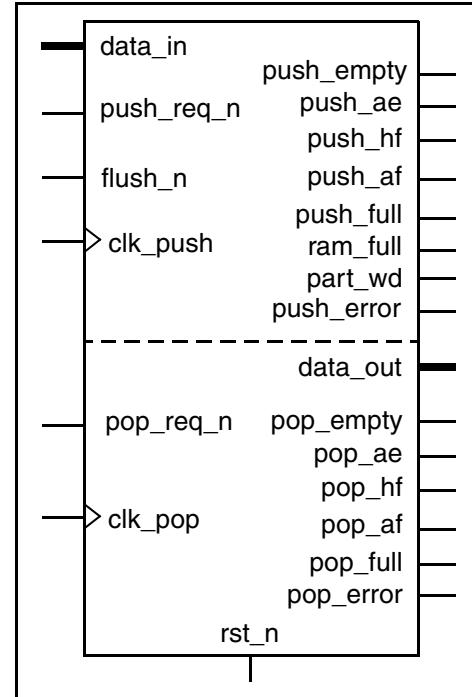
**DW\_asymfifo\_s2\_sf**

Asymmetric Synchronous (Dual Clock) FIFO with Static Flags

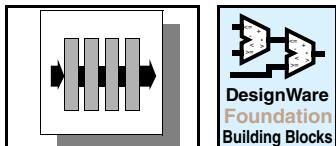
**DW\_asymfifo\_s2\_sf**

Asymmetric Synchronous (Dual Clock) FIFO with Static Flags

- ❖ Parameterized asymmetric input and output bit widths (must be integer-multiple relationship)
- ❖ Fully registered synchronous flag output ports
- ❖ Separate status flags for each clock system
- ❖ FIFO empty, half full, and full flags
- ❖ Parameterized almost full and almost empty flags
- ❖ FIFO push error (overflow) and pop error (underflow) flags
- ❖ D flip-flop-based memory array for high testability
- ❖ Single clock cycle push and pop operations
- ❖ Word integrity flag for  $data\_in\_width < data\_out\_width$
- ❖ Partial word flush for  $data\_in\_width < data\_out\_width$
- ❖ Parameterized byte order within a word
- ❖ Parameterized reset mode (synchronous or asynchronous, memory array initialized or not)

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk_push	1 bit	Input	Input clock for push interface
clk_pop	1 bit	Input	Input clock for pop interface
rst_n	1 bit	Input	Reset input, active low
push_req_n	1 bit	Input	FIFO push request, active low
flush_n	1 bit	Input	Flushes the partial word into memory (fills in 0's for empty bits) (for $data\_in\_width < data\_out\_width$ only), active low
pop_req_n	1 bit	Input	FIFO pop request, active low
data_in	$data\_in\_width$ bit(s)	Input	FIFO data to push
push_empty	1 bit	Output	FIFO empty <sup>a</sup> output flag synchronous to clk_push, active high
push_ae	1 bit	Output	FIFO almost empty <sup>a</sup> output flag synchronous to clk_push (determined by push_ae_lvl parameter), active high
push_hf	1 bit	Output	FIFO half full <sup>a</sup> output flag synchronous to clk_push, active high
push_af	1 bit	Output	FIFO almost full <sup>a</sup> output flag synchronous to clk_push (determined by push_af_lvl parameter), active high

**Table 1-1 Pin Description (Continued)**

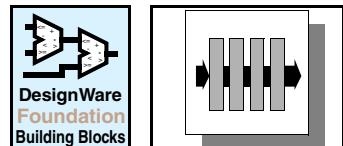
Pin Name	Width	Direction	Function
push_full	1 bit	Output	FIFO's RAM full <sup>a</sup> output flag (including the input buffer of FIFO for $data\_in\_width < data\_out\_width$ ) synchronous to clk_push, active high
ram_full	1 bit	Output	FIFO's RAM (excluding the input buffer of FIFO for $data\_in\_width < data\_out\_width$ ) full output flag synchronous to clk_push, active high
part_wd	1 bit	Output	Partial word accumulated in the input buffer synchronous to clk_push (for $data\_in\_width < data\_out\_width$ only; otherwise, tied low), active high
push_error	1 bit	Output	FIFO push error (overrun) output flag synchronous to clk_push, active high
pop_empty	1 bit	Output	FIFO empty <sup>b</sup> output flag synchronous to clk_pop, active high
pop_ae	1 bit	Output	FIFO almost empty <sup>b</sup> output flag synchronous to clk_pop (determined by pop_ae_lvl parameter), active high
pop_hf	1 bit	Output	FIFO half full <sup>b</sup> output flag synchronous to clk_pop, active high
pop_af	1 bit	Output	FIFO almost full <sup>b</sup> output flag synchronous to clk_pop (determined by pop_af_lvl parameter), active high
pop_full	1 bit	Output	FIFO's RAM full <sup>b</sup> output flag (excluding the input buffer of FIFO for case $data\_in\_width < data\_out\_width$ ) synchronous to clk_pop, active high
pop_error	1 bit	Output	FIFO pop error (underrun) output flag synchronous to clk_pop, active high
data_out	$data\_out\_width$ bit(s)	Output	FIFO data to pop

a. As perceived by the push interface.

b. As perceived by the pop interface.

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_in_width	1 to 256	Width of the data_in bus. data_in_width must be an integer-multiple of data_out_width. That is, either $data\_in\_width = K \times data\_out\_width$ , or $data\_out\_width = K \times data\_in\_width$ , where K is an integer.
data_out_width	1 to 256	Width of the data_out bus. data_out_width must be an integer-multiple of data_in_width. That is, either $data\_in\_width = K \times data\_out\_width$ , or $data\_out\_width = K \times data\_in\_width$ , where K is an integer.
depth	4 to 256	Number of words that can be stored in FIFO
push_ae_lvl	1 to depth-1	Almost empty level for the push_ae output port (the number of words in the FIFO at or below which the push_ae flag is active).

**DW\_asymfifo\_s2\_sf**

Asymmetric Synchronous (Dual Clock) FIFO with Static Flags

**Table 1-2 Parameter Description (Continued)**

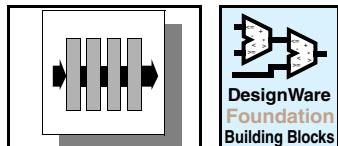
Parameter	Values	Description
push_af_lvl	1 to <i>depth</i> -1	Almost full level for the push_af output port (the number of empty memory locations in the FIFO at which the push_af flag is active.)
pop_ae_lvl	1 to <i>depth</i> -1	Almost empty level for the pop_ae output port (the number of words in the FIFO at or below which the pop_ae flag is active)
pop_af_lvl	1 to <i>depth</i> -1	Almost full level for the pop_af output port (the number of empty memory locations in the FIFO at which the pop_af flag is active.)
err_mode	0 or 1	Error mode 0 = stays active until reset [latched], 1 = active only as long as error condition exists [unlatched]
push_sync	1 to 3	Push flag synchronization mode 1 = single register synchronization from pop pointer, 2 = double register, 3 = triple register
pop_sync	1 to 3	Pop flag synchronization mode 1 = single register synchronization from push pointer, 2 = double register, 3 = triple register)
rst_mode	0 or 3 Default: 1	Reset mode 0 = asynchronous reset including memory, 1 = synchronous reset including memory, 2 = asynchronous reset excluding memory, 3 = synchronous reset excluding memory).
byte_order	0 or 1 Default: 0	Order of bytes or subword within a word 0 = first byte is in most significant bits position; 1 = first byte is in the least significant bits position.

a. Valid depth values include binary numbers from 8 to 256 (i.e. 8, 16, 32, 64, etc.) and all odd values between 8 and 256.

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cl2	Full carry look-ahead model	DesignWare

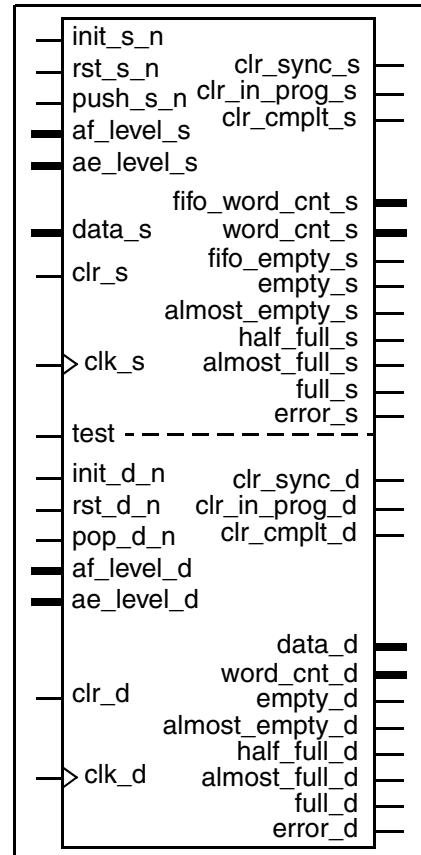
a. During synthesis, Design Compiler will select the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table. For more details, please refer to the *DesignWare Building Block IP User Guide*



## DW\_fifo\_2c\_df

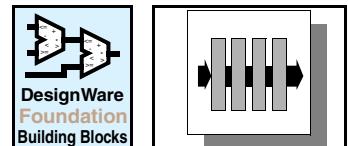
Dual Independent Clock FIFO

- ❖ Pop interface caching (pre-fetching)
- ❖ Alternative pop cache implementations provided for optimum power savings
- ❖ Configurable pipelining of push and pop control/data to accommodate synchronous RAMs
- ❖ Single clock cycle push and pop operations
- ❖ Fully registered synchronous status flag outputs
- ❖ Status flags provided from each clock domain
- ❖ Parameterized data width
- ❖ Parameterized RAM depth
- ❖ Parameterized number of synchronization stages in each clock domain
- ❖ Parameterized full-related and empty-related flag thresholds per clock domain
- ❖ Push error (overflow) and pop error (underflow) flags per clock domain
- ❖ Provides minPower benefits with the DesignWare-LP license.



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk_s	1 bit	Input	Source domain clock
rst_s_n	1 bit	Input	Source domain asynchronous reset (active low)
init_s_n	1 bit	Input	Source domain synchronous reset (active low)
clr_s	1 bit	Input	Source domain clear RAM contents
ae_level_s	$\text{ceil}(\log_2[\text{ram\_depth}]+1)$	Input	Source domain almost empty level for the almost_empty_s output (the number of words in the RAM at or below which the almost_empty_s flag is active) (see Note on eff_depth below)
af_level_s	$\text{ceil}(\log_2[\text{ram\_depth}]+1)$	Input	Source domain almost full level for the almost_full_s output (the number of empty memory locations in the RAM at which the almost_full_s flag is active)
push_s_n	1 bit	Input	Source domain push request (active low)
data_s	width	Input	Source domain push data

**DW\_fifo\_2c\_df**

Dual Independent Clock FIFO

**Table 1-1 Pin Description (Continued)**

Pin Name	Width	Direction	Function
clr_sync_s	1 bit	Output	Source domain coordinated clear synchronized (reset pulse that goes source sequential logic)
clr_in_prog_s	1 bit	Output	Source domain clear in progress
clr_cmplt_s	1 bit	Output	Source domain clear complete (single <code>clk_s</code> cycle pulse)
fifo_word_cnt_s	$\text{ceil}(\log_2(\text{eff\_depth}+1))$	Output	Source domain total word count in the RAM and cache (see Note on <code>eff_depth</code> parameter below)
word_cnt_s	$\text{ceil}(\log_2(\text{ram\_depth}+1))$	Output	Source domain RAM word count (see Note on <code>ram_depth</code> parameter below)
fifo_empty_s	1 bit	Output	Source domain FIFO empty flag
empty_s	1 bit	Output	Source domain RAM empty flag
almost_empty_s	1 bit	Output	Source domain RAM almost empty flag (determined by <code>ae_level_s</code> input)
half_full_s	1 bit	Output	Source domain RAM half full flag
almost_full_s	1 bit	Output	Source domain RAM almost full flag (determined by <code>af_level_s</code> input)
full_s	1 bit	Output	Source domain RAM full flag
error_s	1 bit	Output	Source domain push error flag (overrun)
clk_d	1 bit	Input	Destination domain clock
rst_d_n	1 bit	Input	Destination domain asynchronous reset (active low)
init_d_n	1 bit	Input	Destination domain synchronous reset (active low)
clr_d	1 bit	Input	Destination domain clear RAM contents
ae_level_d	$\text{ceil}(\log_2(\text{ram\_depth})+1)$	Input	Destination domain almost empty level for the <code>almost_empty_d</code> output (the number of words in the FIFO at or below which the <code>almost_empty_d</code> flag is active) (see Note on <code>eff_depth</code> below)
af_level_d	$\text{ceil}(\log_2(\text{ram\_depth})+1)$	Input	Destination domain almost full level for the <code>almost_full_d</code> output (the number of empty memory locations in the FIFO at which the <code>almost_full_d</code> flag is active)
pop_d_n	1 bit	Input	Destination domain pop request (active low)
clr_sync_d	1 bit	Output	Destination domain coordinated clear synchronized (reset pulse that goes to destination sequential logic)
clr_in_prog_d	1 bit	Output	Destination initiated clear in progress
clr_cmplt_d	1 bit	Output	Destination domain clear complete (single <code>clk_d</code> cycle pulse)
data_d	width	Output	Destination domain data to pop

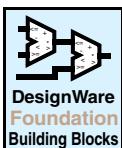
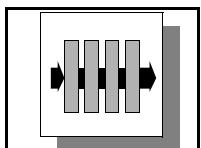
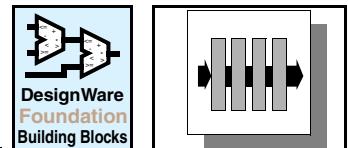


Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
word_cnt_d	$\text{ceil}(\log_2(\text{eff\_depth}+1))$	Output	Destination domain total word count in the RAM and cache (see Note on <i>eff_depth</i> below)
empty_d	1 bit	Output	Destination domain empty flag
almost_empty_d	1 bit	Output	Destination domain FIFO almost empty flag (determined by ae_level_d input)
half_full_d	1 bit	Output	Destination domain FIFO half full flag
almost_full_d	1 bit	Output	Destination domain FIFO almost full flag
full_d	1 bit	Output	Destination domain FIFO full flag (determined by af_level_d input)
error_d	1 bit	Output	Destination domain error flag (underrun)
test	<i>width</i> bit(s)	Input	Scan test mode select

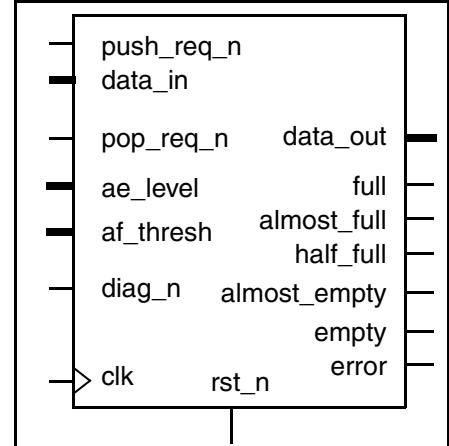
**DW\_fifo\_s1\_df**

Synchronous (Single Clock) FIFO with Dynamic Flags

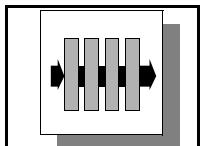
**DW\_fifo\_s1\_df**

Synchronous (Single Clock) FIFO with Dynamic Flags

- ❖ Fully registered synchronous flag output ports
- ❖ D flip-flop-based memory array for high testability
- ❖ All operations execute in a single clock cycle
- ❖ FIFO empty, half full, and full flags
- ❖ FIFO error flag indicating underflow, overflow, and pointer corruption
- ❖ Dynamically programmable almost full and almost empty flags
- ❖ Parameterized word width
- ❖ Parameterized word depth
- ❖ Parameterized reset mode (synchronous or asynchronous, memory array initialized or not)

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
<code>clk</code>	1 bit	Input	Input clock
<code>rst_n</code>	1 bit	Input	Reset input, active low (asynchronous if <code>rst_mode</code> = 0 or 2, synchronous if <code>rst_mode</code> = 1 or 3)
<code>push_req_n</code>	1 bit	Input	FIFO push request, active low
<code>pop_req_n</code>	1 bit	Input	FIFO pop request, active low
<code>diag_n</code>	1 bit	Input	Diagnostic control, active low
<code>ae_level</code>	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Almost empty level (the number of words in the FIFO at or below which the <code>almost_empty</code> flag is active)
<code>af_thresh</code>	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Almost full threshold (the number of words stored in the FIFO at or above which the <code>almost_full</code> flag is active)
<code>data_in</code>	<i>width</i> bit(s)	Input	FIFO data to push
<code>empty</code>	1 bit	Output	FIFO empty output, active high
<code>almost_empty</code>	1 bit	Output	FIFO almost empty output, active high
<code>half_full</code>	1 bit	Output	FIFO half full output, active high
<code>almost_full</code>	1 bit	Output	FIFO almost full output, active high
<code>full</code>	1 bit	Output	FIFO full output, active high
<code>error</code>	1 bit	Output	FIFO error output, active high
<code>data_out</code>	<i>width</i> bit(s)	Output	FIFO data to pop

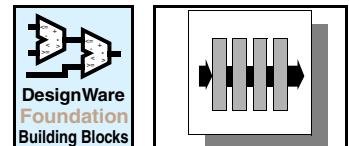
**DW\_fifo\_s1\_df**  
Synchronous (Single Clock) FIFO with Dynamic Flags**Table 1-2 Parameter Description**

Parameter	Values	Description
width	1 to 2048, Default: 8	Width of data_in and data_out buses
depth	2 to 1024, Default: 4	Number of memory elements used in FIFO
err_mode	0 to 2 Default: 0	Error mode 0 = underflow/overflow and pointer latched checking, 1 = underflow/overflow latched checking, 2 = underflow/overflow unlatched checking
rst_mode	0 to 3 Default: 0	Reset mode 0 = asynchronous reset including memory, 1 = synchronous reset including memory, 2 = asynchronous reset excluding memory, 3 = synchronous reset excluding memory

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl <sup>a</sup>	Synthesis Model	DesignWare

- a. The implementation “rtl” replaces the obsolete implementations “rpl,” “cl1,” and “cl2.” Information messages listing implementation replacements (SYNDB-37) may be generated by DC at compile time. Existing designs that specify an obsolete implementation (“rpl,” “cl1,” and “cl2”) will automatically have that implementation replaced by the new superseding implementation (“rtl”) noted by an information message (SYNDB-36) generated during DC compilation. The new implementation is capable of producing any of the original architectures automatically based on user constraints.

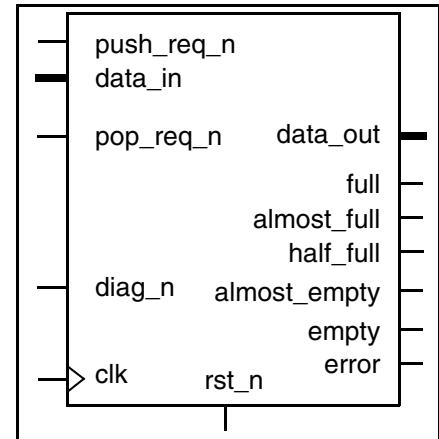
**DW\_fifo\_s1\_sf**

Synchronous (Single Clock) FIFO with Static Flags

**DW\_fifo\_s1\_sf**

Synchronous (Single Clock) FIFO with Static Flags

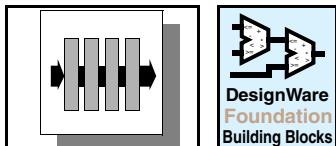
- ❖ Fully registered synchronous flag output ports
- ❖ D flip-flop-based memory array for high testability
- ❖ All operations execute in a single clock cycle
- ❖ FIFO empty, half full, and full flags
- ❖ FIFO error flag indicating underflow, overflow, and pointer corruption
- ❖ Parameterized word width
- ❖ Parameterized word depth
- ❖ Parameterized almost full and almost empty flags
- ❖ Parameterized reset mode (synchronous or asynchronous, memory array initialized or not)

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset input, active low asynchronous if <code>rst_mode = 0 or 2</code> , synchronous if <code>rst_mode = 1 or 3</code>
push_req_n	1 bit	Input	FIFO push request, active low
pop_req_n	1 bit	Input	FIFO pop request, active low
diag_n	1 bit	Input	Diagnostic control, active low
data_in	<i>width</i> bit(s)	Input	FIFO data to push
empty	1 bit	Output	FIFO empty output, active high
almost_empty	1 bit	Output	FIFO almost empty output, active high
half_full	1 bit	Output	FIFO half full output, active high
almost_full	1 bit	Output	FIFO almost full output, active high
full	1 bit	Output	FIFO full output, active high
error	1 bit	Output	FIFO error output, active high
data_out	<i>width</i> bit(s)	Output	FIFO data to pop

**Table 1-2 Parameter Description**

Parameter	Values	Function
width	1 to 2048 Default: 8	Width of the data_in and data_out buses



**DW\_fifo\_s1\_sf**  
Synchronous (Single Clock) FIFO with Static Flags

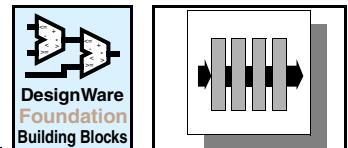
**Table 1-2 Parameter Description**

Parameter	Values	Function
depth	2 to 1024 Default: 4	Number of memory elements used in FIFO ( $\text{addr\_width} = \text{ceil}(\log_2(\text{depth}))$ )
ae_level	1 to $\text{depth} - 1$ Default: 1	Almost empty level (the number of words in the FIFO at or below which the almost_empty flag is active)
af_level	1 to $\text{depth} - 1$ Default: 1	Almost full level (the number of empty memory locations in the FIFO at which the almost_full flag is active.)
err_mode	0 to 2 Default: 0	Error mode 0 = underflow/overflow and pointer latched checking, 1 = underflow/overflow latched checking, 2 = underflow/overflow unlatched checking
rst_mode	0 to 3 Default: 0	Reset mode 0 = asynchronous reset including memory, 1 = synchronous reset including memory, 2 = asynchronous reset excluding memory, 3 = synchronous reset excluding memory

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl <sup>a</sup>	Synthesis Model	DesignWare

- a. The implementation “rtl” replaces the obsolete implementations “rpl,” “cl1,” and “cl2.” Information messages listing implementation replacements (SYNDB-37) may be generated by DC at compile time. Existing designs that specify an obsolete implementation (“rpl,” “cl1,” and “cl2”) will automatically have that implementation replaced by the new superseding implementation (“rtl”) noted by an information message (SYNDB-36) generated during DC compilation. The new implementation is capable of producing any of the original architectures automatically based on user constraints.

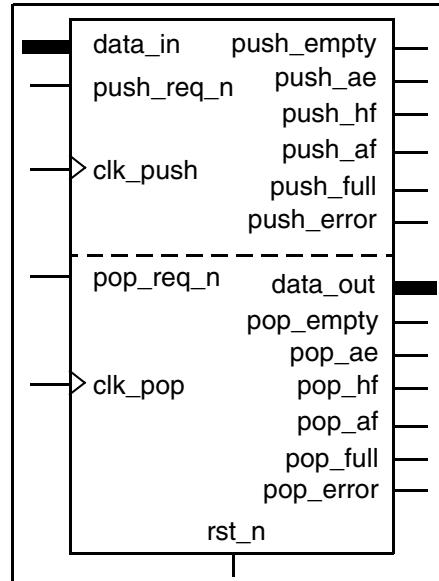
**DW\_fifo\_s2\_sf**

Synchronous (Dual-Clock) FIFO with Static Flags

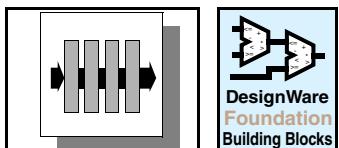
**DW\_fifo\_s2\_sf**

Synchronous (Dual-Clock) FIFO with Static Flags

- ❖ Fully registered synchronous flag output ports
- ❖ Single clock cycle push and pop operations
- ❖ Parameterized word width
- ❖ Parameterized word depth
- ❖ Separate status flags for each clock system
- ❖ FIFO empty, half full, and full flags
- ❖ Parameterized almost full and almost empty flag thresholds
- ❖ FIFO push error (overflow) and pop error (underflow) flags

**Table 1-1 Pin Description**

<b>Pin Name</b>	<b>Width</b>	<b>Direction</b>	<b>Function</b>
clk_push	1 bit	Input	Input clock for push interface
clk_pop	1 bit	Input	Input clock for pop interface
rst_n	1 bit	Input	Reset input, active low
push_req_n	1 bit	Input	FIFO push request, active low
pop_req_n	1 bit	Input	FIFO pop request, active low
data_in	<i>width</i> bit(s)	Input	FIFO data to push
push_empty	1 bit	Output	FIFO empty <sup>a</sup> output flag synchronous to clk_push, active high
push_ae	1 bit	Output	FIFO almost empty <sup>a</sup> output flag synchronous to clk_push, active high (determined by push_ae_lvl parameter)
push_hf	1 bit	Output	FIFO half full <sup>a</sup> output flag synchronous to clk_push, active high
push_af	1 bit	Output	FIFO almost full <sup>a</sup> output flag synchronous to clk_push, active high (determined by push_af_lvl parameter)
push_full	1 bit	Output	FIFO full <sup>a</sup> output flag synchronous to clk_push, active high
push_error	1 bit	Output	FIFO push error (overrun) output flag synchronous to clk_push, active high
pop_empty	1 bit	Output	FIFO empty <sup>b</sup> output flag synchronous to clk_pop, active high
pop_ae	1 bit	Output	FIFO almost empty <sup>b</sup> output flag synchronous to clk_pop, active high (determined by pop_ae_lvl parameter)
pop_hf	1 bit	Output	FIFO half full <sup>b</sup> output flag synchronous to clk_pop, active high

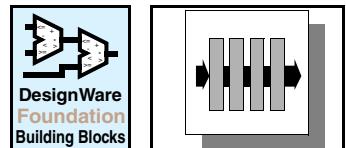

**DW\_fifo\_s2\_sf**  
Synchronous (Dual-Clock) FIFO with Static Flags
**Table 1-1 Pin Description (Continued)**

Pin Name	Width	Direction	Function
pop_af	1 bit	Output	FIFO almost full <sup>b</sup> output flag synchronous to clk_pop, active high (determined by pop_af_lvl parameter)
pop_full	1 bit	Output	FIFO full <sup>b</sup> output flag synchronous to clk_pop, active high
pop_error	1 bit	Output	FIFO pop error (underrun) output flag synchronous to clk_pop, active high
data_out	<i>width</i> bit(s)	Output	FIFO data to pop

- a. As perceived by the push interface.  
b. As perceived by the pop interface.

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	1 to 256 Default: 8	Width of the data_in and data_out buses
depth	4 to 256 Default: 8	Number of words that can be stored in FIFO
push_ae_lvl	1 to <i>depth</i> -1 Default: 2	Almost empty level for the push_ae output port (the number of words in the FIFO at or below which the push_ae flag is active)
push_af_lvl	1 to <i>depth</i> -1 Default: 2	Almost full level for the push_af output port (the number of empty memory locations in the FIFO at which the push_af flag is active.)
pop_ae_lvl	1 to <i>depth</i> -1 Default: 2	Almost empty level for the pop_ae output port (the number of words in the FIFO at or below which the pop_ae flag is active)
pop_af_lvl	1 to <i>depth</i> -1 Default: 2	Almost full level for the pop_af output port (the number of empty memory locations in the FIFO at which the pop_af flag is active.)
err_mode	0 or 1 Default: 0	Error mode 0 = stays active until reset [latched], 1 = active only as long as error condition exists [unlatched]
push_sync	1 to 3 Default: 2	Push flag synchronization mode 1 = single register synchronization from pop pointer, 2 = double register, 3 = triple register
pop_sync	1 to 3 Default: 2	Pop flag synchronization mode 1 = single register synchronization from push pointer, 2 = double register, 3 = triple register)
rst_mode	0 to 3 Default: 0	Reset mode 0 = asynchronous reset including memory, 1 = synchronous reset including memory, 2 = asynchronous reset excluding memory, 3 = synchronous reset excluding memory

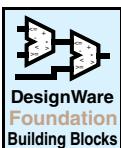
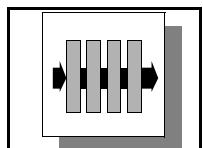
**DW\_fifo\_s2\_sf**

Synchronous (Dual-Clock) FIFO with Static Flags

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cl2	Full carry look-ahead 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*



## DW\_asymfifoctl\_s1\_df

Asymmetric Synchronous (1-Clock) FIFO Controller with Dynamic Flags

- ❖ Fully registered synchronous address and flag output ports
- ❖ All operations execute in a single clock cycle
- ❖ FIFO empty, half full, and full flags
- ❖ Asymmetric input and output bit widths (must be integer-multiple relationship)
- ❖ Word integrity flag for  $data\_in\_width < data\_out\_width$
- ❖ Flushing out partial word for  $data\_in\_width < data\_out\_width$
- ❖ Parameterized byte order within a word
- ❖ FIFO error flag indicating underflow, overflow, and pointer corruption
- ❖ Parameterized word depth
- ❖ Dynamically programmable almost full and almost empty flags
- ❖ Parameterized reset mode (synchronous or asynchronous)
- ❖ Interfaces to common hard macro or compiled ASIC dual-port synchronous RAMs
- ❖ Provides minPower benefits with the DesignWare-LP license.

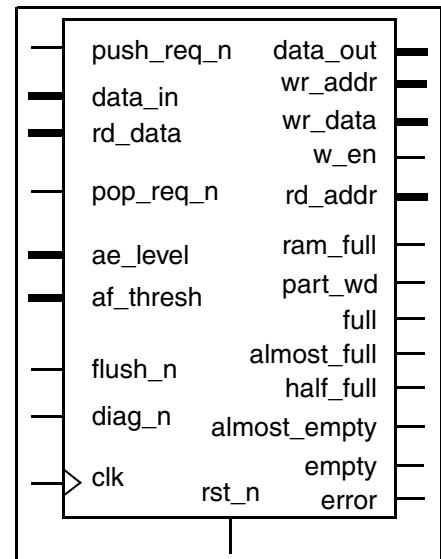
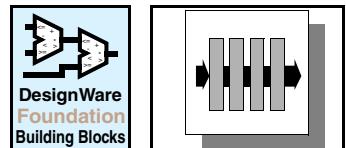


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset input, active low asynchronous if $rst\_mode=0$ , synchronous if $rst\_mode=1$ )
push_req_n	1 bit	Input	FIFO push request, active low
flush_n	1 bit	Input	Flushes the partial word into memory (fills in 0's) (for $data\_in\_width < data\_out\_width$ only)
pop_req_n	1 bit	Input	FIFO pop request, active low
diag_n	1 bit	Input	Diagnostic control, active low (for $err\_mode=0$ , NC for other $err\_mode$ values)
data_in	$data\_in\_width$ bit(s)	Input	FIFO data to push
rd_data	max ( $data\_in\_width$ , $data\_out\_width$ ) bit(s)	Input	RAM data input to FIFO controller
ae_level	$\text{ceil}(\log_2[depth])$ bit(s)	Input	Almost empty level (the number of words in the FIFO at or below which the almost_empty flag is active)
af_thresh	$\text{ceil}(\log_2[depth])$ bit(s)	Input	Almost full threshold (the number of words stored in the FIFO at or above which the almost_full flag is active)

**DW\_asymfifoctl\_s1\_df**

Asymmetric Synchronous (1-Clock) FIFO Controller with Dynamic Flags

**Table 1-1 Pin Description (Continued)**

Pin Name	Width	Direction	Function
w_en	1 bit	Output	Write enable output for write port of RAM, active low
empty	1 bit	Output	FIFO empty output, active high
almost_empty	1 bit	Output	FIFO almost empty output, active high, asserted when FIFO level $\leq ae\_level$
half_full	1 bit	Output	FIFO half full output, active high
almost_full	1 bit	Output	FIFO almost full output, active high, asserted when FIFO level $\geq af\_thresh$
full	1 bit	Output	FIFO full output, active high
ram_full	1 bit	Output	RAM full output, active high
error	1 bit	Output	FIFO error output, active high
part_wd	1 bit	Output	Partial word, active high (for $data\_in\_width < data\_out\_width$ only; otherwise, tied low)
wr_data	max ( $data\_in\_width$ , $data\_out\_width$ ) bit(s)	Output	FIFO controller output data to RAM
wr_addr	ceil( $\log_2[depth]$ ) bit(s)	Output	Address output to write port of RAM
rd_addr	ceil( $\log_2[depth]$ ) bit(s)	Output	Address output to read port of RAM
data_out	$data\_out\_width$ bit(s)	Output	FIFO data to pop

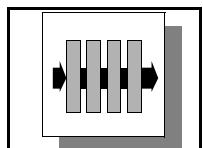


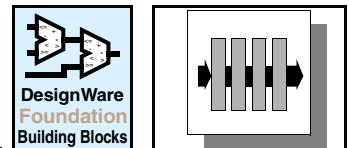
Table 1-2 Parameter Description

Parameter	Values	Description
data_in_width	1 to 256	Width of the data_in bus. data_in_width must be in an integer-multiple relationship with data_out_width. That is, either $data\_in\_width = K \times data\_out\_width$ , or $data\_out\_width = K \times data\_in\_width$ , where $K$ is an integer.
data_out_width	1 to 256	Width of the data_out bus. data_out_width must be in an integer-multiple relationship with data_in_width. That is, either $data\_in\_width = K \times data\_out\_width$ , or $data\_out\_width = K \times data\_in\_width$ , where $K$ is an integer.
depth	2 to $2^{24}$	Number of memory elements used in the FIFO ( $addr\_width = \text{ceil}[\log_2(depth)]$ )
err_mode	0 to 2 Default: 1	Error mode 0 = underflow/overflow with pointer latched checking, 1 = underflow/overflow latched checking, 2 = underflow/overflow unlatched checking).
rst_mode	0 or 1 Default: 1	Reset mode 0 = asynchronous reset, 1 = synchronous reset).
byte_order	0 or 1 Default: 0	Order of bytes or subword [subword < 8 bits > subword] within a word 0 = first byte is in most significant bits position; 1 = first byte is in the least significant bits position.

Table 1-3 Synthesis Implementations<sup>a</sup>

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cl1	Partial carry look-ahead model	DesignWare
cl2	Full carry look-ahead 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*

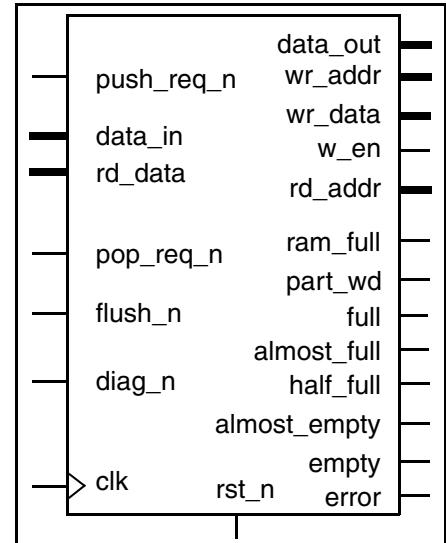
**DW\_asymfifoctl\_s1\_sf**

Asymmetric Synchronous (1-Clock) FIFO Controller with Static Flags

**DW\_asymfifoctl\_s1\_sf**

Asymmetric Synchronous (1-Clock) FIFO Controller with Static Flags

- ❖ Fully registered synchronous address and flag output ports
- ❖ All operations execute in a single clock cycle
- ❖ FIFO empty, half full, and full flags
- ❖ Asymmetric input and output bit widths (must be integer-multiple relationship)
- ❖ Word integrity flag for  $data\_in\_width < data\_out\_width$
- ❖ Flushing out partial word for  $data\_in\_width < data\_out\_width$
- ❖ Parameterized byte order within a word
- ❖ FIFO error flag indicating underflow, overflow, and pointer corruption
- ❖ Parameterized word depth
- ❖ Parameterized almost full and almost empty flags
- ❖ Parameterized reset mode (synchronous or asynchronous)
- ❖ Interfaces to common hard macro or compiled ASIC dual-port synchronous RAMs
- ❖ Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset input, active low asynchronous if $rst\_mode = 0$ , synchronous if $rst\_mode = 1$ )
push_req_n	1 bit	Input	FIFO push request, active low
flush_n	1 bit	Input	Flushes the partial word into memory (fills in 0's) (for $data\_in\_width < data\_out\_width$ only)
pop_req_n	1 bit	Input	FIFO pop request, active low
diag_n	1 bit	Input	Diagnostic control, active low (for $err\_mode = 0$ , NC for other $err\_mode$ values)
data_in	$data\_in\_width$ bit(s)	Input	FIFO data to push
rd_data	max ( $data\_in\_width$ , $data\_out\_width$ ) bit(s)	Input	RAM data input to FIFO controller
w_en	1 bit	Output	Write enable output for write port of RAM, active low
empty	1 bit	Output	FIFO empty output, active high
almost_empty	1 bit	Output	FIFO almost empty output, active high, asserted when FIFO level $\leq ae\_level$

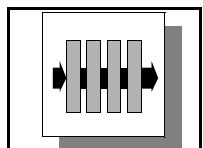
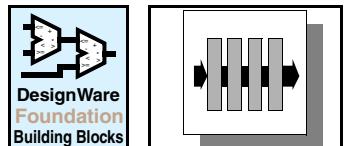


Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
half_full	1 bit	Output	FIFO half full output, active high
almost_full	1 bit	Output	FIFO almost full output, active high, asserted when FIFO level $\geq (depth - af\_level)$
full	1 bit	Output	FIFO full output, active high
ram_full	1 bit	Output	RAM full output, active high
error	1 bit	Output	FIFO error output, active high
part_wd	1 bit	Output	Partial word, active high (for $data\_in\_width < data\_out\_width$ only; otherwise, tied low)
wr_data	max ( $data\_in\_width$ , $data\_out\_width$ ) bit(s)	Output	FIFO controller output data to RAM
wr_addr	$\text{ceil}(\log_2[depth])$ bit(s)	Output	Address output to write port of RAM
rd_addr	$\text{ceil}(\log_2[depth])$ bit(s)	Output	Address output to read port of RAM
data_out	$data\_out\_width$ bit(s)	Output	FIFO data to pop

**DW\_asymfifoctl\_s1\_sf**

Asymmetric Synchronous (1-Clock) FIFO Controller with Static Flags

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_in_width	1 to 256	Width of the data_in bus. Values for data_in_width must be in an integer-multiple relationship with data_out_width. That is, either $data\_in\_width = K \times data\_out\_width$ , or $data\_out\_width = K \times data\_in\_width$ , where $K$ is an integer.
data_out_width	1 to 256	Width of the data_out bus. data_out_width must be in an integer-multiple relationship with data_in_width. That is, either $data\_in\_width = K \times data\_out\_width$ , or $data\_out\_width = K \times data\_in\_width$ , where $K$ is an integer.
depth	2 to $2^{24}$	Number of memory elements used in the FIFO ( $addr\_width = ceil[\log_2(depth)]$ )
ae_level	1 to $depth - 1$	Almost empty level (the number of words in the FIFO at or below which the almost_empty flag is active)
af_level	1 to $depth - 1$	Almost full level (the number of empty memory locations in the FIFO at which the almost_full flag is active).
err_mode	0 to 2 Default: 1	Error mode 0 = underflow/overflow with pointer latched checking, 1 = underflow/overflow latched checking, 2 = underflow/overflow unlatched checking).
rst_mode	0 or 1 Default: 1	Reset mode 0 = asynchronous reset, 1 = synchronous reset).
byte_order	0 or 1 Default: 0	Order of bytes or subword [subword < 8 bits > subword] within a word 0 = first byte is in most significant bits position; 1 = first byte is in the least significant bits position).

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cl1	Partial carry look-ahead model	DesignWare
cl2	Full carry look-ahead model	DesignWare

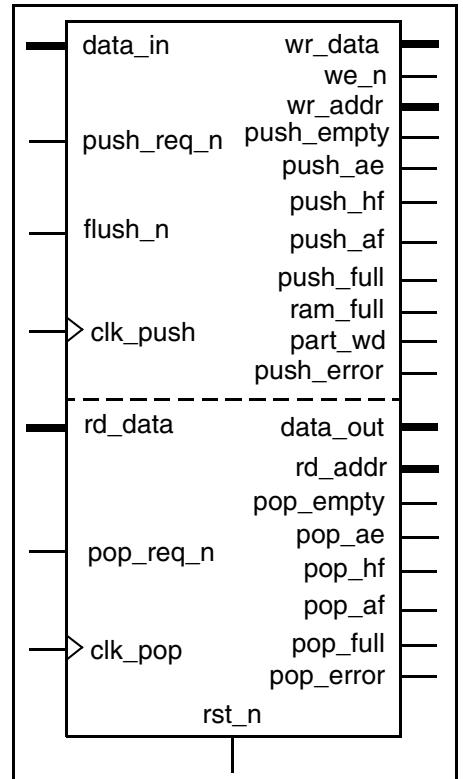
a. During synthesis, Design Compiler will select the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table. For more details, please refer to the *DesignWare Building Block IP User Guide*



## DW\_asymfifoctl\_s2\_sf

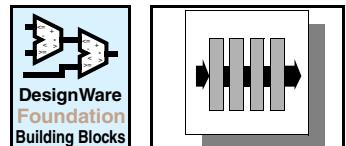
Asymmetric Synchronous (Dual-Clock) FIFO Controller with Static Flags

- ❖ Parameterized asymmetric input and output bit widths (must be integer-multiple relationship)
- ❖ Parameterized word depth
- ❖ Fully registered synchronous flag output ports
- ❖ Separate status flags for each clock domain
- ❖ FIFO empty, half full, and full flags
- ❖ Parameterized almost full and almost empty flags
- ❖ FIFO push error (overflow) and pop error (underflow) flags
- ❖ Single clock cycle push and pop operations
- ❖ Parameterized byte order within a word
- ❖ Word integrity flag for  $\text{data\_in\_width} < \text{data\_out\_width}$
- ❖ Partial word flush for  $\text{data\_in\_width} < \text{data\_out\_width}$
- ❖ Interfaces to common hard macro or compiled ASIC dual-port synchronous RAMs
- ❖ Provides minPower benefits with the DesignWare-LP license.



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk_push	1 bit	Input	Input clock for push interface
clk_pop	1 bit	Input	Input clock for pop interface
rst_n	1 bit	Input	Reset input, active low
push_req_n	1 bit	Input	FIFO push request, active low
flush_n	1 bit	Input	Flushes the partial word into memory (fills in 0's) (for $\text{data\_in\_width} < \text{data\_out\_width}$ only)
pop_req_n	1 bit	Input	FIFO pop request, active low
data_in	$\text{data\_in\_width}$ bit(s)	Input	FIFO data to push
rd_data	$\max(\text{data\_in\_width}, \text{data\_out\_width})$ bit(s)	Input	RAM data input to FIFO controller
we_n	1 bit	Output	Write enable output for write port of RAM, active low
push_empty	1 bit	Output	FIFO empty <sup>a</sup> output flag synchronous to clk_push, active high

**DW\_asymfifoctl\_s2\_sf**

Asymmetric Synchronous (Dual-Clock) FIFO Controller with Static Flags

**Table 1-1 Pin Description (Continued)**

Pin Name	Width	Direction	Function
push_ae	1 bit	Output	FIFO almost empty <sup>a</sup> output flag synchronous to <code>clk_push</code> , active high (determined by <code>push_ae_lv</code> parameter)
push_hf	1 bit	Output	FIFO half full <sup>a</sup> output flag synchronous to <code>clk_push</code> , active high
push_af	1 bit	Output	FIFO almost full <sup>a</sup> output flag synchronous to <code>clk_push</code> , active high (determined by <code>push_af_lv</code> parameter)
push_full	1 bit	Output	FIFO's RAM full <sup>a</sup> output flag (including the input buffer of FIFO controller for $data\_in\_width < data\_out\_width$ ) synchronous to <code>clk_push</code> , active high
ram_full	1 bit	Output	FIFO's RAM (excluding the input buffer of FIFO controller for $data\_in\_width < data\_out\_width$ ) full output flag synchronous to <code>clk_push</code> , active high
part_wd	1 bit	Output	Partial word accumulated in the input buffer synchronous to <code>clk_push</code> , active high (for $data\_in\_width < data\_out\_width$ only; otherwise, tied low)
push_error	1 bit	Output	FIFO push error (overrun) output flag synchronous to <code>clk_push</code> , active high
pop_empty	1 bit	Output	FIFO empty <sup>b</sup> output flag synchronous to <code>clk_pop</code> , active high
pop_ae	1 bit	Output	FIFO almost empty <sup>b</sup> output flag synchronous to <code>clk_pop</code> , active high (determined by <code>pop_ae_lv</code> parameter)
pop_hf	1 bit	Output	FIFO half full <sup>b</sup> output flag synchronous to <code>clk_pop</code> , active high
pop_af	1 bit	Output	FIFO almost full <sup>b</sup> output flag synchronous to <code>clk_pop</code> , active high (determined by <code>pop_af_lv</code> parameter)
pop_full	1 bit	Output	FIFO's RAM full <sup>b</sup> output flag (excluding the input buffer of FIFO controller for case $data\_in\_width < data\_out\_width$ ) synchronous to <code>clk_pop</code> , active high
pop_error	1 bit	Output	FIFO pop error (underrun) output flag synchronous to <code>clk_pop</code> , active high
wr_data	max ( $data\_in\_width$ , $data\_out\_width$ ) bit(s)	Output	FIFO controller output data to RAM
wr_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Output	Address output to write port of RAM
rd_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Output	Address output to read port of RAM
data_out	$data\_out\_width$ bit(s)	Output	FIFO data to pop

a. As perceived by the push interface.

b. As perceived by the pop interface.

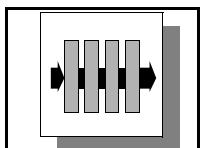
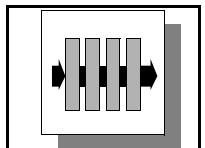
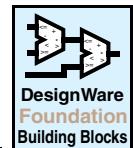


Table 1-2 Parameter Description

Parameter	Values	Description
data_in_width	1 to 4096 Default: 8	Width of the <code>data_in</code> bus. <code>data_in_width</code> must be in an integer-multiple relationship with <code>data_out_width</code> . That is, either <code>data_in_width = K * data_out_width</code> , or <code>data_out_width = K * data_in_width</code> , where K is an integer.
data_out_width	1 to 256 Default: 24	Width of the <code>data_out</code> bus. <code>data_out_width</code> must be in an integer-multiple relationship with <code>data_in_width</code> . That is, either <code>data_in_width = K * data_out_width</code> , or <code>data_out_width = K * data_in_width</code> , where K is an integer.
depth	4 to $2^{24}$ Default: 8	Number of words that can be stored in FIFO
push_ae_lvl	1 to <code>depth - 1</code> Default: 2	Almost empty level for the <code>push_ae</code> output port (the number of words in the FIFO at or below which the <code>push_ae</code> flag is active)
push_af_lvl	1 to <code>depth - 1</code> Default: 28	Almost full level for the <code>push_af</code> output port (the number of empty memory locations in the FIFO at which the <code>push_af</code> flag is active)
pop_ae_lvl	1 to <code>depth - 1</code> Default: 2	Almost empty level for the <code>pop_ae</code> output port (the number of words in the FIFO at or below which the <code>pop_ae</code> flag is active)
pop_af_lvl	1 to <code>depth - 1</code> Default: 2	Almost full level for the <code>pop_af</code> output port (the number of empty memory locations in the FIFO at which the <code>pop_af</code> flag is active)
err_mode	0 or 1 Default: 0	Error mode 0 = stays active until reset [latched], 1 = active only as long as error condition exists [unlatched])
push_sync	1 to 3 Default: 2	Push flag synchronization mode 1 = single register synchronization from pop pointer, 2 = double register, 3 = triple register)
pop_sync	1 to 3 Default: 2	Pop flag synchronization mode 1 = single register synchronization from push pointer, 2 = double register, 3 = triple register)
rst_mode	0 or 1 Default: 1	Reset mode 0 = asynchronous reset, 1 = synchronous reset)
byte_order	0 or 1 Default: 0	Order of bytes or subword within a word 0 = first byte is in most significant bits position; 1 = first byte is in the least significant bits position).

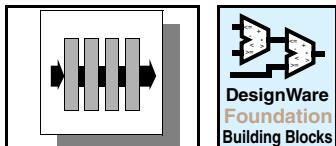
Table 1-3 Synthesis Implementations<sup>a</sup>

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cl2	Full carry look-ahead model	DesignWare

**DW\_asymfifoctl\_s2\_sf**

Asymmetric Synchronous (Dual-Clock) FIFO Controller with Static Flags

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*



## DW\_fifoctl\_2c\_df

### Dual Clock FIFO Controller with Dynamic Flags

- ❖ Pop interface caching (pre-fetching)
- ❖ Alternative pop cache implementations provided for optimum power savings
- ❖ Configurable pipelining of push and pop control/data to accommodate synchronous RAMs
- ❖ Single clock cycle push and pop operations
- ❖ Fully registered synchronous status flag outputs
- ❖ Status flags provided from each clock domain
- ❖ Parameterized data width
- ❖ Parameterized RAM depth
- ❖ Parameterized full-related and empty-related flag thresholds per clock domain
- ❖ Push error (overflow) and pop error (underflow) flags per clock domain
- ❖ Provides minPower benefits with the DesignWare-LP license.

### Description

DW\_fifoctl\_2c\_df is a dual independent clock FIFO controller intended to interface with dual-port synchronous RAM. Word caching (or pre-fetching) is performed in the pop interface to minimize latencies and allow for bursting of contiguous words. The caching depth is configurable.

init_s_n	clr_sync_s
rst_s_n	clr_in_prog_s
push_s_n	clr_cmplt_s
af_level_s	wr_en_s_n
ae_level_s	wr_addr_s
	fifo_word_cnt_s
	word_cnt_s
	fifo_empty_s
	empty_s
	almost_empty_s
clr_s	half_full_s
	almost_full_s
clk_s	full_s
	error_s
test	-
init_d_n	clr_sync_d
rst_d_n	clr_in_prog_d
pop_d_n	clr_cmplt_d
af_level_d	ram_re_d_n
ae_level_d	rd_addr_d
	ram_word_cnt_d
	word_cnt_d
rd_data_d	data_d
	empty_d
	almost_empty_d
clr_d	half_full_d
	almost_full_d
clk_d	full_d
	error_d

**Table 1-1 Pin Description**

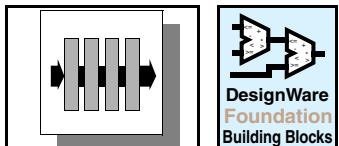
Pin Name	Width (bits)	Direction	Function
clk_s	1 bit	Input	Source domain clock
rst_s_n	1 bit	Input	Source domain asynchronous reset (active low)
init_s_n	1 bit	Input	Source domain synchronous reset (active low)
clr_s	1 bit	Input	Source domain clear RAM contents
ae_level_s	$\text{ceil}(\log_2[\text{ram\_depth}+1])$	Input	Source domain almost empty level for the <code>almost_empty_s</code> output (the number of words in the RAM at or below which the <code>almost_empty_s</code> flag is active) (see <code>eff_depth</code> note below)
af_level_s	$\text{ceil}(\log_2[\text{ram\_depth}+1])$	Input	Source domain almost full level for the <code>almost_full_s</code> output (the number of empty memory locations in the RAM at which the <code>almost_full_s</code> flag is active).

**DW\_fifoctl\_2c\_df**

Dual Clock FIFO Controller with Dynamic Flags

**Table 1-1 Pin Description (Continued)**

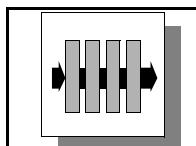
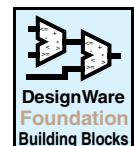
Pin Name	Width (bits)	Direction	Function
push_s_n	1 bit	Input	Source domain push request (active low)
clr_sync_s	1 bit	Output	Source domain coordinated clear synchronized (reset pulse that goes source sequential logic)
clr_in_prog_s	1 bit	Output	Source domain clear in progress
clr_cmplt_s	1 bit	Output	Source domain clear complete (single <code>clk_s</code> cycle pulse)
wr_en_s_n	1 bit	Output	Source domain write enable to RAM (active low and unregistered)
wr_addr_s	$\text{ceil}(\log_2[\text{ram\_depth}+1])$	Output	Source domain write address to RAM (registered)
fifo_word_cnt_s	$\text{ceil}(\log_2[\text{eff\_depth}+1])$	Output	Source domain total word count in the RAM and cache (see <i>eff_depth</i> note below)
word_cnt_s	$\text{ceil}(\log_2[\text{ram\_depth}+1])$	Output	Source domain RAM word count (see Note on <i>ram_depth</i> )
fifo_empty_s	1 bit	Output	Source domain FIFO empty flag
empty_s	1 bit	Output	Source domain RAM empty flag
almost_empty_s	1 bit	Output	Source domain RAM almost empty flag (determined by <code>ae_level_s</code> port)
half_full_s	1 bit	Output	Source domain RAM half full flag
almost_full_s	1 bit	Output	Source domain RAM full flag (determined by <code>af_level_s</code> port)
full_s	1 bit	Output	Source domain RAM almost full flag
error_s	1 bit	Output	Source domain push error flag (overrun)
clk_d	1 bit	Input	Destination domain clock
rst_d_n	1 bit	Input	Destination domain asynchronous reset (active low)
init_d_n	1 bit	Input	Destination domain synchronous reset (active low)
clr_d	1 bit	Input	Destination domain clear RAM contents
ae_level_d	$\text{ceil}(\log_2[\text{ram\_depth}+1])$	Input	Destination domain almost empty level for the <code>almost_empty_d</code> output (the number of words in the FIFO at or below which the <code>almost_empty_d</code> flag is active) (see <i>eff_depth</i> note below)
af_level_d	$\text{ceil}(\log_2[\text{ram\_depth}+1])$	Input	Destination domain almost full level for the <code>almost_full_d</code> output (the number of empty memory locations in the FIFO at which the <code>almost_full_d</code> flag is active).
pop_d_n	1 bit	Input	Destination domain pop request (active low)
rd_data_d	<i>width</i> bit(s)	Input	Destination domain read data
clr_sync_d	1 bit	Output	Destination domain coordinated clear synchronized (reset pulse that goes to source sequential logic)
clr_in_prog_d	1 bit	Output	Destination domain clear in progress
clr_cmplt_d	1 bit	Output	Destination domain clear complete (single <code>clk_d</code> cycle pulse)


**DW\_fifoctl\_2c\_df**  
 Dual Clock FIFO Controller with Dynamic Flags
**Table 1-1 Pin Description (Continued)**

Pin Name	Width (bits)	Direction	Function
ram_re_d_n	1 bit	Output	Destination domain read enable to RAM (active low)
rd_addr_d	$\text{ceil}(\log_2(\text{ram\_depth}))$	Output	Destination domain read address to RAM (registered)
data_d	<i>width</i> bit(s)	Output	Destination domain data to pop
word_cnt_d	$\text{ceil}(\log_2[\text{eff\_depth}+1])$	Output	Destination domain FIFO word count (see note on <i>ram_depth</i> parameter)
ram_word_cnt_d	$\text{ceil}(\log_2[\text{ram\_depth}+1])$	Output	Destination domain RAM word count (see note on <i>ram_depth</i> parameter below)
empty_d	1 bit	Output	Destination domain FIFO empty flag
almost_empty_d	1 bit	Output	Destination domain FIFO almost empty flag (determined by ae_level_d parameter)
half_full_d	1 bit	Output	Destination domain FIFO half full flag
almost_full_d	1 bit	Output	Destination domain FIFO almost full flag (determined by af_level_d port)
full_d	1 bit	Output	Destination domain FIFO full flag
error_d	1 bit	Output	Destination domain push error flag (overrun)
test	1 bit	Input	Scan test mode select

**Table 1-2 Parameters**

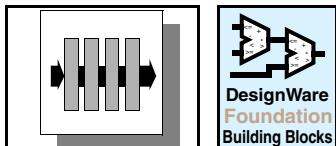
Parameter	Values	Function
width	1 to 1024 Default: 8	Vector width of input data_s and output data_d
ram_depth <sup>a</sup>	4 to 16777216 Default: 8	Desired number of FIFO locations to be operated out of RAM not including the cache.
mem_mode	0 or 7 Default: 3	Memory Control/Datapath Pipelining. Defines where and how many re-timing stages in RAM: 0 = no pre or post retiming 1 = RAM data out (post) re-timing 2 = RAM read address (pre) re-timing 3 = RAM data out and read address re-timing 4 = RAM write interface (pre) re-timing 5 = RAM write interface and RAM data out re-timing 6 = RAM write interface and read address re-timing 7 = RAM data out, write interface and read address re-timing

**DW\_fifoctl\_2c\_df**

Dual Clock FIFO Controller with Dynamic Flags

**Table 1-2 Parameters (Continued)**

Parameter	Values	Function
f_sync_type	0 to 4 Default: 2	Forward Synchronization Stages (direction from source to destination domains) 0 = no synchronizing stages 1 = 2-stage synchronization w/ 1st stage negative edge & 2nd stage positive edge capturing 2 = 2-stage synchronization w/ both stages positive edge capturing 3 = 3-stage synchronization w/ all stages positive edge capturing
r_sync_type	0 to 4 Default: 2	Return Synchronization Stages (direction from destination to source domains) 0 = no synchronizing stages 1 = 2-stage synchronization w/ 1st stage negative edge & 2nd stage positive edge capturing 2 = 2-stage synchronization w/ both stages positive edge capturing 3 = 3-stage synchronization w/ all stages positive edge capturing
clk_ratio	-7 to -1 or 1 to 7 Default: 1	Rounded quotient between clk_s and clk_d frequencies NOTE: 0 is illegal NOTE: This parameter is ignored when mem_mode is 0 or 1, and should be set to the default value. See " <a href="#">When Is It Necessary to Determine clk_ratio</a> " on page <a href="#">6</a> . 1 to 7 = when clk_d rate faster than clk_s rate: round(clk_d rate / clk_s rate) -7 to -1 = when clk_d rate slower than clk_s rate: 0 - round(clk_s rate / clk_d rate)
ram_re_ext	0 or 1 Default: 0	Extend ram_re_d_n during active read through RAM 0 = single-pulse of ram_re_d_n at read event of RAM 1 = extend assertion of ram_re_d_n while active read event traverses through RAM
err_mode	0 or 1 Default: 0	Error Reporting 0 = sticky error flag 1 = dynamic error flag
tst_mode	0 to 2 Default: 0	Test Mode 0 = no 'latch' is inserted for scan testing 1 = insert negative-edge capturing flip-flop on data_s input vector when test input is asserted 2 = insert hold latch using active low latch
verif_en	0 to 4 Default: 1	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 See the Simulation Model section for a more information.

**Table 1-2 Parameters (Continued)**

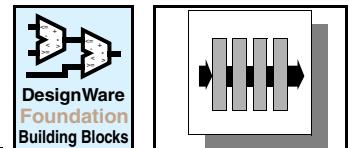
Parameter	Values	Function
clr_dual_domain	0 or 1 Default: 1	Activity of <code>clr_s</code> and/or <code>clr_d</code> 0 = either <code>clr_s</code> or <code>clr_d</code> can be activated, but the other must be tied 'low' 1 = both <code>clr_s</code> and <code>clr_d</code> can be activated
arch_type <sup>b</sup>	0 or 1 Default: 0	Pre-fetch cache architecture type: 0 = Pipeline style (PL cache) - 'rtl' implementation 1 = Register File style (RF cache) - 'lpwr' implementation

- a. Parameter `ram_depth` is not necessarily the number of RAM locations needed to operate the FIFO. See "Memory Depth Considerations and Setting `ram_depth`."
- b. Note: For `arch_type` equal to 1, the RF cache is used (`lpwr` implementation) when a Low-Power (DesignWare-LP) license is available and `mem_mode` is not 0 or 4. If a Low-Power license is not available or `mem_mode` is 0 or 4, the PL cache (`rtl` implementation) is always used not matter the setting of `arch_type`.

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Required
rtl	Synthesis model	DesignWare
lpwr <sup>a</sup>	Low power synthesis model	DesignWare-LP

- a. Note: The `lpwr` implementation is automatically selected when `arch_type` is 1 and `mem_mode` is not 0 nor 4 (with the DesignWare-LP license) unless overridden with "set implementation" of the `rtl` implementation. The `lpwr` implementation is synonymous with RF cache usage and the `rtl` implementation implies PL cache usage.

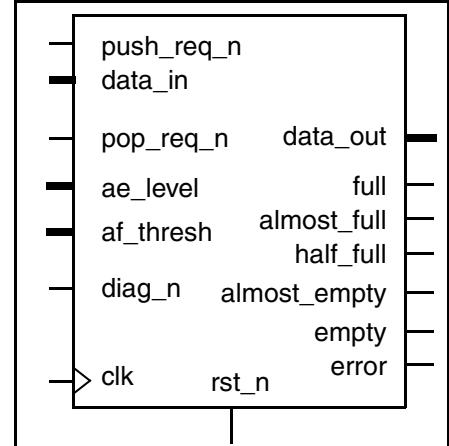
**DW\_fifoctl\_s1\_df**

Synchronous (Single Clock) FIFO Controller with Dynamic Flag

**DW\_fifoctl\_s1\_df**

Synchronous (Single Clock) FIFO Controller with Dynamic Flag

- ❖ Fully registered synchronous flag output ports
- ❖ D flip-flop-based memory array for high testability
- ❖ All operations execute in a single clock cycle
- ❖ FIFO empty, half full, and full flags
- ❖ FIFO error flag indicating underflow, overflow, and pointer corruption
- ❖ Dynamically programmable almost full and almost empty flags
- ❖ Parameterized word width
- ❖ Parameterized word depth
- ❖ Parameterized reset mode (synchronous or asynchronous, memory array initialized or not)

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset input, active low (asynchronous if rst_mode = 0 or 2, synchronous if rst_mode = 1 or 3)
push_req_n	1 bit	Input	FIFO push request, active low
pop_req_n	1 bit	Input	FIFO pop request, active low
diag_n	1 bit	Input	Diagnostic control, active low
ae_level	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Almost empty level (the number of words in the FIFO at or below which the almost_empty flag is active)
af_thresh	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Almost full threshold (the number of words stored in the FIFO at or above which the almost_full flag is active)
data_in	<i>width</i> bit(s)	Input	FIFO data to push
empty	1 bit	Output	FIFO empty output, active high
almost_empty	1 bit	Output	FIFO almost empty output, active high
half_full	1 bit	Output	FIFO half full output, active high
almost_full	1 bit	Output	FIFO almost full output, active high
full	1 bit	Output	FIFO full output, active high
error	1 bit	Output	FIFO error output, active high
data_out	<i>width</i> bit(s)	Output	FIFO data to pop

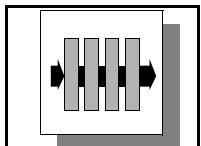


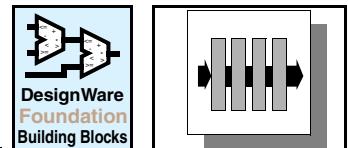
Table 1-2 Parameter Description

Parameter	Values	Description
width	1 to 2048, Default: 8	Width of data_in and data_out buses
depth	2 to 1024, Default: 4	Number of memory elements used in FIFO
err_mode	0 to 2 Default: 0	Error mode 0 = underflow/overflow and pointer latched checking, 1 = underflow/overflow latched checking, 2 = underflow/overflow unlatched checking
rst_mode	0 to 3 Default: 0	Reset mode 0 = asynchronous reset including memory, 1 = synchronous reset including memory, 2 = asynchronous reset excluding memory, 3 = synchronous reset excluding memory

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl <sup>a</sup>	Synthesis Model	DesignWare

- a. The implementation “rtl” replaces the obsolete implementations “rpl,” “cl1,” and “cl2.” Information messages listing implementation replacements (SYNDB-37) may be generated by DC at compile time. Existing designs that specify an obsolete implementation (“rpl,” “cl1,” and “cl2”) will automatically have that implementation replaced by the new superseding implementation (“rtl”) noted by an information message (SYNDB-36) generated during DC compilation. The new implementation is capable of producing any of the original architectures automatically based on user constraints.

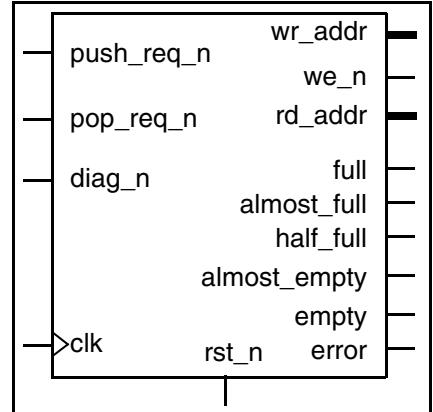
**DW\_fifoctl\_s1\_sf**

Synchronous (Single Clock) FIFO Controller with Static Flags

**DW\_fifoctl\_s1\_sf**

Synchronous (Single Clock) FIFO Controller with Static Flags

- ❖ Fully registered synchronous address and flag output ports
- ❖ All operations execute in a single clock cycle
- ❖ FIFO empty, half full, and full flags
- ❖ FIFO error flag indicating underflow, overflow, and pointer corruption
- ❖ Parameterized word depth
- ❖ Parameterized almost full and almost empty flags
- ❖ Parameterized reset mode (synchronous or asynchronous)
- ❖ Interfaces to common hard macro or compiled ASIC dual-port synchronous RAMs



Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset input, active low asynchronous if <i>rst_mode</i> = 0, synchronous if <i>rst_mode</i> = 1
push_req_n	1 bit	Input	FIFO push request, active low
pop_req_n	1 bit	Input	FIFO pop request, active low
diag_n	1 bit	Input	Diagnostic control for <i>err_mode</i> = 0, NC for other <i>err_mode</i> values), active low
we_n	1 bit	Output	Write enable output for write port of RAM, active low
empty	1 bit	Output	FIFO empty output, active high
almost_empty	1 bit	Output	FIFO almost empty output, asserted when FIFO level $\leq ae\_level$ , active high
half_full	1 bit	Output	FIFO half full output, active high
almost_full	1 bit	Output	FIFO almost full output, asserted when FIFO level $\geq (depth - af\_level)$ , active high
full	1 bit	Output	FIFO full output, active high
error	1 bit	Output	FIFO error output, active high
wr_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Output	Address output to write port of RAM
rd_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Output	Address output to read port of RAM

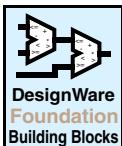
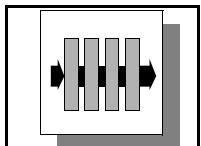


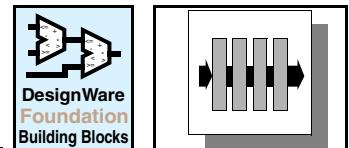
Table 1-2 Parameter Description

Parameter	Values	Function
depth	2 to $2^{24}$ Default: 4	Number of memory elements used in FIFO (used to size the address ports)
ae_level	1 to $depth - 1$ Default: 1	Almost empty level (the number of words in the FIFO at or below which the almost_empty flag is active)
af_level	1 to $depth - 1$ Default: 1	Almost full level (the number of empty memory locations in the FIFO at which the almost_full flag is active.)
err_mode	0 to 2 Default: 0	Error mode 0 = underflow/overflow and pointer latched checking, 1 = underflow/overflow latched checking, 2 = underflow/overflow unlatched checking
rst_mode	0 or 1 Default: 0	Reset mode 0 = asynchronous reset, 1 = synchronous reset

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl <sup>a</sup>	Synthesis model	DesignWare

- a. The implementation, “rtl” replaces the obsolete implementations “rpl,” “cl1,” and “cl2.” Information messages listing implementation replacements (SYNDB-37) may be generated by DC at compile time. Existing designs that specify an obsolete implementation (“rpl,” “cl1,” and “cl2”) will automatically have that implementation replaced by the new superseding implementation (“rtl”) noted by an information message (SYNDB-36) generated during DC compilation. The new implementation is capable of producing any of the original architectures automatically based on user constraints.

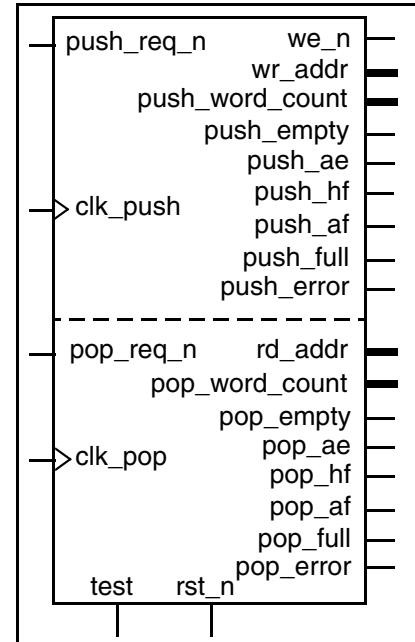
**DW\_fifoctl\_s2\_sf**

Synchronous (Dual Clock) FIFO Controller with Static Flags

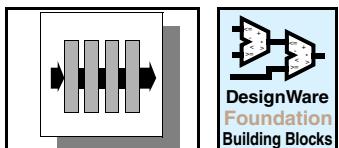
**DW\_fifoctl\_s2\_sf**

Synchronous (Dual Clock) FIFO Controller with Static Flags

- ❖ Fully registered synchronous flag output ports
- ❖ Single clock cycle push and pop operations
- ❖ Separate status flags for each clock system
- ❖ FIFO empty, half full, and full flags
- ❖ FIFO push error (overflow) and pop error (underflow) flags
- ❖ Parameterized word depth
- ❖ Parameterized almost full and almost empty flag thresholds
- ❖ Interfaces to common hard macro or compiled ASIC dual-port synchronous RAMs
- ❖ Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

<b>Pin Name</b>	<b>Width</b>	<b>Direction</b>	<b>Function</b>
clk_push	1 bit	Input	Input clock for push interface
clk_pop	1 bit	Input	Input clock for pop interface
rst_n	1 bit	Input	Reset input, active low
push_req_n	1 bit	Input	FIFO push request, active low
pop_req_n	1 bit	Input	FIFO pop request, active low
we_n	1 bit	Output	Write enable output for write port of RAM, active low
push_empty	1 bit	Output	FIFO empty <sup>a</sup> output flag synchronous to clk_push, active high
push_ae	1 bit	Output	FIFO almost empty <sup>a</sup> output flag synchronous to clk_push, active high (determined by <i>push_ae_lvl</i> parameter)
push_hf	1 bit	Output	FIFO half full <sup>a</sup> output flag synchronous to clk_push, active high
push_af	1 bit	Output	FIFO almost full <sup>a</sup> output flag synchronous to clk_push, active high (determined by <i>push_af_lvl</i> parameter)
push_full	1 bit	Output	FIFO full <sup>a</sup> output flag synchronous to clk_push, active high



## DW\_fifoctl\_s2\_sf

### Synchronous (Dual Clock) FIFO Controller with Static Flags

Table 1-1 Pin Description (Continued)

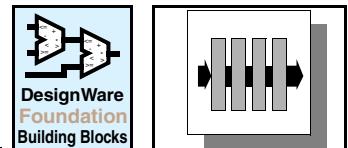
Pin Name	Width	Direction	Function
push_error	1 bit	Output	FIFO push error (overrun) output flag synchronous to clk_push, active high
pop_empty	1 bit	Output	FIFO empty <sup>b</sup> output flag synchronous to clk_pop, active high
pop_ae	1 bit	Output	FIFO almost empty <sup>b</sup> output flag synchronous to clk_pop, active high (determined by pop_ae_lvl parameter)
pop_hf	1 bit	Output	FIFO half full <sup>b</sup> output flag synchronous to clk_pop, active high
pop_af	1 bit	Output	FIFO almost full <sup>b</sup> output flag synchronous to clk_pop, active high (determined by pop_af_lvl parameter)
pop_full	1 bit	Output	FIFO full <sup>b</sup> output flag synchronous to clk_pop, active high
pop_error	1 bit	Output	FIFO pop error (underrun) output flag synchronous to clk_pop, active high
wr_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Output	Address output to write port of RAM
rd_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Output	Address output to read port of RAM
push_word_count	ceil(log <sub>2</sub> [depth+1]) bit(s)	Output	Words in FIFO (as perceived by the push/pop interface)
pop_word_count	ceil(log <sub>2</sub> [depth+1]) bit(s)	Output	Words in FIFO (as perceived by the push/pop interface)
test	1 bit	Input	Active high, test input control for inserting scan test lock-up latches

a. As perceived by the push interface.

b. As perceived by the pop interface.

Table 1-2 Parameter Description

Parameter	Values	Description
depth	4 to 2 <sup>24</sup> Default: 8	Number of words that can be stored in FIFO
push_ae_lvl	1 to depth – 1 Default: 2	Almost empty level for the push_ae output port (the number of words in the FIFO at or below which the push_ae flag is active)
push_af_lvl	1 to depth – 1 Default: 2	Almost full level for the push_af output port (the number of empty memory locations in the FIFO at which the push_af flag is active)
pop_ae_lvl	1 to depth – 1 Default: 2	Almost empty level for the pop_ae output port (the number of words in the FIFO at or below which the pop_ae flag is active)
pop_af_lvl	1 to depth – 1 Default: 2	Almost full level for the pop_af output port (the number of empty memory locations in the FIFO at which the pop_af flag is active)

**DW\_fifoctl\_s2\_sf**

Synchronous (Dual Clock) FIFO Controller with Static Flags

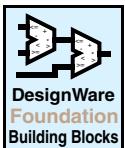
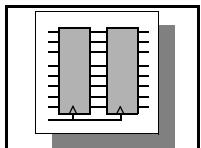
**Table 1-2 Parameter Description (Continued)**

Parameter	Values	Description
err_mode	0 or 1 Default: 0	Error mode 0 = stays active until reset [latched], 1 = active only as long as error condition exists [unlatched]
push_sync	1 to 3 Default: 2	Push flag synchronization mode 1 = single register synchronization from pop pointer, 2 = double register, 3 = triple register)
pop_sync	1 to 3 Default: 2	Pop flag synchronization mode 1 = single register synchronization from push pointer, 2 = double register, 3 = triple register
rst_mode	0 or 1 Default: 0	Reset mode 0 = asynchronous reset, 1 = synchronous reset)
tst_mode	0 or 1 Default: 0	Test Mode 0 = test input not connected 1 = lock-up latches inserted for scan test

**Table 1-3 Synthesis Implementations<sup>a</sup>**

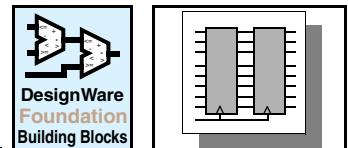
Implementation Name	Function	License Feature Required
rpl	Ripple Carry synthesis model	DesignWare
cl2	Full Carry lookahead model	DesignWare

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*



# Memory – Registers

This section documents the various memory registers found in the library of DesignWare Building Block IP.



## DW03\_pipe\_reg

### Pipeline Register

## DW03\_pipe\_reg

### Pipeline Register

- ❖ Parameterized data width and depth



**Table 1-1 Pin Description**

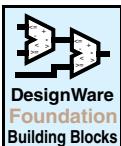
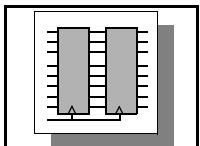
Pin Name	Width	Direction	Function
A	<i>width</i> bit(s)	Input	Input data bus
clk	1 bit	Input	Clock
B	<i>width</i> bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
depth	$\geq 1$	Depth of registers
width	$\geq 1$	Width of A and B buses

**Table 1-3 Synthesis Implementations**

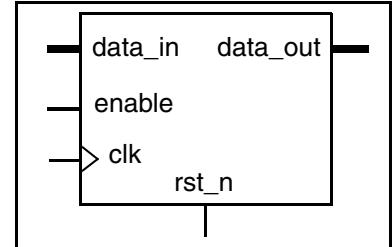
Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



## DW\_pl\_reg

### Pipeline Register

- ❖ Parameter controlled width
- ❖ Parameter controlled logic stages
- ❖ Parameter controlled input or output register
- ❖ Individual enables per register level
- ❖ Auto ungroups itself into its parent design for register retiming



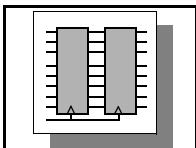
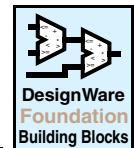
**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock input
rst_n	1 bit	Input	Active low reset input (see rst_mode parameter)
enable	stages+in_reg+out_reg-1	Input	Bus of enables for each stage of registers
data_in	<i>width</i> bit(s)	Input	Input data bus
data_out	<i>width</i> bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	>0 Default: 8	Width of data_in and data_out buses (and all register between data_in and data_out)
in_reg	0 or 1 Default: 0	Input register control 0 => no input register 1 => input register **
stages	1 to 1024 Default: 4	Controls the number of logic stages in the pipeline.
out_reg	0 or 1 Default 0	Output register control 0 => no output register 1 => output register **
rst_mode	0 or 1 Default 0	Reset type control 0 => asynchronous reset 1 => synchronous reset

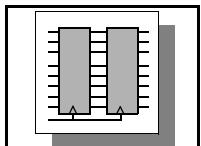
\*\* - The parameters in\_reg and out\_reg cannot both be set to 1. In DC versions prior to A-2007.12, input and output registers are not allowed. Thus the value of both in\_reg and out\_reg parameters must be 0 when using DC versions earlier than A-2007.12

**DW\_pl\_reg**

## Pipeline Register

**Table 1-3 Synthesis Implementations**

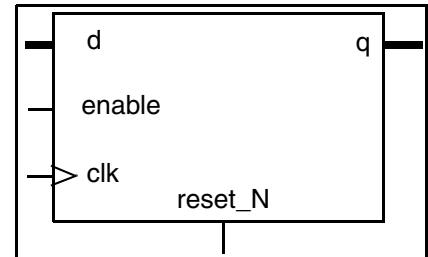
Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



## DW03\_reg\_s\_pl

Register with Synchronous Enable Reset

- ❖ Parameterizable data width
- ❖ Parameterized reset to any constant value
- ❖ Multiple synthesis implementations
- ❖ Provides minPower benefits with the DesignWare-LP license.



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
d	<i>width</i> bit(s)	Input	Input data bus
clk	1 bit	Input	Clock
reset_N	1 bit	Input	Synchronous reset
enable	1 bit	Input	Enables all operations
q	<i>width</i> bi(s)	Output	Output data bus

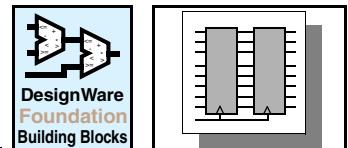
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	1 to 31 Default: 8	Width of d and q buses
reset_value	0 to $2^{width}-1$ when width $\leq 31$ ; 0 when width $\geq 32$ Default: 0	Resets to a constant

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
str	Single-bit flip-flops synthesis model	DesignWare
mbstr	Multiple-bit flip-flops 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*

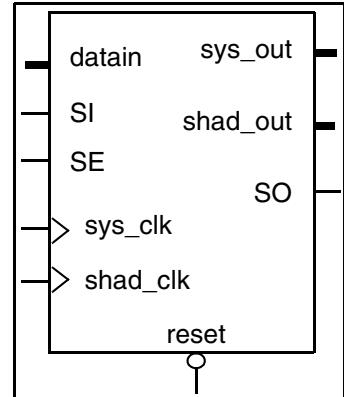
**DW04\_shad\_reg**

Shadow and Multibit Register

**DW04\_shad\_reg**

Shadow and Multibit Register

- ❖ Captures the state of system registers dynamically during system operation
- ❖ Serial access on shadow register to scan out the state of captured data
- ❖ Constructed with multibit flip-flop cells where possible; can be used as a simple, non-shadowed multibit register
- ❖ Parameterized width and number of registers (one or two)

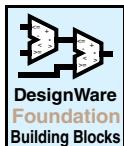
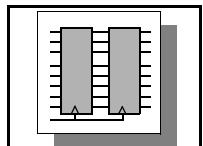
**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
datain	<i>width</i> bit(s)	Input	Data input driving the input to the system register
sys_clk	1 bit	Input	Clock that samples the system register, positive edge triggered
shad_clk	1 bit	Input	Signal that clocks the output of the system register into the shadow register, positive edge triggered
reset	1 bit	Input	Asynchronous reset signal that clears the system and shadow registers
SI	1 bit	Input	Serial scan input, clocked by shad_clk when SE is high
SE	1 bit	Input	Serial scan enable, active high. Enables scan only on the shadow register.
sys_out	<i>width</i> bit(s)	Output	Output of the system register
shad_out	<i>width</i> bit(s)	Output	Parallel output of shadow register that lags system register by one cycle
SO	1 bit	Output	Serial scan output from shadow register. When SE is low, represents the state of the MSB of the shadow register. When SE is high, each successive bit is shifted up one and SI is clocked into the LSB.

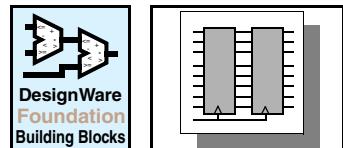
**Table 1-2 Parameter Description**

Parameter	Values	Description
width <sup>1</sup>	1 to 512 <sup>a</sup>	Defines width of system and shadow registers, and the input and output buses
bld_shad_reg	0 or 1	Defines whether to build both the system and shadow registers (bld_shad_reg = 1) or just the system register (bld_shad_reg = 0)

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

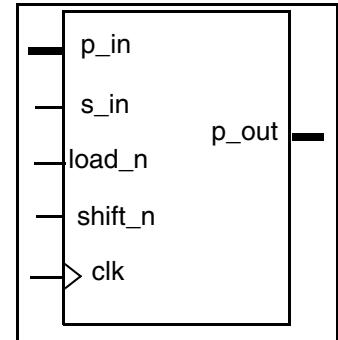
**DW03\_shftreg**

Shift Register

**DW03\_shftreg**

Shift Register

- ❖ Parameterized word length
- ❖ Active low shift enable
- ❖ Active low load enable
- ❖ Provides minPower benefits with the DesignWare-LP license.

**Table 1-1 Pin Description**

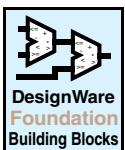
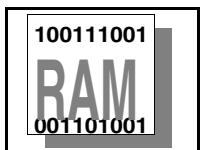
Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
s_in	1 bit	Input	Serial shift input
p_in	<i>length</i> bit(s)	Input	Parallel input
shift_n	1 bit	Input	Shift enable, active low
load_n	1 bit	Input	Parallel load enable, active low
p_out	<i>length</i> bit(s)	Output	Shift register parallel output

**Table 1-2 Parameter Description**

Parameter	Values	Description
length	$\geq 1$	Length of shifter

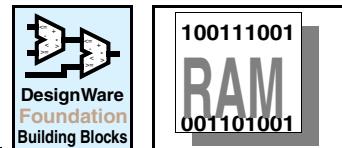
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



## Memory – Synchronous RAMs

This section documents the various DesignWare Building Block IP memory synchronous RAMs.

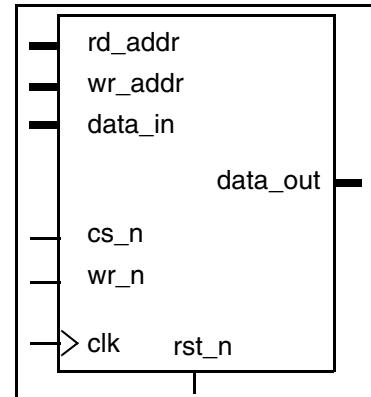
**DW\_ram\_r\_w\_s\_dff**

Synchronous Write-Port, Asynchronous Read-Port RAM (Flip-Flop-Based)

**DW\_ram\_r\_w\_s\_dff**

Synchronous Write-Port, Asynchronous Read-Port RAM (Flip-Flop-Based)

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Synchronous static memory
- ❖ Parameterized reset mode (synchronous or asynchronous)
- ❖ High testability using DFT Compiler

**Table 1-1 Pin Description**

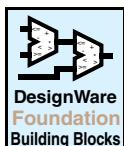
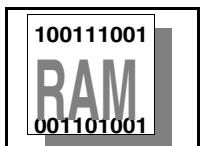
Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rd_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Read address bus
wr_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Write address bus
data_in	<i>data_width</i> bit(s)	Input	Input data bus
data_out	<i>data_width</i> bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 2048 Default = none	Width of data_in and data_out buses
depth	2 to 1024 Default = none	Number of words in the memory array (address width)
rst_mode	0 or 1 Default = 1	Determines the reset methodology: 0 = rst_n asynchronously initializes the RAM, 1 = rst_n synchronously initializes the RAM

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

**DW\_ram\_r\_w\_s\_lat**

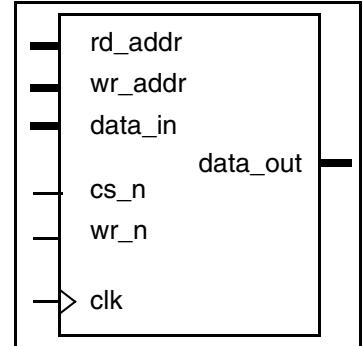
Synchronous Write Port, Asynchronous Read Port RAM (Latch-Based)

**DW\_ram\_r\_w\_s\_lat**

Synchronous Write Port, Asynchronous Read Port RAM (Latch-Based)

**Features and Benefits**

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Synchronous static memory

**Description**

DW\_ram\_r\_w\_s\_lat implements a parameterized synchronous, dual-port static RAM. The RAM can perform simultaneous read and write operations.

**Table 1-1 Pin Description**

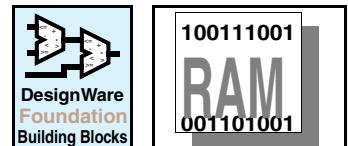
Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rd_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Read address bus
wr_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Write address bus
data_in	<i>data_width</i> bit(s)	Input	Input data bus
data_out	<i>data_width</i> bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 256 Default = none	Width of data_in and data_out buses
depth	2 to 256 Default = none	Number of words in the memory array (address width)

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

**DW\_ram\_2r\_w\_s\_dff**

Synchronous Write Port, Asynchronous Dual Read Port RAM (FF-Based)

**DW\_ram\_2r\_w\_s\_dff**

Synchronous Write Port, Asynchronous Dual Read Port RAM (FF-Based)

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Synchronous static memory
- ❖ Parameterized reset mode (synchronous or asynchronous)
- ❖ High testability using DFT Compiler

**Table 1-1 Pin Description**

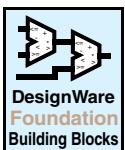
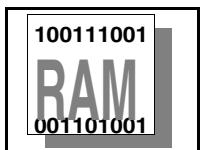
Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rd1_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Read1 address bus
rd2_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Read2 address bus
wr_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Write address bus
data_in	<i>data_width</i> bit(s)	Input	Input data bus
data_rd1_out	<i>data_width</i> bit(s)	Output	Output data bus for read1
data_rd2_out	<i>data_width</i> bit(s)	Output	Output data bus for read2

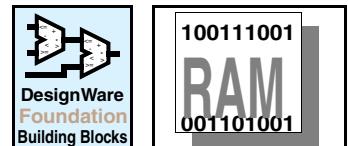
**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 2048 Default = none	Width of data_in and data_out buses
depth	2 to 1024 Default = none	Number of words in the memory array (address width)
rst_mode	0 or 1 Default = 1	Determines the reset methodology: 0 = rst_n asynchronously initializes the RAM, 1 = rst_n synchronously initializes the RAM

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



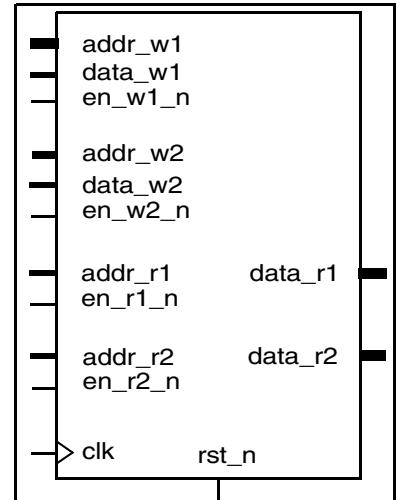
**DW\_ram\_2r\_2w\_s\_dff**

Synchronous Dual Write Port, Async. Dual Read Port RAM (FF-Based)

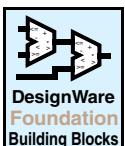
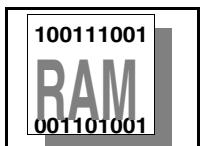
**DW\_ram\_2r\_2w\_s\_dff**

Synchronous Dual Write Port, Async. Dual Read Port RAM (FF-Based)

- ❖ Parameter controlled data width
- ❖ Parameter controlled address width (controls memory size)
- ❖ Synchronous static memory
- ❖ Parameter controlled reset mode (synchronous or asynchronous)

**Table 1-1 Pin Description**

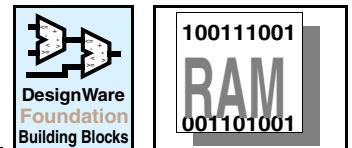
Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, active low
en_w1_n	1 bit	Input	Write port 1 enable, active low
addr_w1	<i>addr_width</i>	Input	Write port 1 address
data_w1	<i>width</i>	Input	Write port 1 data in
en_w2_n	1 bit	Input	Write port 2 enable, active low
addr_w2	<i>addr_width</i>	Input	Write port 2 address
data_w2	<i>width</i>	Input	Write port 2 data in
en_r1_n	1 bit	Input	Read port 1 enable, active low
addr_r1	<i>addr_width</i>	Input	Read port 1 address
data_r1	<i>width</i>	Output	Read port 1 data out
en_r2_n	1 bit	Input	Read port 2 enable, active low
addr_r2	<i>addr_width</i>	Input	Read port 2 address
data_r2	<i>width</i>	Output	Read port 2 data out

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	1 to 8192 Default = 8	Data width
addr_width	1 to 12 Default = 3	Address bus width - which controls memory depth.
rst_mode	0 or 1 Default = 0	Determines the reset methodology: 0 = rst_n asynchronously initializes the RAM, 1 = rst_n synchronously initializes the RAM

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

**DW\_ram\_2r\_w\_s\_lat**

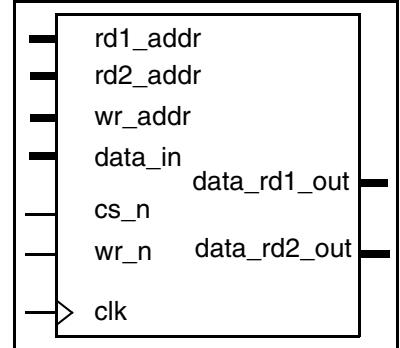
Synchronous Write Port, Asynchronous Dual Read Port RAM (Latch-Based)

**DW\_ram\_2r\_w\_s\_lat**

Synchronous Write Port, Asynchronous Dual Read Port RAM (Latch-Based)

**Features and Benefits**

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Synchronous static memory

**Table 1-1 Pin Description**

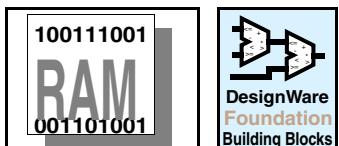
Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rd1_addr	ceil(log <sub>2</sub> [depth]) bit	Input	Read1 address bus
rd2_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Read2 address bus
wr_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Write address bus
data_in	<i>data_width</i> bit(s)	Input	Input data bus
data_rd1_out	<i>data_width</i> bit(s)	Output	Output data bus for read1
data_rd2_out	<i>data_width</i> bit(s)	Output	Output data bus for read2

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 256 Default = none	Width of data_in and data_out buses
depth	2 to 256 Default = none	Number of words in the memory array (address width)

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

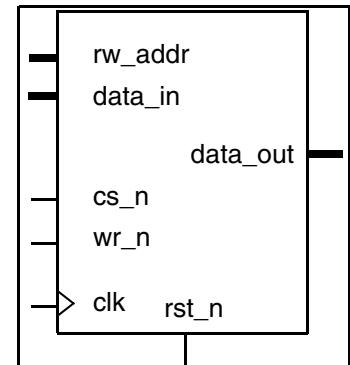


## **DW\_ram\_rw\_s\_dff**

Synchronous Single Port Read/Write RAM (Flip-Flop-Based)

### Features and Benefits

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Synchronous static memory
- ❖ Parameterized reset mode (asynchronous or synchronous )
- ❖ High testability using DFT Compiler



**Table 1-1 Pin Description**

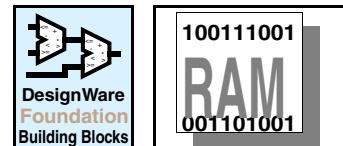
Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rw_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Address bus
data_in	<i>data_width</i> bit(s)	Input	Input data bus
data_out	<i>data_width</i> bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 2048 Default = none	Width of data_in and data_out buses
depth	2 to 1024 Default = none	Number of words in the memory array (address width)
rst_mode	0 or 1 Default = 1	Determines the reset methodology: 0 = rst_n asynchronously initializes the RAM, 1 = rst_n synchronously initializes the RAM

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

**DW\_ram\_rw\_s\_lat**

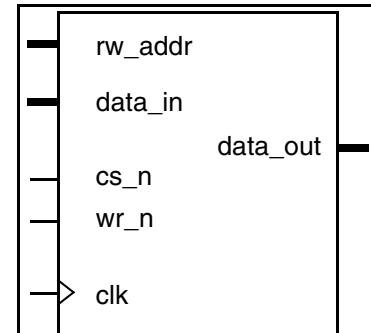
Synchronous Single Port Read/Write RAM (Latch-Based)

**DW\_ram\_rw\_s\_lat**

Synchronous Single Port Read/Write RAM (Latch-Based)

**Features and Benefits**

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Synchronous static memory

**Description**

DW\_ram\_rw\_s\_lat implements a parameterized, synchronous, single-port static RAM.

**Table 1-1 Pin Description**

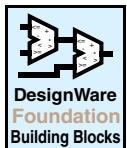
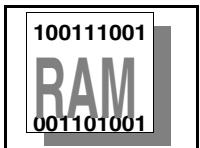
Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rw_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Address bus
data_in	$\text{data\_width}$ bit(s)	Input	Input data bus
data_out	$\text{data\_width}$ bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 256 Default = none	Width of data_in and data_out buses
depth	2 to 256 Default = none	Number of words in the memory array (address width)

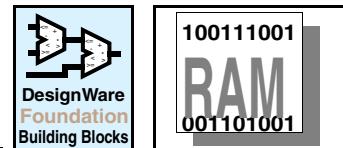
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



## Memory – Asynchronous RAMs

This section documents the various DesignWare Building Block IP memory asynchronous RAMs.

**DW\_ram\_r\_w\_a\_dff**

Asynchronous Dual Port RAM (Flip-Flop-Based)

**DW\_ram\_r\_w\_a\_dff**

Asynchronous Dual Port RAM (Flip-Flop-Based)

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Asynchronous static memory
- ❖ Parameterized reset implementation
- ❖ High testability using DFT Compiler

**Table 1-1 Pin Description**

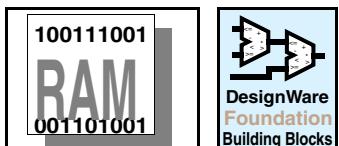
Pin Name	Width	Direction	Function
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
test_mode	1 bit	Input	Enables test_clk
test_clk	1 bit	Input	Test clock to capture data during test_mode
rd_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Read address bus
wr_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Write address bus
data_in	<i>data_width</i> bit(s)	Input	Input data bus
data_out	<i>data_width</i> bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 256 Default = none	Width of data_in and data_out buses
depth	2 to 256 Default = none	Number of words in the memory array (address width)
rst_mode	0 or 1 Default = 1	Determines if the rst_n input is used. 0 = rst_n initializes the RAM, 1 = rst_n is not connected

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



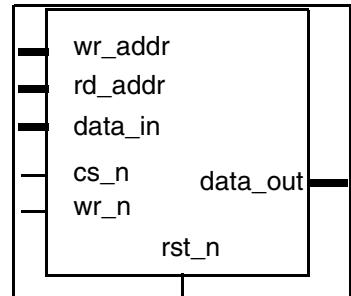
## DW\_ram\_r\_w\_a\_lat

Asynchronous Dual Port RAM (Latch-Based)

### Features and Benefits

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Asynchronous static memory
- ❖ Parameterized reset implementation

DW\_ram\_r\_w\_a\_lat implements a parameterized, asynchronous, dual-port static RAM.



**Table 1-1 Pin Description**

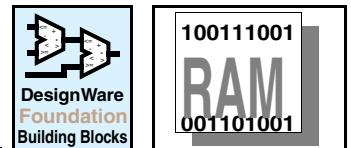
Pin Name	Width	Direction	Function
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rd_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Read address bus
wr_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Write address bus
data_in	<i>data_width</i> bit(s)	Input	Input data bus
data_out	<i>data_width</i> bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 256 Default = none	Width of data_in and data_out buses
depth	2 to 256 Default = none	Number of words in the memory array (address width)
rst_mode	0 or 1 Default = 1	Determines if the rst_n input is used. 0= rst_n initializes the RAM, 1= rst_n is not connected

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

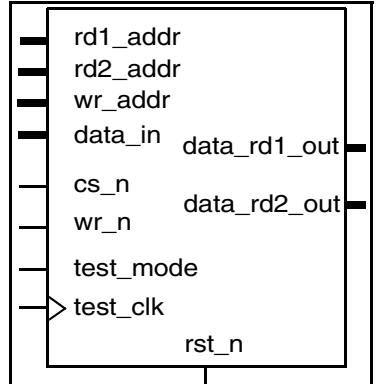
**DW\_ram\_2r\_w\_a\_dff**

Write Port, Dual Read Port RAM (Flip-Flop-Based)

**DW\_ram\_2r\_w\_a\_dff**

Write Port, Dual Read Port RAM (Flip-Flop-Based)

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Asynchronous static memory
- ❖ Parameterized reset implementation
- ❖ High testability using DFT Compiler

**Table 1-1 Pin Description**

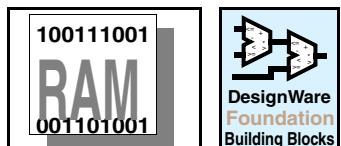
Pin Name	Width	Direction	Function
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
test_mode	1 bit	Input	Enables test_clk
test_clk	1 bit	Input	Test clock to capture data during test_mode
rd1_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Read1 address bus
rd2_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Read2 address bus
wr_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Write address bus
data_in	data_width bit(s)	Input	Input data bus
data_rd1_out	data_width bit(s)	Output	Output data bus for read1
data_rd2_out	data_width bit(s)	Output	Output data bus for read2

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 256	Width of data_in and data_out buses. Default = none
depth	2 to 256	Number of words in the memory array (address width). Default = none
rst_mode	0 or 1 Default = 1	Determines if the rst_n input is used. 0 = rst_n initializes the RAM, 1 = rst_n is not connected

**Table 1-3 Synthesis Implementations**

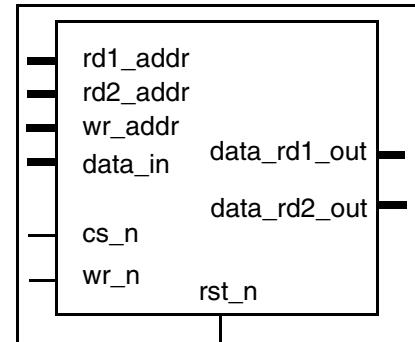
Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



## DW\_ram\_2r\_w\_a\_lat

Write Port, Dual Read Port RAM (Latch-Based)

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Asynchronous static memory
- ❖ Parameterized reset implementation



**Table 1-1 Pin Description**

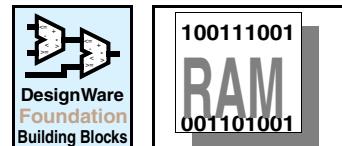
Pin Name	Width	Direction	Function
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rd1_addr	$\text{ceil}(\log_2[\text{depth}])$ bit	Input	Read1 address bus
rd2_addr	$\text{ceil}(\log_2[\text{depth}])$ bit	Input	Read2 address bus
wr_addr	$\text{ceil}(\log_2[\text{depth}])$ bit	Input	Write address bus
data_in	<i>data_width</i> bit	Input	Input data bus
data_rd1_out	<i>data_width</i> bit	Output	Output data bus for read1
data_rd2_out	<i>data_width</i> bit	Output	Output data bus for read2

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 256 Default = none	Width of data_in and data_out buses
depth	2 to 256 Default = none	Number of words in the memory array (address width)
rst_mode	0 or 1 Default = 1	Determines if the rst_n input is used. 0 = rst_n initializes the RAM, 1 = rst_n is not connected

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare

**DW\_ram\_rw\_a\_dff**

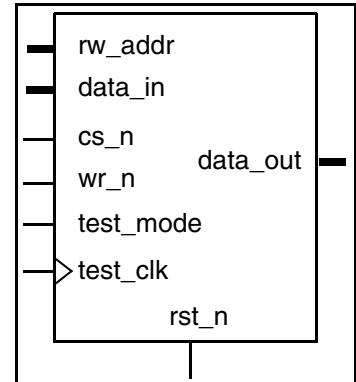
Asynchronous Single Port RAM (Flip-Flop-Based)

**DW\_ram\_rw\_a\_dff**

Asynchronous Single Port RAM (Flip-Flop-Based)

**Features and Benefits**

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Asynchronous static memory
- ❖ Parameterized reset implementation
- ❖ High testability using DFT Compiler

**Table 1-1 Pin Description**

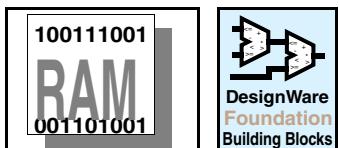
Pin Name	Width	Direction	Function
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
test_mode	1 bit	Input	Enables test_clk
test_clk	1 bit	Input	Test clock to capture data during test_mode
rw_addr	ceil(log <sub>2</sub> [depth]) bit(s)	Input	Address bus
data_in	<i>data_width</i> bit(s)	Input	Input data bus
data_out	<i>data_width</i> bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 256	Width of data_in and data_out buses. Default = none
depth	2 to 256	Number of words in the memory array (address width). Default = none
rst_mode	0 or 1 Default = 1	Determines if the rst_n input is used. 0 = rst_n initializes the RAM, 1 = rst_n is not connected

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



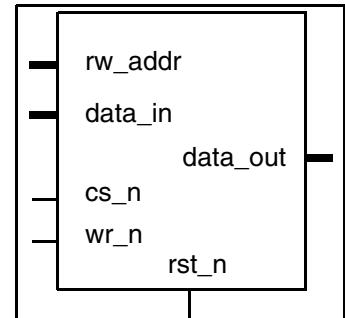
## DW\_ram\_rw\_a\_lat

Asynchronous Single-Port RAM (Latch-Based)

### Features and Benefits

- ❖ Parameterized word depth
- ❖ Parameterized data width
- ❖ Asynchronous static memory
- ❖ Parameterized reset implementation

### Description



DW\_ram\_rw\_a\_lat implements a parameterized, asynchronous, single-port static RAM.

**Table 1-1 Pin Description**

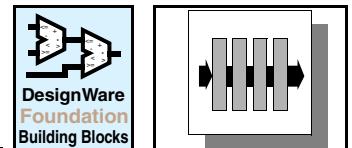
Pin Name	Width	Direction	Function
rst_n	1 bit	Input	Reset, active low
cs_n	1 bit	Input	Chip select, active low
wr_n	1 bit	Input	Write enable, active low
rw_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Input	Address bus
data_in	<i>data_width</i> bit(s)	Input	Input data bus
data_out	<i>data_width</i> bit(s)	Output	Output data bus

**Table 1-2 Parameter Description**

Parameter	Values	Description
data_width	1 to 256 Default = none	Width of data_in and data_out buses
depth	2 to 256 Default = none	Number of words in the memory array (address width)
rst_mode	0 or 1 Default = 1	Determines if the rst_n input is used. 0 = rst_n initializes the RAM, 1 = rst_n is not connected

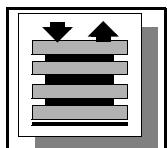
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare



## Memory – Stacks

This section documents the various DesignWare Building Block IP memory stacks.

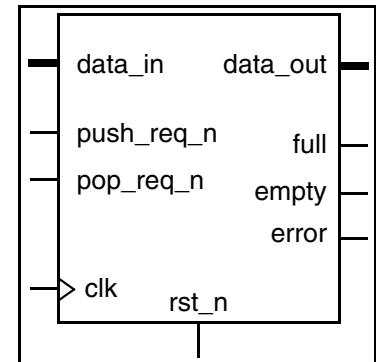


# DW\_stack

Synchronous (Single-Clock) Stack

## Features and Benefits

- ❖ Parameterized word width and depth
- ❖ Stack empty and full status flags
- ❖ Stack error flag indicating underflow and overflow
- ❖ Fully registered synchronous flag output ports
- ❖ All operations execute in a single clock cycle
- ❖ D flip-flop based memory array for high testability
- ❖ Parameterized reset mode (synchronous or asynchronous)

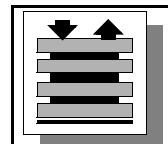
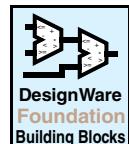


## Description

DW\_stack is a fully synchronous, single-clock stack. It combines the DW\_stackctl stack controller and the DW\_ram\_r\_w\_s\_dff flip-flop-based RAM DesignWare components.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset input, active low asynchronous if <i>rst_mode</i> = 0 or 2, synchronous if <i>rst_mode</i> = 1 or 3
push_req_n	1 bit	Input	Stack push request, active low
pop_req_n	1 bit	Input	Stack pop request, active low
data_in	<i>data_width</i> bit(s)	Input	Stack push data
empty	1 bit	Output	Stack empty flag, active high
full	1 bit	Output	Stack full flag, active high
error	1 bit	Output	Stack error output, active high
data_out	<i>data_width</i> bit(s)	Output	Stack pop data

**DW\_stack**

Synchronous (Single-Clock) Stack

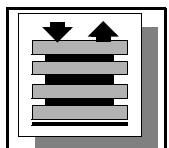
**Table 1-2 Parameter Description**

Parameter	Values	Description
width	1 to 256 Default: None	Width of data_in and data_out buses
depth	2 to 256 Default: None	Depth (in words) of memory array
err_mode	0 or 1 Default: 0	Error mode 0 = underflow/overflow error, hold until reset, 1 = underflow/overflow error, hold until next clock.
rst_mode	0 to 3 Default: 0	Reset mode 0 = asynchronous reset including memory, 1 = synchronous reset including memory, 2 = asynchronous reset excluding memory, 3 = synchronous reset excluding memory.

**Table 1-3 Synthesis Implementations<sup>a</sup>**

Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cl2	Full carry look-ahead 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*

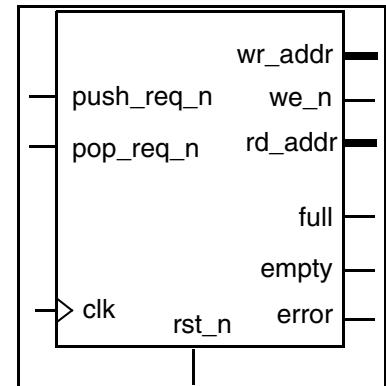


# DW\_stackctl

Synchronous (Single-Clock) Stack Controller

## Features and Benefits

- ❖ Parameterized word width and depth
- ❖ Stack empty and full status flags
- ❖ Stack error flag indicating underflow and overflow
- ❖ Fully registered synchronous address and flag output ports
- ❖ All operations execute in a single clock cycle
- ❖ Parameterized reset mode (synchronous or asynchronous)
- ❖ Interfaces with common hard macro or compiled ASIC dual-port synchronous RAMs
- ❖ Provides minPower benefits with the DesignWare-LP license.

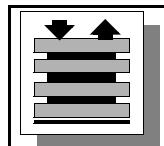
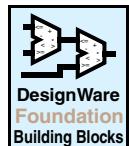


## Description

DW\_stackctl is a stack RAM controller designed to interface with a dual-port synchronous RAM.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock
rst_n	1 bit	Input	Reset input, active low asynchronous if <code>rst_mode = 0</code> , synchronous if <code>rst_mode = 1</code>
push_req_n	1 bit	Input	Stack push request, active low
pop_req_n	1 bit	Input	Stack pop request, active low
we_n	1 bit	Output	Write enable for RAM write port, active low
empty	1 bit	Output	Stack empty flag, active high
full	1 bit	Output	Stack full flag, active high
error	1 bit	Output	Stack error output, active high
wr_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Output	Address output to write port of RAM
rd_addr	$\text{ceil}(\log_2[\text{depth}])$ bit(s)	Output	Address output to read port of RAM

**DW\_stackctl**

Synchronous (Single-Clock) Stack Controller

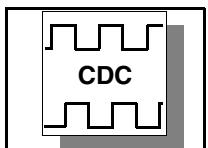
**Table 1-2 Parameter Description**

Parameter	Values	Function
depth	2 to $2^{24}$ Default: None	Number of memory elements in the stack [used to size the address ports]
err_mode	0 or 1 Default: 0	Error mode 0 = underflow/overflow error, hold until reset, 1 = underflow/overflow error, hold until next clock.
rst_mode	0 or 1 Default: 0	Reset mode 0 = asynchronous reset, 1 = synchronous reset.

**Table 1-3 Synthesis Implementations<sup>a</sup>**

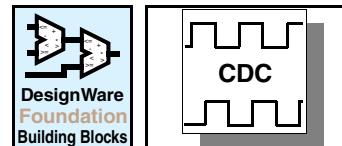
Implementation Name	Function	License Feature Required
rpl	Ripple carry synthesis model	DesignWare
cl2	Full carry look-ahead model	DesignWare

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*.



# Interface – Clock Domain Crossing

This section documents the Clock Domain Crossing (CDC) components in the DesignWare Building Block IP.

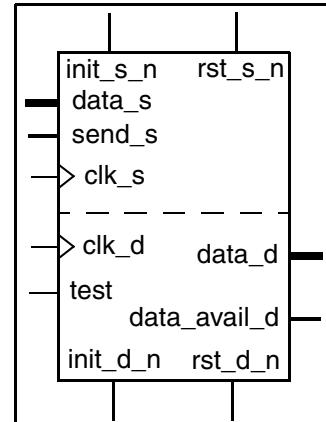
**DW\_data\_qsync\_hl**

Quasi-Synchronous Data Interface for H-to-L Frequency Clocks

**DW\_data\_qsync\_hl**

Quasi-Synchronous Data Interface for H-to-L Frequency Clocks

- ❖ Fully parametrized bus width
- ❖ Parameterized clock ratio
- ❖ Data available flag in destination clock domain

**Table 1-1 Pin Description**

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)
send_s	1	Input	Source Domain send request input
data_s	width	Input	Source Domain send data input
clk_d	1	Input	Destination clock source
rst_d_n	1	Input	Destination domain asynchronous reset (active low)
init_d_n	1	Input	Destination domain synchronous reset (active low)
test	1	Input	Scan test mode select
data_d	width	Output	Destination domain data output
data_avail_d	1	Output	Destination domain data update output

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	1 to 1024 Default: 8	Vector width of input data_s and output data_d
clk_ratio	2 to 1024 Default: 2	Integer value of the high speed clock period divided by the low speed clock period

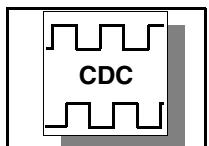
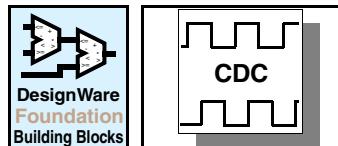


Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
tst_mode	0 to 2 Default: 0	Test mode 0 = no latch is inserted for scan testing 1 = insert negative-edge capturing register on data_s input vector when test input is asserted 2 = reserved

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

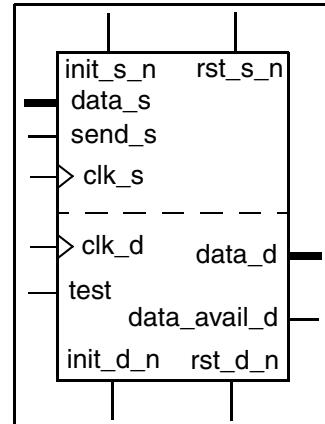
**DW\_data\_qsync\_lh**

Quasi-Synchronous Data Interface for L-to-H Frequency Clocks

**DW\_data\_qsync\_lh**

Quasi-Synchronous Data Interface for L-to-H Frequency Clocks

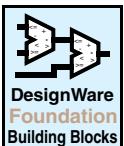
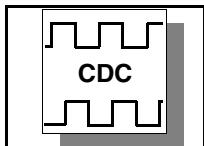
- ❖ Fully Tested clock domain crossing
- ❖ Fully parametrized
- ❖ Parameterized output registration: either combinatorial or registered outputs
- ❖ Applications: data bus controllers, bus-based communication circuits, any interface sending parallel data between two clock domains

**Table 1-1 Pin Description**

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)
send_s	1	Input	Source Domain send request input
data_s	width	Input	Source Domain send data input
clk_d	1	Input	Destination clock source
rst_d_n	1	Input	Destination asynchronous reset (active low)
init_d_n	1	Input	Destination synchronous reset (active low)
test	1	Input	Scan test mode select
data_d	width	Output	Destination data vector
data_avail_d	1	Output	Destination data update output

**Table 1-2 Parameter Description**

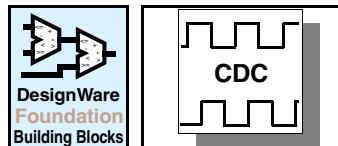
Parameter	Values	Description
width	1 to 1024 Default: 8	Vector width of input data_s and output data_d
clk_ratio	2 to 1024 Default: 2	Integer value of the high speed clock period divided by the low speed clock period
reg_data_s	0 to 1 Default: 1	0 = input data is not registered 1 = input data is registered
reg_data_d	0 to 1 Default: 1	0 = output data is not registered 1 = output data is registered

**Table 1-2 Parameter Description (Continued)**

Parameter	Values	Description
tst_mode	0 to 2 Default: 0	Test mode 0 = no latch is inserted for scan testing 1 = insert negative-edge capturing register on data_s input vector when test input is asserted 2 = reserved

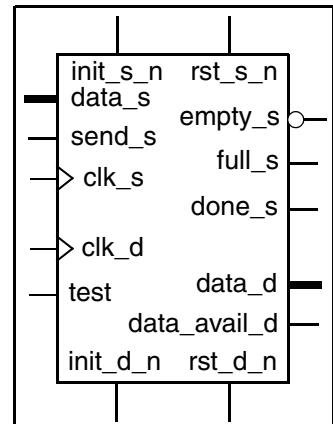
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

**DW\_data\_sync****DW\_data\_sync**

Single Clock Data Bus Synchronizer

- ❖ Fully Tested clock domain crossing
- ❖ Fully parametrized
- ❖ Selectable clock edge
- ❖ Parameterized output registration: either combinatorial or registered outputs
- ❖ Applications: data bus controllers, bus-based communication circuits, any interface sending parallel data between two clock domains

**Table 1-1 Pin Description**

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)
send_s	1	Input	Source initiate valid data vector control
data_s	width	Input	Source Domain data vector
empty_s	1	Output	Source domain transaction reg empty output (active low)
full_s	1	Output	Source domain transaction reg full output
done_s	1	Output	Source domain transaction done output
clk_d	1	Input	Destination clock source
rst_d_n	1	Input	Destination asynchronous reset (active low)
init_d_n	1	Input	Destination synchronous reset (active low)
data_avail_d	1	Output	Destination data update output
data_d	width	Output	Destination data vector
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
pend_mode	0 to 1 Default: 1	Buffer pending data
ack_delay	0 to 1 Default: 0	The acknowledge signal returned from the destination domain occurs either (0) before the second edge register, or (1) after the second edge detect register.

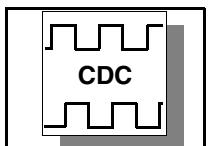
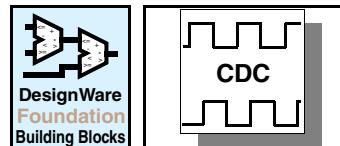


Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
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 first stage negative-edge capturing and second 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
r_sync_type	0 to 4 Default: 2	0 = single clock design (that is, clk_s == clk_d) 1 = first synchronization in clk_s domain is done on the negative edge and the rest on positive edge. This reduces latency req. of synchronization slightly but quicker metastability resolution for the negative edge sensitive FF. It also requires the technology library to contain an acceptable negative edge sensitive FF. 2 = all synchronization in clk_s domain is done on positive edges - 2 D flip flops in source domain. 3 = all synchronization in clk_s domain is done on positive edges - 3 D flip flops in source domain. 4= all synchronization in clk_s domain is done on positive edges - 4 D flip flops in source domain.
tst_mode	0 or 1 Default: 0	Test mode 0 = no latch is inserted for scan testing 1 = insert negative-edge capturing register on data_s input vector when test input is asserted
verif_en	0 to 4 Default: 0	Verification enable 0 = no sampling errors inserted, 1 = sampling errors are randomly inserted with 0 or up to 1 destination clock cycle delays 2 = sampling errors 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 randomly inserted with 0 or up to 0.5 destination clock cycle delays * For more information about verif_en, see the Simulation Methodology section.
send_mode	0 to 3 Default: 1	0 = single clock cycle pulse in produces single clock cycle pulse out 1 = rising edge transition in produces single clock cycle pulse out 2 = falling edge transition in produces single clock cycle pulse out 3 = rising & falling transition each produce single clock cycle pulse out

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

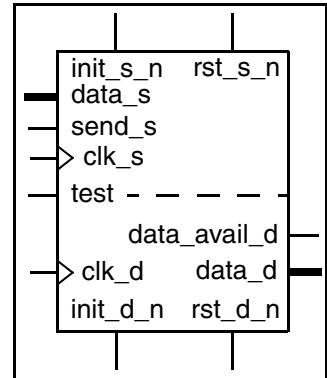
**DW\_data\_sync\_na**

Data Bus Synchronizer without Acknowledge

**DW\_data\_sync\_na**

Data Bus Synchronizer without Acknowledge

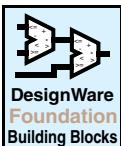
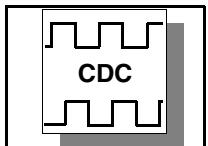
- ❖ Interface between dual asynchronous clock domains
- ❖ Parameterized data bus width
- ❖ Parameterized number of synchronizing stages
- ❖ Parameterized test feature
- ❖ All output registered
- ❖ Ability to model missampling of data on incoming clock domain

**Table 1-1 Pin Description**

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)
send_s	1	Input	Source initiate valid data vector control
data_s	<i>width</i>	Input	Source Domain data vector
clk_d	1	Input	Destination clock source
rst_d_n	1	Input	Destination asynchronous reset (active low)
init_d_n	1	Input	Destination synchronous reset (active low)
test	1	Input	Scan test mode select
data_avail_d	1	Output	Destination data update output
data_d	<i>width</i>	Output	Destination data vector

**Table 1-2 Parameter Description**

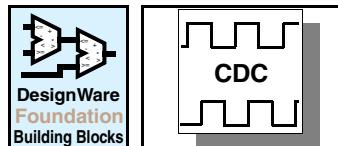
Parameter	Values	Description
width	1 to 1024 Default: 8	Vector width of input data_s and output data_d
f_sync_type	0 to 4 Default: 2	Forward synchronization type Defines type and number of synchronizing stages: 0 = single clock design, no synchronizing stages implemented 1 = 2-stage synchronization with first stage negative-edge capturing and second 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

**Table 1-2 Parameter Description**

Parameter	Values	Description
tst_mode	0 to 2 Default: 0	<p>Test mode</p> <p>0 = no latch is inserted for scan testing</p> <p>1 = insert negative-edge capturing register on data_s input vector when test input is asserted</p> <p>2 = insert hold latch using active low latch</p>
verif_en*	0 to 4 Default: 0	<p>Verification enable control</p> <p>0 = no sampling errors inserted</p> <p>1 = sampling errors are randomly inserted with 0 or up to 1 destination clock cycle delays</p> <p>2 = sampling errors are randomly inserted with 0, 0.5, 1, or 1.5 destination clock cycle delays</p> <p>3 = sampling errors are randomly inserted with 0, 1, 2, or 3 destination clock cycle delays</p> <p>4 = sampling errors are randomly inserted with 0 or up to 0.5 destination clock cycle delays</p> <p>* For more information about verif_en, see the Simulation Methodology section.</p>
send_mode	0 to 3 Default: 1	<p>0 = single clock cycle pulse in produces single clock cycle pulse out</p> <p>1 = rising edge transition in produces single clock cycle pulse out</p> <p>2 = falling edge transition in produces single clock cycle pulse out</p> <p>3 = rising &amp; falling transition each produce single clock cycle pulse out</p>

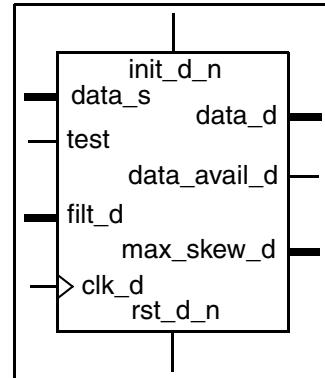
**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

**DW\_data\_sync\_1c****DW\_data\_sync\_1c**

Single Clock Filtered Data Bus Synchronizer

- ❖ Synchronizes on data without incoming clock source
- ❖ Parameterized data bus width
- ❖ Parameterized number of synchronizing stages
- ❖ Parameterized test feature
- ❖ All output registered
- ❖ Ability to model missampling of data on incoming clock domain

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data_s	width	Input	Source Domain data vector
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)
filt_d	filt_size	Input	Destination domain filter time specification
test	1	Input	Scan test mode select
data_avail_d	1	Output	Destination domain data update output
data_d	width	Output	Destination Domain data vector
max_skew_d	filt-size+1	Output	Destination domain maximum skew detected between bits for any data_s bus transition

**Table 1-2 Parameter Description**

Parameter	Values	Description
width	1 to 1024 Default: 8	Vector width of input data_s and output data_d
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
filt_size	1 to 8 Default: 1	filt_d vector size The width in bits for the filt_d vector input

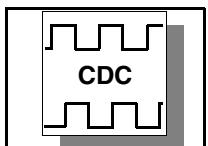
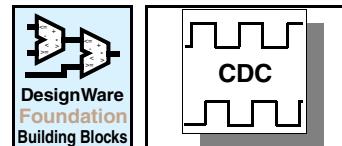


Table 1-2 Parameter Description

Parameter	Values	Description
tst_mode	0 or 1 Default: 0	Test Mode 0 = no 'latch' is inserted for scan testing 1 = insert negative-edge capturing flip-flop on data_s input vector when "test" input is asserted.
verif_en*	0 to 4 Default: 0	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 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 randomly inserted with 0 or up to 0.5 destination clock cycle delays * For more information about verif_en, see the Simulation Methodology section.

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

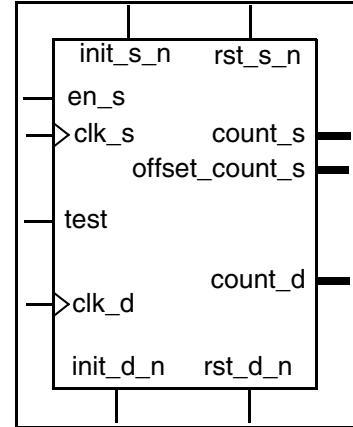
**DW\_gray\_sync**

Gray Coded Synchronizer

**DW\_gray\_sync**

Gray Coded Synchronizer

- ❖ 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

**Table 1-1 Pin Descriptions**

Pin Name	Width	Direction	Function
<code>clk_s</code>	1	Input	Source Domain clock source
<code>rst_s_n</code>	1	Input	Source Domain asynchronous reset (active low)
<code>init_s_n</code>	1	Input	Source Domain synchronous reset (active low)
<code>en_s</code>	1	Input	Source Domain counter advance initiation
<code>count_s</code>	width	Output	Source Domain counter value
<code>offset_count_s</code>	width	Output	Source Domain offset counter value
<code>clk_d</code>	1	Input	Destination Domain clock source
<code>rst_d_n</code>	1	Input	Destination Domain asynchronous reset (active low)
<code>init_d_n</code>	1	Input	Destination Domain synchronous reset (active low)
<code>count_d</code>	width	Output	Destination Domain counter value (synch'd version of <code>count_s</code> )
<code>test</code>	1	Input	Scan test mode select

**Table 1-2 Parameter Description**

Parameter	Values	Description
<code>width</code>	1 to 1024 Default: 8	Vector width of input <code>data_s</code> and output <code>data_d</code>
<code>offset</code>	0 to $(2^{width-1}) - 1$ Default: 0	Offset for non integer power of 2 counter value
<code>reg_count_d</code>	0 or 1 Default: 1	Register <code>count_d</code> output 0 = <code>count_d</code> not registered 1 = <code>count_d</code> registered

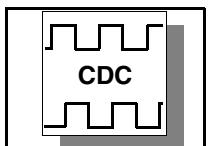
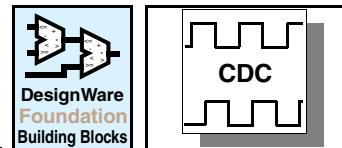


Table 1-2 Parameter Description

Parameter	Values	Description
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-flop on data_s input vector when "test" input is asserted. 2 = insert hold latch using active-low latch.
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

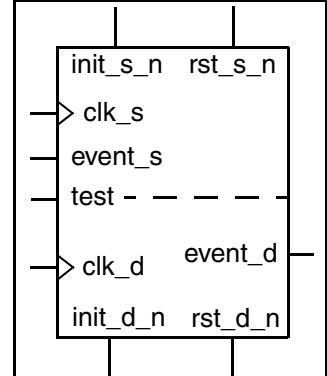
**DW\_pulse\_sync**

Dual Clock Pulse Synchronizer

**DW\_pulse\_sync**

Dual Clock Pulse Synchronizer

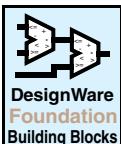
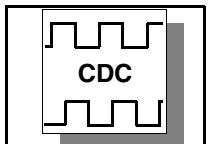
- ❖ Fully tested cross clock domain
- ❖ Fully parameterized
- ❖ Able to use both positive and negative clock edge for sending clock domain
- ❖ Provides for both combinatorial and registered output, via parameter

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
event_s	1	Input	Input pulse
clk_s	1	Input	Source clock
init_s_n	1	Input	Synchronous source reset
rst_s_n	1	Input	Asynchronous source reset
clk_d	1	Input	Destination clock
init_d_n	1	Input	Synchronous destination reset
rst_d_n	1	Input	Asynchronous destination reset
test	1	Input	Scan test mode select input
event_d	1	Output	Output pulse

**Table 1-2 Parameter Description**

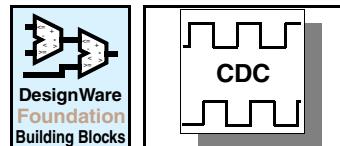
Parameter	Values	Description
reg_event	0 to 1 Default: 1	0 – no register on output 1 – register event_d output
f_sync_type	0 to 4 Default: 2	0 – single clock design clk_d = clk_s 1 – negedge to posedge sync 2 – posedge to posedge sync 3 – 3 posedge flops in dest domain 4 – 4 posedge flops in dest domain
tst_mode	0 to 2 Default: 0	0 – no test latch insertion 1 – hold latch using negedge flop 2 – hold latch using active low latch

**Table 1-2 Parameter Description**

Parameter	Values	Description
verif_en	0 to 4 Default: 1	Verify 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
pulse_mode	0 to 3 Default: 0	Selects the type of pulse presented to the input. 0 – single source domain clock cycle pulse transmitted to destination domain 1 – rising transition detect transmitted as single cycle pulse in the destination domain 2 – falling transition detect transmitted as single clock cycle pulse in the destination domain 3 – toggle of rising followed by falling separated by any number of clock cycles transmitted as a single-cycle pulse in the destination domain

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

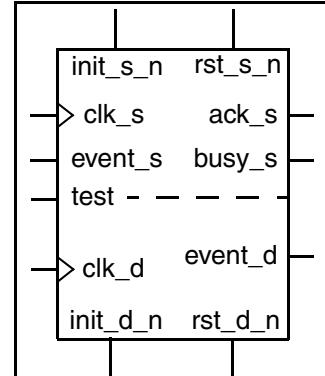
**DW\_pulseack\_sync**

Pulse Synchronizer with Acknowledge

**DW\_pulseack\_sync**

Pulse Synchronizer with Acknowledge

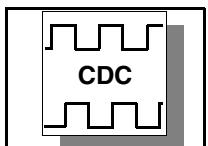
- ❖ Fully tested cross-clock domain
- ❖ Fully parameterized
- ❖ Able to use both positive and negative clock edge for sending clock domain
- ❖ Provides for both combinatorial and registered output, via parameter

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
clk_s	1	Input	Source clock
rst_s_n	1	Input	Asynchronous source reset
init_s_n	1	Input	Synchronous source reset
event_s	1	Input	Input pulse
busy_s	1	Output	busy_s is active high while the transmitted pulse and the acknowledge are in transit
ack_s	1	Output	ack_s is active high when the transmitted pulse has traveled to the destination domain and returned to the transmitting domain; active for one clock cycle
clk_d	1	Input	Destination clock
rst_d_n	1	Input	Asynchronous destination reset
init_d_n	1	Input	Synchronous destination reset
event_d	1	Output	Output pulse
test	1	Input	Scan test mode select input

**Table 1-2 Parameter Description**

Parameter	Values	Description
reg_event	0 to 1 Default: 1	0 – no register on output 1 – register event_d output
reg_ack	0 to 1 Default: 1	0 – ack_s has combination logic, but latency is 1 cycle sooner 1 – ack_s is retimed to eliminate combinational logic in the output; requires an additional clock cycle of delay

**DW\_pulseack\_sync**  
Pulse Synchronizer with Acknowledge**Table 1-2 Parameter Description**

Parameter	Values	Description
ack_delay	0 to 1 Default: 1	0 – ack_s has combination logic, but latency is 1 cycle sooner 1 – ack_s is retimed so there is no logic between register and port, but event is delayed 1 cycle
f_sync_type	0 to 4 Default: 2	0 – single clock design clk_d=clk_s 1 – negedge to posedge sync 2 – posedge to posedge sync 3 – 3 posedge registers in destination domain 4 – 4 posedge registers in destination domain
r_sync_type	0 to 4 Default: 2	0 – single clock design clk_d=clk_s 1 – negedge to posedge sync 2 – posedge to posedge sync 3 – 3 posedge registers in destination domain 4 – 4 posedge registers in destination domain
tst_mode	0 to 2 Default: 0	0 – no test latch insertion 1 – hold latch using negedge flop 2 – hold latch using active low latch
verif_en	0 to 4 Default: 1	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

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

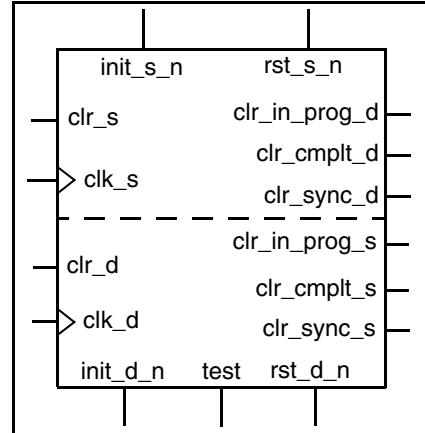
**DW\_reset\_sync**

Reset Sequence Synchronizer

**DW\_reset\_sync**

Reset Sequence Synchronizer

- ❖ Interface between dual asynchronous clock domains
- ❖ Coordinated clearing between clock domains
- ❖ Parameterized number of synchronizing stage
- ❖ All outputs registered (except "in progress" outputs which are configurable)
- ❖ Parameterized test feature
- ❖ Ability to model missampling of data on source clock domain

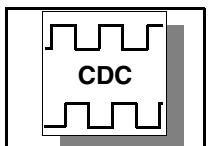


**Note** As of Release DWBB\_201109.1 (F-2011.09-SP1), the DW\_reset\_sync clearing behavior has been enhanced to lengthen the assertion of `clr_in_prog_d`. [Figure 1-2](#), [Figure 1-3](#), [Figure 1-4](#) and [Figure 1-5](#) have been updated to reflect the new behavior. In addition, [Figure 1-7](#) and [Figure 1-8](#) have been added to show an example of the difference in behavior.

The main purpose of enhancing the DW\_reset\_sync was to lengthen the `clr_in_prog_d` assertion. However, in doing so, the timing behavior of outputs `clr_in_prog_s` and `clr_cmplt_d` have also been slightly altered.

**Table 1-1 Pin Description**

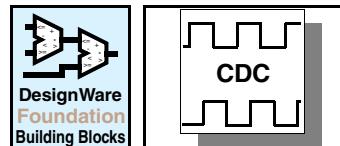
Pin Name	Width	Direction	Function
<code>clk_s</code>	1	Input	Source Domain clock source
<code>rst_s_n</code>	1	Input	Source Domain asynchronous reset (active low)
<code>init_s_n</code>	1	Input	Source Domain synchronous reset (active low)
<code>clr_s</code>	1	Input	Source Domain clear
<code>clr_sync_s</code>	1	Output	Source Domain clear for sequential logic (single <code>clk_s</code> cycle pulse)
<code>clr_in_prog_s</code>	1	Output	Source Domain clear sequence in progress
<code>clr_cmplt_s</code>	1	Output	Source Domain that clear sequence complete (single <code>clk_s</code> cycle pulse)
<code>clk_d</code>	1	Input	Destination Domain clock source
<code>rst_d_n</code>	1	Input	Destination Domain asynchronous reset (active low)
<code>init_d_n</code>	1	Input	Destination Domain synchronous reset (active low)
<code>clr_d</code>	1	Input	Destination Domain clear
<code>clr_in_prog_d</code>	1	Output	Destination Domain clear sequence in progress
<code>clr_sync_d</code>	1	Output	Destination Domain clear for sequential logic (single <code>clk_d</code> cycle pulse)
<code>clr_cmplt_d</code>	1	Output	Destination Domain that clear sequence complete (single <code>clk_d</code> cycle pulse)
<code>test</code>	1	Input	Scan test mode select input

**Table 1-2 Parameter Description**

Parameter	Values	Description
f_sync_type	0 to 4 Default: 2	Forward Synchronization Type Defines type and number of synchronizing stages: 0 – single clock design <code>clk_d = clk_s</code> 1 – negedge to posedge sync in destination domain 2 – posedge to posedge sync destination domain 3 – 3 posedge flops in destination domain 4 – 4 posedge flops in destination domain
r_sync_type	0 to 4 Default: 2	Reverse Synchronization Type (Destination to Source Domains) 0 – single clock design <code>clk_d = clk_s</code> 1 – negedge to posedge sync source domain 2 – posedge to posedge sync source domain 3 – 3 posedge flops in source domain 4 – 4 posedge flops in source domain
clk_d_faster	0 to 15 Default: 1	Obsolete parameter. The value setting is ignored. This parameter is kept in place only for backward compatibility.
reg_in_prog	0 or 1 Default: 1	Register the <code>clr_in_prog_s</code> and <code>clr_in_prog_d</code> Outputs 0 – unregistered 1 – registered
tst_mode	0 to 2 Default: 0	Test Mode 0 – no 'latch' is inserted for scan testing 1 – insert negative-edge capturing flip-flop on <code>data_s</code> input vector when <code>test</code> input is asserted. 2 – insert hold latch using active-low latch
verif_en*	0 to 2 Default: 1	Verification Enable Control 0 = no sampling errors inserted 1 = sampling errors randomly inserted with 0 or up to 1 destination clock cycle delays 2 = sampling errors randomly inserted with 0, 0.5, 1, or 1.5 destination clock cycle delays 3 = sampling errors randomly inserted with 0, 1, 2, or 3 destination clock cycle delays 4 = sampling errors randomly inserted with 0 or up to 0.5 destination clock cycle delays

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



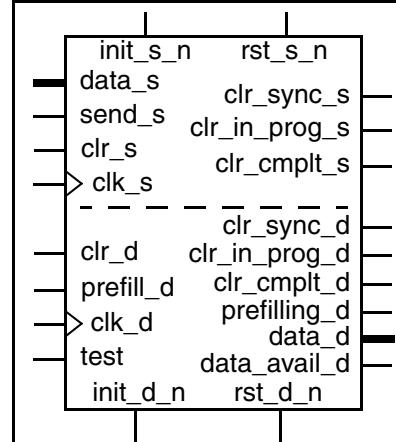
## DW\_stream\_sync

### Data Stream Synchronizer

# DW\_stream\_sync

## Data Stream Synchronizer

- ❖ Interface between dual asynchronous clock domains
- ❖ Coordinated clearing between clock domains
- ❖ Parameterized data bus width
- ❖ Parameterized number of synchronizing stages
- ❖ Parameterized test feature
- ❖ All data-related outputs registered
- ❖ Ability to model missampling of data on source clock domain



### Note

As of DesignWare versions F-2011.09-SP1 and later, the underlying component DW\_reset\_sync was enhanced to improve the clearing sequence. As a result, [Figure 1-3](#), [Figure 1-3](#) and [Figure 1-3](#) of this datasheet have been updated to reflect the change of behavior of the clearing sequence. This enhancement of DW\_reset\_sync does not impact the basic function of DW\_stream\_sync with respect to the streaming of data before, during, and after the clearing sequence. For more details about the changes to DW\_reset\_sync refer to the [DW\\_reset\\_sync](#) datasheet or Technical Bulletin "XXXXXX".

**Table 1-1 Pin Description**

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)
clr_s	1	Input	Source Domain clear
send_s	1	Input	Source initiate valid data vector control
data_s	width	Input	Source Domain data vector
clr_sync_s	1	Output	Source Domain clear for sequential logic
clr_in_prog_s	1	Output	Source Domain clear sequence in progress
clr_cmplt_s	1	Output	Source Domain that clear sequence complete (single clk_s cycle pulse)
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)
clr_d	1	Input	Destination Domain clear
prefill_d	1	Input	Destination Domain prefill control

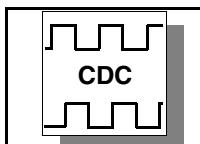
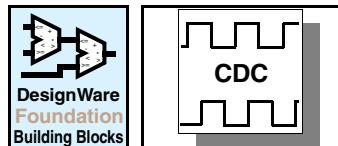


Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clr_in_prog_d	1	Output	Destination Domain clear sequence in progress
clr_sync_d	1	Output	Destination Domain clear for sequential logic
clr_cmplt_d	1	Output	Destination Domain that clear sequence complete (single clk_d cycle pulse)
data_avail_d	1	Output	Destination Domain data update
data_d	width	Output	Destination Domain data vector
prefilling_d	1	Output	Destination Domain prefilling FIFO in progress
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
depth	1 to 256	Depth of FIFO
prefill_lvl	0 to depth-1 Default: 0	The number of valid entries in the FIFO before transferring packets to destination (enabled when prefill_d asserted)
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 first stage negative-edge capturing and second 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
reg_stat	0 or 1 Default: 1	Registering Internal Status (affects prefilling_d) 0 = don't register internally calculated status (prefilling_d appears one cycle sooner than if reg_stat = 1) 1 = register internally calculated status
tst_mode	0 to 2 Default: 0	Test Mode 0 = no latch is inserted for scan testing 1 = insert negative-edge capturing register on data_s input vector when test input is asserted 2 = insert hold latch using active low latch



## DW\_stream\_sync

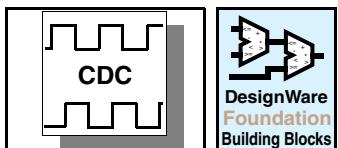
### Data Stream Synchronizer

**Table 1-2 Parameter Description**

Parameter	Values	Description
verif_en	0 to 4 Default: 2	Verification Enable Control 0 = no sampling errors inserted 1 = sampling errors randomly inserted with 0 or up to 1 destination clock cycle delays 2 = sampling errors randomly inserted with 0, 0.5, 1, or 1.5 destination clock cycle delays 3 = sampling errors randomly inserted with 0, 1, 2, or 3 destination clock cycle delays 4 = sampling errors randomly inserted with 0 or up to 0.5 destination clock cycle delays For more information, see the Simulation Methodology section.
r_sync_type	0 to 4 Default: 2	Reverse Synchronization Type (Destination to Source Domains) 0 = no synchronization, single clock system 1 = 2-stage synchronization w/ 1st stage negative-edge & 2nd stage positive-edge capturing 2 = 2-stage synchronization w/ both stages positive-edge capturing 3 = 3-stage synchronization w/ all stages positive-edge capturing 4 = 4-stage synchronization w/ all stages positive-edge capturing
clk_d_faster	0 to 15 Default: 1	Obsolete parameter. The value setting is ignored. This parameter is kept in place only for backward compatibility.
reg_in_prog	0 or 1 Default: 1	Register the <code>clr_in_prog_s</code> and <code>clr_in_prog_d</code> Outputs 0 = unregistered 1 = registered

**Table 1-3 Synthesis Implementations**

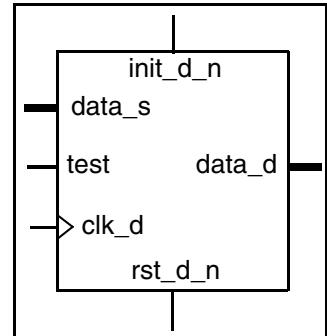
Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



## DW\_sync

### Single Clock Data Bus Synchronizer

- ❖ Parameterized data bus width
- ❖ Parameterized number of synchronizing stages
- ❖ Parameterized test feature
- ❖ Ability to model missampling of data on incoming clock domain



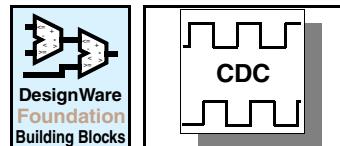
DWbb\_bcm21DWbb\_bcm21DWbb\_bcm21DWbb\_bcm21 module port and parameter behavior can be affected by 'define Macro application (as shown highlighted in the tables below.

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
data_s	width	Input	Source Domain data vector
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)
test	1	Input	Scan test mode select
data_d	width	Output	Destination Domain data vector

**Table 1-2 Parameter Description**

Parameter	Values	Description
WIDTH	1 to 1024 Default: 8	Vector width of input data_s and output data_d
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

**DW\_sync**

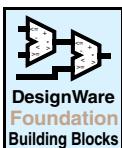
Single Clock Data Bus Synchronizer

**Table 1-2 Parameter Description**

Parameter	Values	Description
TST_MODE	0 to 2 Default: 0	<p>Test Mode</p> <p>0 = no 'latch' is inserted for scan testing</p> <p>1 = insert negative-edge flip-flop on data_s input vector when test input is asserted</p> <p>2 = for compatibility when latching is done outside of DW_sync (functions as with test_mode=0)</p>
VERIF_EN	0 to 4 Default: 1	<p>Verification Enable Control</p> <p>0 = no sampling errors inserted</p> <p>1 = sampling errors are randomly inserted with 0 or up to 1 destination clock cycle delay</p> <p>2 = sampling errors are randomly inserted with 0, 0.5, 1, or 1.5 destination clock cycle delays</p> <p>3 = sampling errors are randomly inserted with 0, 1, 2, or 3 destination clock cycle delays</p> <p>4 = sampling errors randomly inserted with 0 or up to 0.5 destination clock cycle delays</p> <p>* For more information about verif_en, refer to "<a href="#">"Simulation Methodology" on page 3.</a></p>

**Table 1-3 Synthesis Implementations**

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare



## Test – JTAG Overview

The JTAG IP consist of a set of boundary scan IP. The boundary scan IP include a parameterized Test Access Port (TAP) controller (DW\_tap) plus a set of boundary scan cells that you can use to implement a custom IEEE 1149.1 boundary scan test solution for your ASIC.

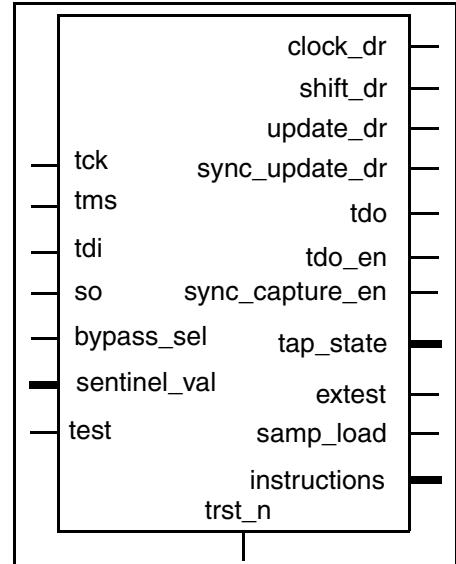
**DW\_tap**

TAP Controller

**DW\_tap**

TAP Controller

- ❖ IEEE Standard 1149.1 compliant
- ❖ Synchronous or asynchronous registers with respect to tck
- ❖ Supports the standard instructions EXTEST, SAMPLE/PRELOAD, and BYPASS
- ❖ Supports the optional instructions IDCODE, INTEST, RUNBIST, CLAMP, and HIGHZ
- ❖ Optional use of device identification register and IDCODE instruction
- ❖ Parameterized instruction register width

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
tck	1 bit	Input	Test clock
trst_n	1 bit	Input	Test reset, active low
tms	1 bit	Input	Test mode select
tdi	1 bit	Input	Test data in
so	1 bit	Input	Serial data from boundary scan register and data registers
bypass_sel	1 bit	Input	Selects the bypass register, active high
sentinel_val	width –1 bit(s)	Input	User-defined status bits
clock_dr	1 bit	Output	Clocks in data in asynchronous mode
shift_dr	1 bit	Output	Enables shifting of data in both synchronous and asynchronous mode, active high
update_dr	1 bit	Output	Enables updating data in asynchronous mode, active high
tdo	1 bit	Output	Test data out
tdo_en	1 bit	Output	Enable for tdo output buffer, active high
tap_state	16 bits	Output	Current state of the TAP finite state machine
extest	1 bit	Output	EXTEST decoded instruction
samp_load	1 bit	Output	SAMPLE/PRELOAD decoded instruction
instructions	width bit(s)	Output	Instruction register output
sync_capture_en	1 bit	Output	Enable for synchronous capture, active low

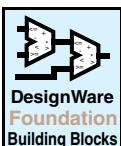


Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
sync_update_dr	1 bit	Output	Enables updating new data in synchronous_mode, active high
test	1 bit	Input	For scannable designs, the test_mode pin is held active (HIGH) during testing. For normal operation, it is held inactive (LOW).

Table 1-2 Parameter Description

Parameter	Values	Description
width	2 to 32 Default: None	Width of instruction register
id	0 or 1 Default: 0	Determines whether the device identification register is present 0 = not present, 1 = present
version	0 to 15 Default: 0	4-bit version number
part	0 to 65535 Default: 0	16-bit part number
man_num	0 to 2047, man_num ≠ 127 Default: 0	11-bit JEDEC manufacturer identity code
sync_mode	0 or 1 Default: 0	Determines whether the bypass, device identification, and instruction registers are synchronous with respect to tck 0 = asynchronous, 1 = synchronous

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1

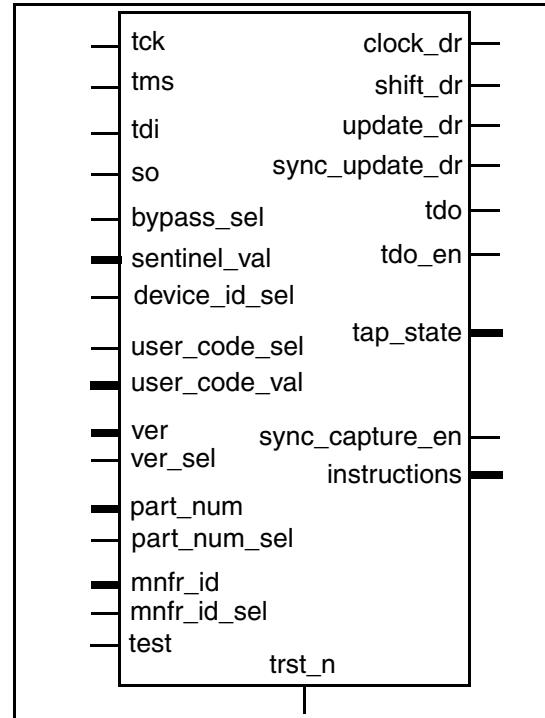
**DW\_tap\_uc**

TAP Controller with USERCODE support

**DW\_tap\_uc**

TAP Controller with USERCODE support

- ❖ IEEE Standard 1149.1 compliant
- ❖ Synchronous or asynchronous registers with respect to tck
- ❖ Provides interface to supports the standard IEEE 1149.1 and optional instructions
- ❖ Optional use of device identification register and IDCODE instruction and support of USERCODE instruction
- ❖ User defined opcode for IDCODE
- ❖ Parameterized instruction register width
- ❖ External interface to program device identification register

**Table 1-1 Pin Description**

<b>Pin Name</b>	<b>Width</b>	<b>Direction</b>	<b>Function</b>
tck	1 bit	Input	Test clock
trst_n	1 bit	Input	Test reset, active low
tms	1 bit	Input	Test mode select
tdi	1 bit	Input	Test data in
so	1 bit	Input	Serial data from boundary scan register and data registers
bypass_sel	1 bit	Input	Selects the bypass register, active high
sentinel_val	width - 1 bit(s)	Input	User-defined status bits
device_id_sel	1 bit	Input	Selects the device identification register, active high
user_code_sel	1 bit	Input	Selects the user_code_val bus for input in to the device identification register, active high
user_code_val	32 bits	Input	32-bit user defined code
ver	4 bits	Input	4 bit version number
ver_sel	1 bit	Input	Selects version from the parameter or the ver input port 0 = version (parameter) 1 = ver (input port)
part_num	16 bits	Input	16 bit part number

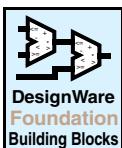


Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
part_num_sel	1 bit	Input	Selects part from the parameter or the part_num from the input port 0 = part (parameter) 1 = part_num (input port)
mnfr_id	11 bits	Input	11 bit JEDEC manufacturer's identity code (mnfr_id ≠ 127)
mnfr_id_sel	1 bit	Input	Selects man_num from the parameter or mnfr_id from the input port 0 = man_num (parameter) 1 = mnfr_id (input port)
test	1 bit	Input	For scannable designs, the test_mode pin is held active (HIGH) during testing. For normal operation, it is held inactive (LOW).
clock_dr	1 bit	Output	Clocks in data in asynchronous mode
shift_dr	1 bit	Output	Enables shifting of data in both synchronous and asynchronous mode, active high
update_dr	1 bit	Output	Enables updating data in asynchronous mode, active high
tdo	1 bit	Output	Test data out
tdo_en	1 bit	Output	Enable for tdo output buffer, active high
tap_state	16 bits	Output	Current state of the TAP finite state machine
instructions	width bit(s)	Output	Instruction register output
sync_capture_en	1 bit	Output	Enable for synchronous capture, active low
sync_update_dr	1 bit	Output	Enables updating new data in synchronous_mode, active high

Table 1-2 Parameter Description

Parameter	Values	Description
width	2 to 32 Default: None	Width of instruction register
id	0 or 1 Default: 0	Determines whether the device identification register is present 0 = not present, 1 = present
idcode_opcode	1 to $2^{\text{width}-1}$ Default: 1	Opcode for IDCODE.
version	0 to 15 Default: 0	4-bit version number
part	0 to 65535 Default: 0	16-bit part number
man_num	0 to 2047, man_num ≠ 127 Default: 0	11-bit JEDEC manufacturer identity code

**DW\_tap\_uc**

TAP Controller with USERCODE support

**Table 1-2 Parameter Description (Continued)**

Parameter	Values	Description
sync_mode	0 or 1 Default: 0	Determines whether the bypass, device identification, and instruction registers are synchronous with respect to tck 0 = asynchronous, 1 = synchronous

**Table 1-3 Synthesis Implementations**

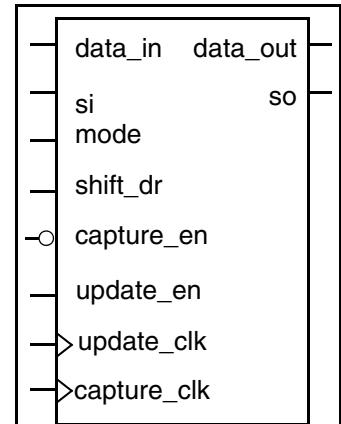
Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1



## DW\_bc\_1

### Boundary Scan Cell Type BC\_1

- ❖ IEEE Standard 1149.1 compliant
- ❖ Synchronous or asynchronous scan cells with respect to tck
- ❖ Supports the standard instructions: EXTEST, SAMPLE/PRELOAD, and BYPASS
- ❖ Supports the optional instructions INTEST, RUNBIST, CLAMP, and HIGHZ

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
capture_clk	1 bit	Input	Clocks data into the capture stage
update_clk	1 bit	Input	Clocks data into the update stage
capture_en	1 bit	Input	Enable for data clocked into the capture stage, active low
update_en	1 bit	Input	Enable for data clocked into the update stage, active high
shift_dr	1 bit	Input	Enables the boundary scan chain to shift data one stage toward its serial output (tdo)
mode	1 bit	Input	Determines whether data_out is controlled by the boundary scan cell or by the data_in signal
si	1 bit	Input	Serial path from the previous boundary scan cell
data_in	1 bit	Input	Input data
data_out	1 bit	Output	Output data
so	1 bit	Output	Serial path to the next boundary scan cell

**Table 1-2 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1

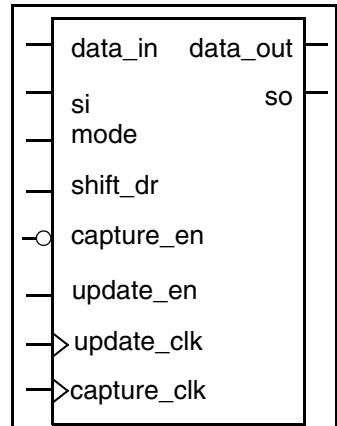
**DW\_bc\_2**

Boundary Scan Cell Type BC\_2

**DW\_bc\_2**

Boundary Scan Cell Type BC\_2

- ❖ IEEE Standard 1149.1 compliant
- ❖ Synchronous or asynchronous scan cells with respect to tck
- ❖ Supports the standard instructions: EXTEST, SAMPLE/PRELOAD, and BYPASS
- ❖ Supports the optional instructions INTEST, RUNBIST, CLAMP, and HIGHZ

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
capture_clk	1 bit	Input	Clocks data into the capture stage
update_clk	1 bit	Input	Clocks data into the update stage
capture_en	1 bit	Input	Enable for data clocked into the capture stage, active low
update_en	1 bit	Input	Enable for data clocked into the update stage, active high
shift_dr	1 bit	Input	Enables the boundary scan chain to shift data one stage toward its serial output (tdo)
mode	1 bit	Input	Determines whether data_out is controlled by the boundary scan cell or by the data_in signal
si	1 bit	Input	Serial path from the previous boundary scan cell
data_in	1 bit	Input	Input data
data_out	1 bit	Output	Output data
so	1 bit	Output	Serial path to the next boundary scan cell

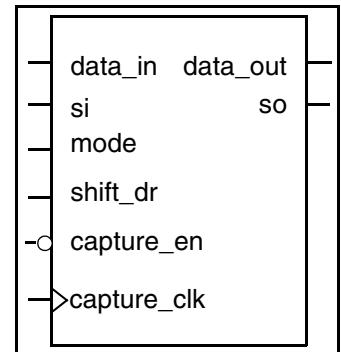
**Table 1-2 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1

## DW\_bc\_3

### Boundary Scan Cell Type BC\_3

- ❖ IEEE Standard 1149.1 compliant
- ❖ Synchronous or asynchronous scan cells with respect to tck
- ❖ Supports the standard instructions: EXTEST, SAMPLE/PRELOAD, and BYPASS
- ❖ Supports the optional instructions INTEST, RUNBIST, CLAMP, and HIGHZ



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
capture_clk	1 bit	Input	Clocks data into the capture stage
capture_en	1 bit	Input	Enable for data clocked into capture stage, active low
shift_dr	1 bit	Input	Enables the boundary scan chain to shift data one stage toward its serial output (tdo)
mode	1 bit	Input	Determines whether data_out is controlled by the boundary scan cell or by the data_in signal
si	1 bit	Input	Serial path from the previous boundary scan cell
data_in	1 bit	Input	Input data from system input pin
data_out	1 bit	Output	Output data to IC logic
so	1 bit	Output	Serial path to the next boundary scan cell

**Table 1-2 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1

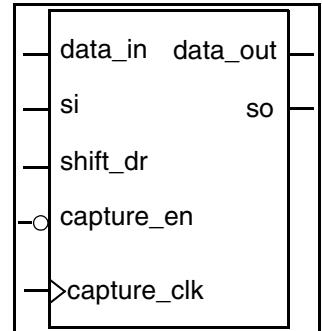
**DW\_bc\_4**

Boundary Scan Cell Type BC\_4

**DW\_bc\_4**

Boundary Scan Cell Type BC\_4

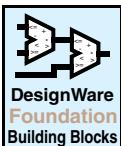
- ❖ IEEE Standard 1149.1 compliant
- ❖ Synchronous or asynchronous scan cells with respect to tck
- ❖ Supports the standard instructions: EXTEST, SAMPLE/PRELOAD, and BYPASS

**Table 1-1 Pin Description**

<b>Pin Name</b>	<b>Width</b>	<b>Direction</b>	<b>Function</b>
capture_clk	1 bit	Input	Clocks data into the capture stage
capture_en	1 bit	Input	Enable for data clocked into the capture stage, active low
shift_dr	1 bit	Input	Enables the boundary scan chain to shift data one stage toward its serial output (tdo)
si	1 bit	Input	Serial path from the previous boundary scan cell
data_in	1 bit	Input	Input data from system input pin
so	1 bit	Output	Serial path to the next boundary scan cell
data_out	1 bit	Output	Output data

**Table 1-2 Synthesis Implementations**

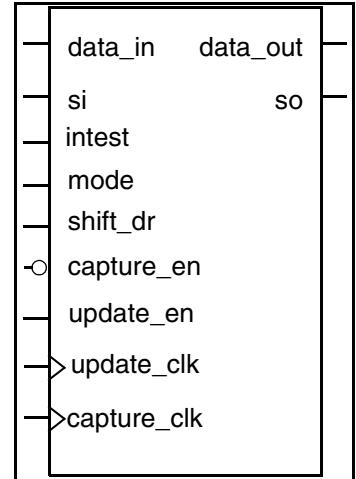
<b>Implementation Name</b>	<b>Function</b>	<b>License Feature Required</b>
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1



## DW\_bc\_5

### Boundary Scan Cell Type BC\_5

- ❖ IEEE Standard 1149.1 compliant
- ❖ Synchronous or asynchronous scan cells with respect to tck
- ❖ Supports the standard instructions: EXTEST, SAMPLE/PRELOAD, and BYPASS
- ❖ Supports the optional instructions INTEST, RUNBIST, CLAMP, and HIGHZ

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
capture_clk	1 bit	Input	Clocks data into the capture stage
update_clk	1 bit	Input	Clocks data into the update stage
capture_en	1 bit	Input	Enable for data clocked into the capture stage, active low
update_en	1 bit	Input	Enable for data clocked into the update stage, active high
shift_dr	1 bit	Input	Enables the boundary scan chain to shift data one stage toward its serial output (tdo)
mode	1 bit	Input	Determines whether data_out is controlled by the boundary scan cell or by the data_in signal
intest	1 bit	Input	INTEST instruction signal
si	1 bit	Input	Serial path from the previous boundary scan cell
data_in	1 bit	Input	Input data from system input pin
data_out	1 bit	Output	Output data
so	1 bit	Output	Serial path to the next boundary scan cell

**Table 1-2 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1

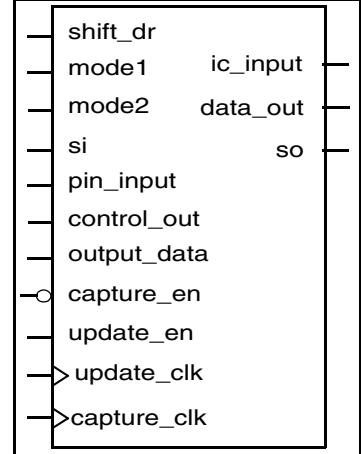
**DW\_bc\_7**

Boundary Scan Cell Type BC\_7

**DW\_bc\_7**

Boundary Scan Cell Type BC\_7

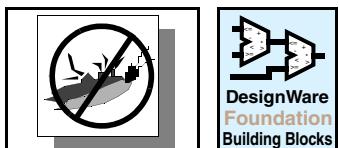
- ❖ IEEE Standard 1149.1 compliant
- ❖ Synchronous or asynchronous scan cells with respect to tck
- ❖ Supports the standard instructions: EXTEST, SAMPLE/PRELOAD, and BYPASS
- ❖ Supports the optional instructions INTEST, RUNBIST, CLAMP, and HIGHZ

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
capture_clk	1 bit	Input	Clocks data into the capture stage
update_clk	1 bit	Input	Clocks data into the update stage
capture_en	1 bit	Input	Enable for data clocked into the capture stage, active low
update_en	1 bit	Input	Enable for data clocked into the update stage, active high
shift_dr	1 bit	Input	Enables the boundary scan chain to shift data one stage toward its serial output (tdo)
mode1	1 bit	Input	Determines whether data_out is controlled by the boundary scan cell or by the output_data signal
mode2	1 bit	Input	Determines whether ic_input is controlled by the boundary scan cell or by the pin_input signal
si	1 bit	Input	Serial path from the previous boundary scan cell
pin_input	1 bit	Input	IC system input pin
control_out	1 bit	Input	Control signal for the output enable
output_data	1 bit	Input	IC output logic signal
ic_input	1 bit	Output	IC input logic signal
data_out	1 bit	Output	Output data
so	1 bit	Output	Serial path to the next boundary scan cell

**Table 1-2 Synthesis Implementations**

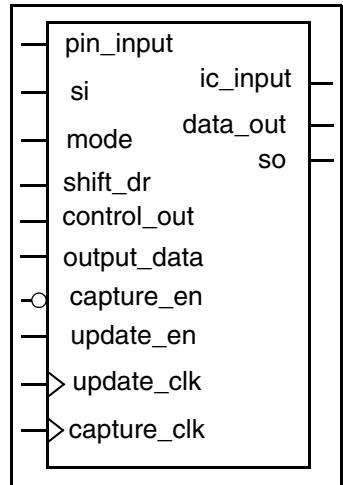
Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1



## DW\_bc\_8

### Boundary Scan Cell Type BC\_8

- IEEE Standard 1149.1-2001 compliant
- Synchronous or asynchronous scan cells with respect to tck
- Supports the standard instructions: EXTEST, SAMPLE/PRELOAD, and BYPASS
- Supports the optional instructions RUNBIST, CLAMP, and HIGHZ



**Table 1-1 Pin Description**

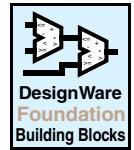
Pin Name	Width	Direction	Function
capture_clk	1 bit	Input	Clocks data into the capture stage
update_clk	1 bit	Input	Clocks data into the update stage
capture_en	1 bit	Input	Enable for data clocked into the capture stage, active low
update_en	1 bit	Input	Enable for data clocked into the update stage, active high
shift_dr	1 bit	Input	Enables the boundary scan chain to shift data one stage toward its serial output (tdo)
mode	1 bit	Input	Determines whether data_out is controlled by the boundary scan cell or by the data_in signal
si	1 bit	Input	Serial path from the previous boundary scan cell
pin_input	1 bit	Input	IC system input pin
output_data	1 bit	Input	IC output logic signal
ic_input	1 bit	Output	Connected to IC input logic
data_out	1 bit	Output	Output data
so	1 bit	Output	Serial path to the next boundary scan cell

**Table 1-2 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare-Foundation or Test-IEEE-STD-1149-1

**DW\_bc\_8**

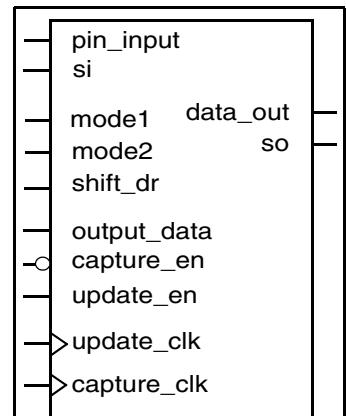
Boundary Scan Cell Type BC\_8



# DW\_bc\_9

## Boundary Scan Cell Type BC\_9

- ❖ IEEE Standard 1149.1-2001 compliant
- ❖ Synchronous or asynchronous scan cells with respect to tck
- ❖ Supports the standard instructions: EXTEST, INTEST, SAMPLE/PRELOAD, and BYPASS
- ❖ Supports the optional instructions RUNBIST, CLAMP, and HIGHZ



**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
capture_clk	1 bit	Input	Clocks data into the capture stage
update_clk	1 bit	Input	Clocks data into the update stage
capture_en	1 bit	Input	Enable for data clocked into the capture stage, active low
update_en	1 bit	Input	Enable for data clocked into the update stage, active high
shift_dr	1 bit	Input	Enables the boundary scan chain to shift data one stage toward its serial output (tdo)
mode1	1 bit	Input	Determines whether data_out is controlled by the boundary scan cell or by the data_in signal
mode2	1 bit	Input	Determines whether data_out is controlled by the boundary scan cell or by the data_in signal
si	1 bit	Input	Serial path from the previous boundary scan cell
pin_input	1 bit	Input	IC system input pin
output_data	1 bit	Input	IC output logic signal
data_out	1 bit	Output	Output data
so	1 bit	Output	Serial path to the next boundary scan cell

**Table 1-2 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1

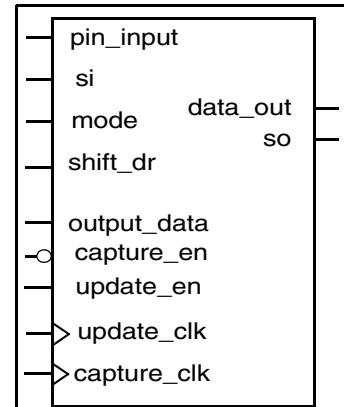
**DW\_bc\_10**

Boundary Scan Cell Type BC\_10

**DW\_bc\_10**

Boundary Scan Cell Type BC\_10

- ❖ IEEE Standard 1149.1-2001 compliant
- ❖ Synchronous or asynchronous scan cells with respect to tck
- ❖ Supports the standard instructions: EXTEST, SAMPLE/PRELOAD, and BYPASS
- ❖ Supports the optional instructions RUNBIST, CLAMP, and HIGHZ

**Table 1-1 Pin Description**

Pin Name	Width	Direction	Function
capture_clk	1 bit	Input	Clocks data into the capture stage
update_clk	1 bit	Input	Clocks data into the update stage
capture_en	1 bit	Input	Enable for data clocked into the capture stage, active low
update_en	1 bit	Input	Enable for data clocked into the update stage, active high
shift_dr	1 bit	Input	Enables the boundary scan chain to shift data one stage toward its serial output (tdo)
mode	1 bit	Input	Determines whether data_out is controlled by the boundary scan cell or by the data_in signal
si	1 bit	Input	Serial path from the previous boundary scan cell
pin_input	1 bit	Input	IC system input pin
output_data	1 bit	Input	IC output logic signal
data_out	1 bit	Output	Output data
so	1 bit	Output	Serial path to the next boundary scan cell

**Table 1-2 Synthesis Implementations**

Implementation Name	Function	License Feature Required
str	Synthesis model	DesignWare or Test-IEEE-STD-1149-1



# GTECH Library Overview

Synopsys provides the GTECH technology-independent library to aid users in developing technology-independent parts. Also, DesignWare IP often use these cells for their implementation. This generic technology library, called gtech.db, contains common logic elements. gtech.db can be found under the Synopsys root directory in libraries/syn. Simulation models are located under the Synopsys root directory in packages/gtech/src (VHDL) and packages/gtech/src\_ver (Verilog).

For more information about the GTECH IP, refer to the [DesignWare GTECH Libraries Databook](#).

# Index

---

## Symbols

\$SYNOPSYS/dw/examples 17

## Numerics

8B/10B coding 164

## A

arbiter 19

ASIC dual port RAM 205

Asynchronous RAM 269

## B

Block IP for DC 17

boundary scan cells 305

Building Block IP

application specific - control logic 19

data integrity 156

data integrity - coding overview 164

datapath - arithmetic overview 31

datapath - floating point overview 101

datapath - sequential 132

datapath - trigonometric overview 139

datapath functions overview 141

datapath generator overview 30

DSP 170

interface - clock domain crossing 281

logic - combinational overview 184

logic - sequential overview 194

memory - asynchronous RAMs 269

memory - FIFO overview 205

memory - registers 251

memory - stacks 276

overview 14, 15

test - JTAG overview 305

Building Block IP in DC-TCL 17

## C

carry look-ahead 139

CDC 281

Clock Domain Crossing 281

Clock Domain Crossing (CDC) 16

Coding Group 164

Combinational 184

Comments? 13

Compiler directives 17

Control Logic IP 19

## D

D flip-flop-based 205

Data Integrity 16

data integrity 156

Datapath 16

Datapath - Sequential IP 132

datapath functions, overview 141

datapath generator, overview 30

datapath synthesis flow 30

DC Ultra 30

DCT and iDCT 16

Design Compiler 17

DesignWare Building Block IP, *See also* Building Block IP

DesignWare Cores 14, 15

DesignWare GTECH Library 321

DesignWare IP Family, overview 14, 15

DesignWare Library 14, 15

DesignWare Library Synthesizable IP 16  
Building Block IP 16

Digital Signal Processing 170

Digital Signal Processing (DSP) 16

Doc on the Web 13

DSP 170

dual-port RAM [205](#)  
 DW\_8b10b\_dec [165](#)  
 DW\_8b10b\_enc [167](#)  
 DW\_8b10b\_unbal [169](#)  
 DW\_addsub\_dx [36](#)  
 DW\_arb\_2t [20](#)  
 DW\_arb\_dp [19, 22](#)  
 DW\_arb\_fcfs [24](#)  
 DW\_arb\_rr [26](#)  
 DW\_arb\_sp [19, 28](#)  
 DW\_asymdata\_inbuf [206](#)  
 DW\_asymfifo\_s1\_df [208, 210](#)  
 DW\_asymfifo\_s1\_sf [213](#)  
 DW\_asymfifo\_s2\_sf [216](#)  
 DW\_asymfifoctl\_s1\_df [229](#)  
 DW\_asymfifoctl\_s1\_sf [232](#)  
 DW\_asymfifoctl\_s2\_sf [235](#)  
 DW\_bc\_1 [311](#)  
 DW\_bc\_10 [320](#)  
 DW\_bc\_2 [312](#)  
 DW\_bc\_3 [313](#)  
 DW\_bc\_4 [314](#)  
 DW\_bc\_5 [315](#)  
 DW\_bc\_7 [316](#)  
 DW\_bc\_8 [317](#)  
 DW\_bc\_9 [319](#)  
 DW\_bin2gray [39](#)  
 DW\_cmp\_dx [44](#)  
 DW\_cntr\_gray [45](#)  
 DW\_crc\_p [157](#)  
 DW\_crc\_s [159](#)  
 DW\_data\_qsync\_hl [282](#)  
 DW\_data\_qsync\_lh [284](#)  
 DW\_data\_sync [282, 286](#)  
 DW\_data\_sync\_1c [290](#)  
 DW\_data\_sync\_na [288](#)  
 DW\_dct\_2d [171](#)  
 DW\_decode\_en [174](#)  
 DW\_div [48, 52](#)  
 DW\_div\_pipe [50](#)  
 DW\_div\_seq [133](#)  
 DW\_dpll\_sd [198](#)  
 DW\_ecc [161](#)  
 DW\_exp2 [53](#)  
 DW\_fifo\_2c\_df [219](#)  
 DW\_fifo\_s1\_df [222](#)  
 DW\_fifo\_s1\_sf [224](#)  
 DW\_fifo\_s2\_sf [226](#)  
 DW\_fifoctl\_2c\_df [239](#)  
 DW\_fifoctl\_s1\_df [244](#)  
 DW\_fifoctl\_s1\_sf [246](#)  
 DW\_fifoctl\_s2\_sf [248](#)  
 DW\_fir [175](#)  
 DW\_fir\_seq [177](#)  
 DW\_fp\_add [102, 123, 125, 126](#)  
 DW\_fp\_addsub [103](#)  
 DW\_fp\_cmp [105](#)  
 DW\_fp\_div [107](#)  
 DW\_fp\_div\_seq [109](#)  
 DW\_fp\_dp2 [110](#)  
 DW\_fp\_dp3 [112](#)  
 DW\_fp\_dp4 [114](#)  
 DW\_fp\_exp [116](#)  
 DW\_fp\_exp2 [117](#)  
 DW\_fp\_flt2i [118](#)  
 DW\_fp\_i2flt [119](#)  
 DW\_fp\_invsqrt [120](#)  
 DW\_fp\_ln [121](#)  
 DW\_fp\_log2 [122](#)  
 DW\_fp\_mac [123](#)  
 DW\_fp\_mult [124](#)  
 DW\_fp\_recip [125](#)  
 DW\_fp\_sincos [126](#)  
 DW\_fp\_sqrt [127](#)  
 DW\_fp\_square [128](#)  
 DW\_fp\_sub [129](#)  
 DW\_fp\_sum3 [130](#)  
 DW\_fp\_sum4 [131](#)  
 DW\_gray\_sync [292](#)  
 DW\_gray2bin [54](#)  
 DW\_iir\_dc [179](#)  
 DW\_iir\_sc [181](#)  
 DW\_inc\_gray [57](#)  
 DW\_inv\_sqrt [58](#)  
 DW\_lbsh [59](#)  
 DW\_ln [60](#)

DW\_lod 187  
 DW\_log2 61  
 DW\_lsd 188  
 DW\_lza 189  
 DW\_lzd 189, 190  
 DW\_minmax 63  
 DW\_mult\_dx 73  
 DW\_mult\_pipe 74  
 DW\_mult\_seq 135  
 DW\_norm 76  
 DW\_norm\_rnd 78  
 DW\_piped\_mac 80  
 DW\_pl\_reg 253  
 DW\_pricode 192  
 DW\_prod\_sum\_pipe 84  
 DW\_pulse\_sync 294  
 DW\_pulseack\_sync 296  
 DW\_ram\_2r\_w\_a\_dff 272  
 DW\_ram\_2r\_w\_a\_lat 273  
 DW\_ram\_2r\_w\_s\_dff 262, 264  
 DW\_ram\_2r\_w\_s\_lat 266  
 DW\_ram\_r\_w\_a\_dff 270  
 DW\_ram\_r\_w\_a\_lat 271  
 DW\_ram\_r\_w\_s\_dff 260  
 DW\_ram\_r\_w\_s\_lat 261  
 DW\_ram\_rw\_a\_dff 274  
 DW\_ram\_rw\_a\_lat 275  
 DW\_ram\_rw\_s\_dff 267  
 DW\_ram\_rw\_s\_lat 268  
 DW\_rash 86  
 DW\_rbsh 87  
 DW\_reset\_sync 298  
 DW\_shifter 89  
 DW\_sincos 140  
 DW\_sla 91  
 DW\_sqrt 96  
 DW\_sqrt\_pipe 97  
 DW\_sqrt\_seq 137  
 DW\_square 93  
 DW\_squarep 94  
 DW\_sra 92  
 DW\_stack 277  
 DW\_stackctl 279  
 DW\_stream\_sync 300  
 DW\_sync 303  
 DW\_tap 306  
 DW\_tap\_uc 308  
 DW\_thermdec 183  
 DW01\_absval 32  
 DW01\_add 33  
 DW01\_addsub 34  
 DW01\_ash 38  
 DW01\_binenc 185  
 DW01\_bsh 40  
 DW01\_cmp2 41  
 DW01\_cmp6 42  
 DW01\_csa 46  
 DW01\_dec 47  
 DW01\_decode 186  
 DW01\_inc 55  
 DW01\_incdec 56  
 DW01\_mux\_any 191  
 DW01\_prienc 193  
 DW01\_satrnd 88  
 DW01\_sub 98  
 DW02\_mac 62  
 DW02\_mult 64  
 DW02\_mult\_2\_stage 68  
 DW02\_mult\_3\_stage 69  
 DW02\_mult\_4\_stage 70  
 DW02\_mult\_5\_stage 71  
 DW02\_mult\_6\_stage 72  
 DW02\_multp 66  
 DW02\_prod\_sum 82  
 DW02\_prod\_sum1 83  
 DW02\_sum 99  
 DW02\_tree 100  
 DW03\_bictr\_dcnto 195  
 DW03\_bictr\_decode 197  
 DW03\_bictr\_scnto 196  
 DW03\_lfsr\_dcnto 200  
 DW03\_lfsr\_load 202  
 DW03\_lfsr\_scnto 201  
 DW03\_lfsr\_updn 203  
 DW03\_pipe\_reg 252  
 DW03\_reg\_s\_pl 255

DW03\_shftreg 258  
 DW03\_updn\_ctr 204  
 DW04\_par\_gen 163  
 DW04\_shad\_reg 256  
 DWF\_dp\_absval 142  
 DWF\_dp\_blend 143  
 DWF\_dp\_count\_ones 144  
 DWF\_dp\_mult\_comb 145  
 DWF\_dp\_mult\_comb\_sat 146  
 DWF\_dp\_mult\_ovfldet 147  
 DWF\_dp\_mult\_sat 148  
 DWF\_dp\_rnd 149  
 DWF\_dp\_rndsat 150  
 DWF\_dp\_sat 151  
 DWF\_dp\_sign\_select 152  
 DWF\_dp\_simd\_add 154  
 DWF\_dp\_simd\_addc 153  
 DWF\_dp\_simd\_mult 155

**F**

Fiber Channel 164  
 FIFO Controllers 205  
 FIFOs 205  
 FIR and IIR filters 16  
 Floating Point components 101  
 format conversions 101

**G**

generic technology library 321  
 get\_license 17  
 Gigabit Ethernet 164  
 GTEC Library Overview 321  
 GTECH 16  
 GTECH Library Overview 321  
 GTECH technology-independent library 321  
 gtech.db 321

**H**

Help, Getting 12

**I**

IEEE 1149.1 305  
 IEEE 754 Floating Point 101  
 IEEE formats 101  
 Interface 16

**J**

JTAG 305  
 JTAG IP 16

**L**

Licensing for Synopsys products 12  
 Logic 16

**M**

Memory 16  
 memory registers 251  
 memory stacks 276

**P**

parking and locking 19  
 Presto 30

product-of-sum 30

**R**

remove\_license 17

**S**

SCL (Licensing) 12  
 Sequential 194  
 sum-of-product 30  
 Support Center 13  
 Synchronous RAM 259  
 synlib\_wait\_for\_design\_license 17  
 Synopsys Common Licensing 12

**T**

TAP controller 16  
 technology-driven synthesis 30  
 Test 16  
 Test Access Port (TAP) 305  
 trigonometric IP 139

**V**

VCS simulator 18  
 VCS Verification Library 14, 15  
 Verilog simulator 18