# DSP48 Macro v3.0

## *LogiCORE IP Product Guide*

Discontinued IP

**Vivado Design Suite**

**PG148 November 18, 2015**

XILINX
ALL PROGRAMMABLE™

# Table of Contents



Discontinued IP

Discontinued IP

# Introduction

The Xilinx® LogiCORE™ DSP48 Macro provides an easy-to-use interface that abstracts the DSP Slice and simplifies its dynamic operation by enabling the specification of multiple operations using a set of user-defined arithmetic expressions. The specified operations are enumerated and can be selected through a single port on the generated core.

# Features

- Simplified and abstracted interface to DSP Slice enhances ease of use, code readability and portability
- Define DSP Slice operation using a list of user-defined arithmetic expressions
- Support for up to 64 instructions
- Supports the DSP Slice pre-adder
- Configurable pipelining
- Choose between DSP Slice or FPGA logic implementation

| LogiCORE IP Facts Table | |
|---|---|
| **Core Specifics** | |
| Supported Device Family(1) | UltraScale+™ Families<br>UltraScale™ Architecture<br>Zynq®-7000 All Programmable SoC<br>7 Series |
| Supported User Interfaces | N/A |
| Resources | Performance and Resource Utilization web page |
| **Provided with Core** | |
| Design Files | Encrypted RTL |
| Example Design | Not Provided |
| Test Bench | Not Provided |
| Constraints File | Not Provided |
| Simulation Model | Encrypted VHDL |
| Supported S/W Driver | N/A |
| **Tested Design Flows(2)** | |
| Design Entry | Vivado® Design Suite<br>System Generator for DSP |
| Simulation | For supported simulators, see the Xilinx Design Tools: Release Notes Guide. |
| Synthesis | Vivado Synthesis |
| **Support** | |
| Provided by Xilinx at the Xilinx Support web page | |

**Notes:**

1. For a complete listing of supported devices, see the Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Discontinued IP

# Overview

The DSP48 Macro core allows straightforward configuration of the DSP Slice by specifying user-defined instructions. Multiple instructions can be specified, and the instruction being performed can be changed dynamically at run time.

Discontinued IP

## Feature Summary

The DSP48 Macro core supports up to 64 separate instructions, which can be selected dynamically though a single control port.

The pipeline stages in the core can be allocated automatically in the user interface in the Vivado® Integrated Design Environment (IDE), specified in tiers, or you can individually specify them.

The option to enable and specify the use of DSP Slice cascade ports allows multiple DSP48 Macro cores to be efficiently connected to construct a larger circuit.

## Applications

- Basic DSP Slice operations such as accumulator, multiplier, adder
- Time-shared math operations using a single DSP Slice such as complex multiplication
- DSP engine / co-processor
- MAC FIR

## Licensing and Ordering Information

This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

For more information, visit the DSP48 Macro product web page.

# Product Specification

## Resource Utilization

For full details about performance and resource utilization, visit the Performance and Resource Utilization web page.

## Port Descriptions

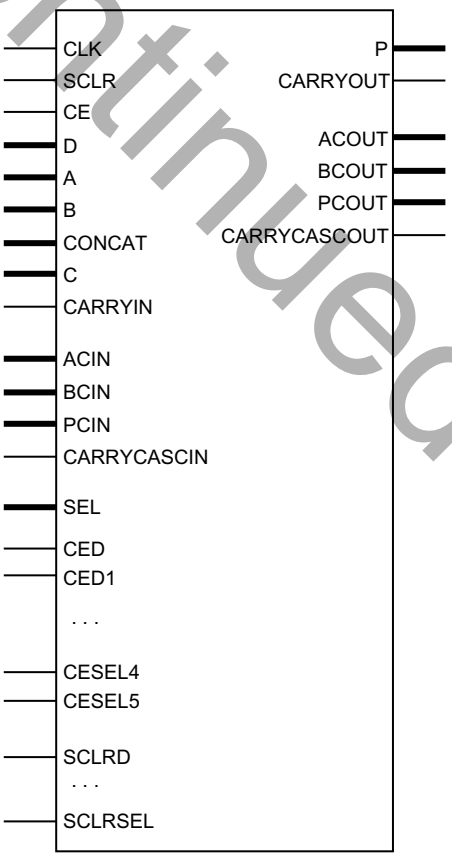


*Figure 2-1:* **Core Schematic Symbol**

Discontinued IP

Figure 2-1 and Table 2-1 illustrate and define the schematic symbol signal names. All control inputs are active-High.

*Table 2-1:* **Core Signal Pinout**

| Signal | Direction | Optional | Description |
|---|---|---|---|
| CLK | Input | No | Clock – active rising edge. |
| CE | Input | Yes | Clock Enable – core clock enable (active-High). |
| SCLR | Input | Yes | Synchronous Clear – synchronous reset (active-High). Asserting SCLR synchronously with CLK resets all registers. SCLR has priority over CE. |
| D [d_width-1:0] | Input | Yes | D port – primary input to DSP Slice pre-adder. The maximum d_width is 25 bits for 7 series devices and 27 bits for UltraScale™ devices. |
| A [a_width-1:0] | Input | Yes | A port – input to DSP Slice multiplier and secondary input (subtrahend) to pre-adder. The maximum a_width is 25 bits for 7 series devices and 27 bits for UltraScale devices. |
| ACIN [ac_width:0] | Input | Yes | Cascaded A port – used as per the A port but must be driven by the ACOUT of the previous DSP Slice, avoids FPGA routing and logic. Static selection between A and ACIN is made by the specified DSP Macro instructions. The maximum ac_width is 25 bits for 7 series devices and 27 bits for UltraScale devices, signed-extended to 30 bits. |
| B [b_width-1:0] | Input | Yes | B port – second input to multiplier.<br>Maximum b_width: 18 bits |
| BCIN [bc_width-1:0] | Input | Yes | Cascaded B port. Must be driven by the BCOUT of the previous DSP Slice. Static selection between B and BCIN.<br>Fixed bc_width: 18 bits |
| CONCAT [concat_width-1:0] | Input | Yes | CONCAT port – concatenation of the A and B DSP Slice inputs. The D input is included in the concatenation. Input to second stage add/sub. Mutually exclusive to the A,B and D ports. The maximum concat_width is 48 bits.<br>**Note**: The A port, when specified, can be passed directly to the add/sub. |
| C [c_width-1:0] | Input | Yes | C port – input to DSP Slice add/sub.<br>Maximum c_width: 48 bits |
| PCIN [pc_width-1:0] | Input | Yes | PCIN port – cascaded P input from the previous DSP Slice. Input to add/sub. Avoids FPGA routing and provides a low latency path.<br>Fixed pc_width: 48 bits |
| CARRYIN | Input | Yes | CARRYIN port – carry input from FPGA logic, single bit. |
| CARRYCASCIN | Input | Yes | CARRYCASCIN port - cascaded carry input from the previous DSP Slice. Can be used to construct large adders. |
| SEL [sel_width-1:0] | Input | Yes | SEL port - selects from the enumerated list of instructions specified in the core user interface. Unsigned.<br>Fixed sel_width: ceil(log2(num_instructions)) |
| CED1..3 | Input | Yes | Clock enables for D path registers (active-High) |

Discontinued IP

*Table 2-1:* **Core Signal Pinout *(Cont'd)***

| Signal | Direction | Optional | Description |
|---|---|---|---|
| CEA1..4 | Input | Yes | Clock enables for A path registers (active-High) |
| CEB1..4 | Input | Yes | Clock enables for B path registers (active-High) |
| CECONCAT3..5 | Input | Yes | Clock enables for CONCAT path registers (active-High |
| CEC1..5 | Input | Yes | Clock enables for C path registers (active-High) |
| CEM | Input | Yes | Clock enables for M path registers (active-High) |
| CEP | Input | Yes | Clock enables for P path registers (active-High) |
| CESEL1..5 | Input | Yes | Clock enables for SEL path registers (active-High) |
| SCLRD | Input | Yes | Synchronous reset for D path registers (active-High) |
| SCLRA | Input | Yes | Synchronous reset for A path registers (active-High) |
| SCLRB | Input | Yes | Synchronous reset for B path registers (active-High) |
| SCLRCONCAT | Input | Yes | Synchronous reset for CONCAT path registers (active-High) |
| SCLRC | Input | Yes | Synchronous reset for C path registers (active-High) |
| SCLRM | Input | Yes | Synchronous reset for M path registers (active-High) |
| SCLRP | Input | Yes | Synchronous reset for P path registers (active-High) |
| SCLRSEL | Input | Yes | Synchronous reset for SEL path registers (active-High) |
| P [p_msb-p_lsb:0] | Output | No | P port - output from DSP Slice add/sub, provides the result of the selected instruction.<br>Maximum p_msb: 47 bits<br>***Note:*** p_msb and p_lsb are derived from the Output Port Properties; Full Precision Width and Width. See Implementation Page for more details. |
| CARRYOUT | Output | Yes | CARRYOUT port - carryout to FPGA logic from add/sub. |
| ACOUT [ac_width-1:0] | Output | Yes | ACOUT port - optional cascade A output, ac_width defined for ACIN. |
| BCOUT [bc_width-1:0] | Output | Yes | BCOUT port - optional cascade B output, bc_width defined for BCIN. |
| PCOUT [pc_width-1:0] | Output | Yes | PCOUT port - optional cascade P output, pc_width defined for PCIN. |
| CARRYCASCOUT | Output | Yes | CARRYCASCOUT port - optional cascade carryout output. |

Discontinued IP

# Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

Discontinued IP

## Clocking

The DSP48 Macro requires a single clock, CLK, and is active-High triggered.

Active-High clock enables can be selected on either a global or a per-register basis.

## Resets

The DSP48 Macro has the option of selecting a global synchronous reset, or a reset per path.

Asserting an SCLR pin for a single cycle resets the relevant registers.

SCLR always overrides CE for all registers.

## Using the DSP48 Macro IP Core

The core user interface performs error-checking on all input parameters. Several files are produced when a core is generated. For detailed instructions, see the *Vivado® Design Suite User Guide, Designing with IP* (UG896) [Ref 1].

### Instruction Format

The instructions are case insensitive and ignore spaces between operands. The left side of the arithmetic expression, P=, is implicitly declared and should not be specified

Valid operands: **D, A, ACIN, B, BCIN, CONCAT, C, PCIN, CARRYIN, CARRYCASCIN, 0**.

Valid operators: **+, -, *, >>17, (, )**.

- The >>17 operator targets the DSP Slice 17-bit wire shift; valid only for P and PCIN operators.

Valid functions: **rndsimple, rndsym, rndmacc**

- Rounding functions require that the P output width is less than full precision.
- See Supported Functions for further details.

### *Examples*

Accumulator

1. **C** : Load
2. **C+P** : Accumulate (add)
3. **P-C** : Accumulate (subtract)

Three input add:

1. **C+CONCAT+P**

Multiply Accumulate

1. **A*B** : Load
2. **P+A*B**: Multiply accumulate

15-tap symmetric filter, where data is provided sequentially on the A,D and B inputs. A and D are used for the data values and B is used for the filter coefficients.

1. **(A+D)*B** : Taps 1 & 15
2. **(A+D)*B+P** : Taps 2...7 & 14...9
3. **A*B+P** : Tap 8
4. **rndsym(P)** : Round result

### *Construction of Supported Instructions*

The following two sections illustrate how the list of supported instructions are constructed.

#### Notation

*varname* = [ I1,I2,....]

This indicates *varname* supports the list of operator/operand combinations I1, I2, ....

Discontinued IP

### *7 Series Devices*

preadder = [ (A+D) , (D+A) , (D-A), D, -A , A, (ACIN+D) , (D+ACIN) , (D-ACIN), -ACIN , ACIN ]

mult_ip1 = [ ACIN , A , preadder ]

mult_ip2 = [ BCIN , B , 1 ]

mult = [ mult_ip1 * mult_ip2 , mult_ip2 * mult_ip1 ]

The 1 is not explicitly required when defining an instruction; the 1 and corresponding * operator are ignored.

xmux = [ CONCAT , P , 0 ]

ymux = [ C , 0 ]

zmux = [ C , PCIN , P , P>>17 , PCIN>>17, 0 ]

cinmux = [ CARRYIN , CARRYCASCIN , 0 ]

Similarly, 0 is not explicitly required when defining an instruction. The 0 and corresponding operator are ignored.

xycomb = [ xmux + ymux , ymux + xmux , mult ]

**valid instructions** = [

xycomb + zmux + cinmux ,

xycomb - zmux,

zmux + xycomb + cinmux ,

zmux - ( xycomb + cinmux ) ,

zmux - xmux - ymux - cinmux ,

zmux - ymux - xmux - cinmux ,

-zmux + xmux + ymux,

-zmux + ymux + xmux,

-zmux - xmux,

rndsimple( zmux + xmux + cinmux )

rndsimple( mult + cinmux )

rndsym( [ P , PCIN ] ) ]

rndsym( mult )

rndmacc( mult + P )

Note that for the rndsimple functions, operand C is not supported for zmux.

Discontinued IP

### *UltraScale™ Devices*

preadder = [ (A+D) , (D+A) , (D-A), D, -A , A, (ACIN+D) , (D+ACIN) , (D-ACIN), -ACIN , ACIN ]

mult_ip1 = [ ACIN , A , preadder ]

mult_ip2 = [ BCIN , B , 1 ]

mult = [ mult_ip1 * mult_ip2 , mult_ip2 * mult_ip1 ]

The 1 is not explicitly required when defining an instruction; the 1 and corresponding * operator are ignored.

wmux = [ C , P , 0 ]

xmux = [ CONCAT , P , 0 ]

ymux = [ C , 0 ]

zmux = [ C , PCIN , P , P>>17 , PCIN>>17, 0 ]

cinmux = [ CARRYIN , CARRYCASCIN , 0 ]

Similarly, 0 is not explicitly required when defining an instruction. The 0 and corresponding operator are ignored.

xycomb = [ wmux + xmux + ymux , wmux + ymux + xmux , mult ]

**valid instructions** = [

xycomb + zmux + cinmux ,

zmux + xycomb + cinmux ,

zmux - ( xycomb + cinmux ) ,

zmux - wmux - xmux - ymux - cinmux ,

zmux - wmux - ymux - xmux - cinmux ,

-zmux +wmux + xmux + ymux,

-zmux +wmux + ymux + xmux,

-zmux - wmux - xmux,

-zmux - xmux - wmux,

-zmux - mult,

rndsimple( zmux + xmux + ymux + cinmux )

rndsimple( mult + zmux + cinmux )

rndsym( [ P , PCIN ] ) ]

rndsym( mult )

rndmacc( mult +wmux + P )

Discontinued IP

### Restrictions

- CONCAT operand is mutually exclusive to *mult* operands. When any input to the multiplier is specified, the CONCAT operand is restricted for all instructions, or vice versa. The inclusion of 1 in *mult_ip2* enables all *mult_ip1* combinations to be used as direct inputs to the second stage add/sub.
- The choice between A and ACIN is static; after one is specified the other is restricted. Similarly for B and BCIN.
- The use of CARRYCASCIN is restricted to a subset of instructions.

## Supported Functions

### RNDSIMPLE(arg)

RNDSIMPLE implements a non-symmetric round to negative; this is equivalent to the MATLAB® function *ceil(arg-0.5)*.

*Table 3-1:* **Illustration of RNDSIMPLE Return Values**

| arg | RNDSIMPLE(*arg*) |
|---|---|
| <x.5 | x |
| x.5 | x |
| >x.5 | x+1 |
| >-x.5 | -x |
| -x.5 | -x-1 |
| <-x.5 | -x-1 |

The binary point of *arg* is defined by the full precision output width and the specified core output width; the core output is taken from the upper MSBs of the DSP Slice output with the remaining LSBs considered as the fractional portion. The binary point is taken as *full precision p_width - p_width*.

A rounding constant with a binary value of 0.0111...(or 0.499...) is added to *arg* and the LSBs removed by the process of reinterpreting the full precision output to the specified core output width. The LSBs remain on the accumulator.

*Arg* can include CARRYIN. This enables CARRYIN to determine the rounding direction. The assertion of CARRYIN is equivalent to adding 0.00...01. This modifies the rounding constant to 0.100.. (or 0.5) and therefore the rounding direction. This can be used to implement random rounding.

Discontinued IP

### *RNDSYM(arg)*

RNDSYM implements a symmetric round to highest magnitude; this is equivalent to the MATLAB function *round(arg)*.

*Table 3-2:* **Illustration of RNDSYM Return Values**

| arg | RNDSYM(*arg*) |
|---|---|
| <x.5 | x |
| x.5 | x+1 |
| >x.5 | x+1 |
| >-x.5 | -x |
| -x.5 | -x-1 |
| <-x.5 | -x-1 |

The binary point of *arg* is defined in the same manner as for the RNDSIMPLE(arg) function.

A rounding constant with a binary value of 0.0111.... (or 0.499...) is added to *arg* along with a carryin, defined as the inverse sign bit of *arg*. The LSBs are removed by the process of reinterpreting the full precision output to the specified core output width. The LSBs remain on the accumulator. For multiply operations, the XNOR of the multiplier inputs is used instead of the inverse sign bit of arg, performing multiply rounding toward infinity.

When carryin is 1, this is equivalent to adding 0.00....01. This modifies the rounding constant to 0.100.. (or 0.5) and therefore the rounding direction.

### *RNDMACC(arg)*

When performing symmetric rounding towards infinity of MACC and add accumulate operations, it is difficult to determine the sign of the output ahead of time, so the round might cost an extra clock cycle. This extra cycle can be eliminated by adding the C input rounding constant (typically a binary value of 0.0111...) on the very first cycle. The sign bit of the last but one cycle of the accumulator can be used for the final rounding operation done in the final accumulate cycle. This implementation is a practical way to save a clock cycle. There is a rare chance that the final accumulate operation can flip the sign of the output from the previous accumulated value.

To perform this for a MACC operation, the following DSP48 Macro instructions are used:

P = (A+D)*B+C (first cycle only)

P = (A+D)*B+P (middle cycles)

P = RNDMACC((A+D)*B+P) (last cycle only)

Discontinued IP

## Detailed Pipeline Implementation

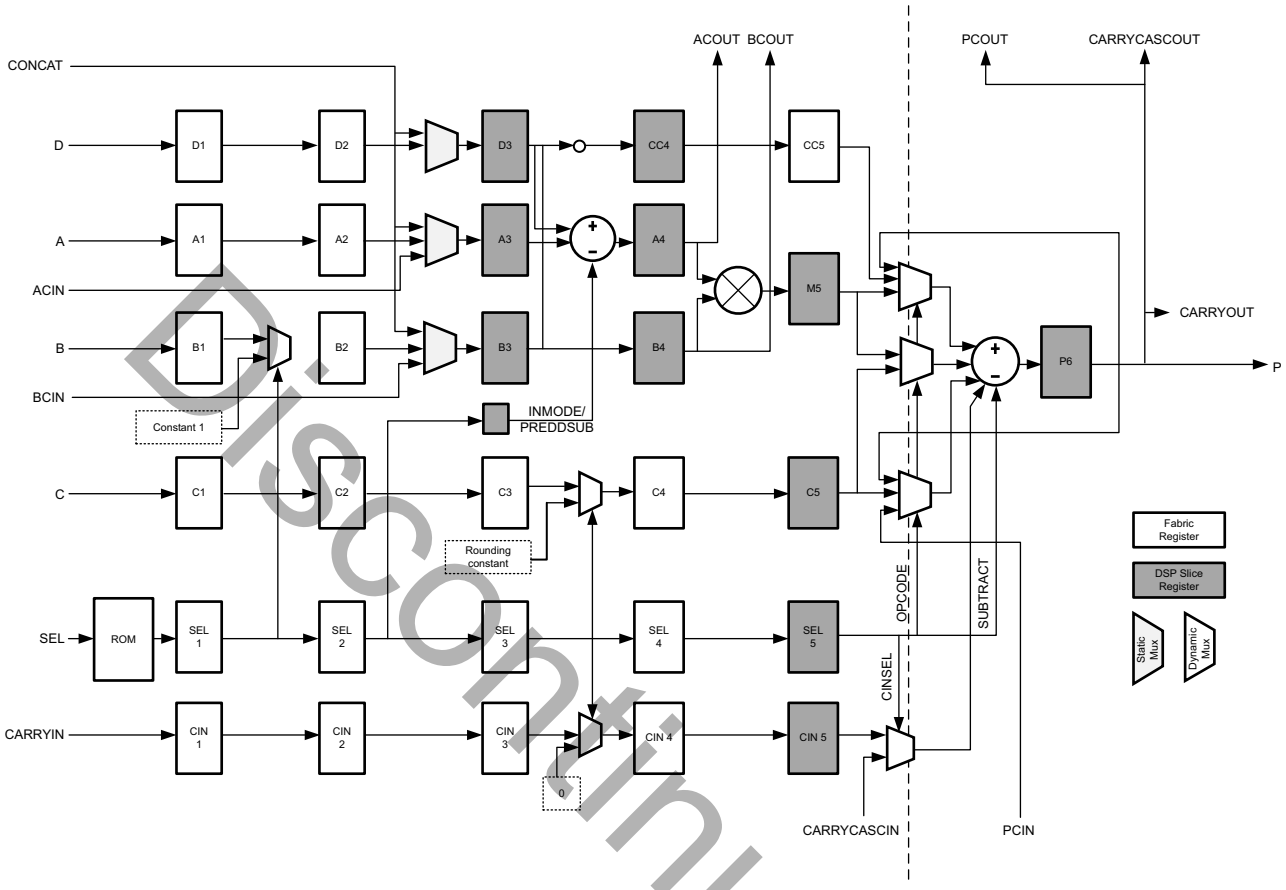


*Figure 3-1:* **DSP48 Macro Detailed Implementation**

Figure 3-1 illustrates the generalized DSP48 Macro implementation for 7 series devices.

***Note:***

- The second stage add/sub input mux implementations vary depending on the selected device.
- A static mux is resolved at core generation, whereas a dynamic mux is implemented in the generated core.

Users requiring a specific mapping to registers in the DSP48 primitive can use Table 3-3 to determine the corresponding DSP48 Macro registers. See the *UltraScale Architecture DSP Slice User Guide* (UG579) [Ref 2] or the *7 Series FPGA DSP48E1 Slice User Guide* (UG479) [Ref 3].

Discontinued IP

*Table 3-3:* **Register Mapping for DSP48 Macro**

| DSP48 Macro | Using Pre-adder[(1)] | No Pre-adder |
|---|---|---|
| D3 | D | N/A |
| A3 | A1 | A1 |
| A4 | AD | A2 |
| B3 | B1 | B1 |
| B4 | B2 | B2 |

**Notes:**

1. The mapping of A4 changes depending on whether the pre-adder is used so the tiered latency model is maintained.

Discontinued IP

# Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 4]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7]

Discontinued IP

## Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the Vivado IP Integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 4] for detailed information. IP Integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5].

The DSP48 Macro core has three pages used to configure the core plus two informational tabs.

### *Tab 1: IP Symbol*

The IP Symbol tab illustrates the core pinout.

### *Tab 2: Instruction Summary*

The Instruction Summary tab displays the complete list of specified instructions and the SEL value that is required to select each instruction.

### *Instructions Page*

This page is used to specify the instructions that the core is to implement.

- **Component Name**: The name of the core component to be instantiated. The name must begin with a letter and be composed of the following characters: a to z, 0 to 9, and "_".
- **Available instructions**: Informational parameter. When the 'Show Filtered' check-box is ticked, the available instruction list dynamically updates to show the remaining valid instructions given the instruction that is currently being entered into the user interface. Instructions can be selected in the Available instruction panel and "drag and dropped" into an Instruction parameter.
- **Instructions 0 to 7**: Specifies the operations the core is to implement. Text entry of the desired arithmetic operation to be generated on the P output port. The left side of the expression, P=, is implicitly declared and should not be specified. Instructions are case insensitive.

  See Instruction Format for further details on the instruction format and supported operations.

- **Instructions 8 to 63**: Specifies instructions 8 to 63 using a comma-delimited list of instructions.

### *Pipeline Options Page*

The pipeline depths of the various input paths are specified on this page.

- **Pipeline Options**: Specifies the pipeline method to be used: Automatic, By Tier and Expert.
- **Custom Pipeline options**: Specifies the pipeline depth of the various input paths.
  - **Tier 1 to 6**: When "By Tier" has been selected for Pipeline Options, these parameters are used to enable/disable the registers across all the input paths for a given pipeline stage. Some restrictions are enforced.

Discontinued IP

    - When P has been specified in an expression, tier 6 is forced, as asynchronous feedback is not supported.

  - **Individual registers**: When "Expert" has been selected for the Pipeline Options, these parameters are used to enable/disable individual register stages. Some restrictions are enforced.
    - The P register is forced when P has been specified in an expression. Asynchronous feedback is not supported.

See Detailed Pipeline Implementation for further details on how the various pipeline stages relate to the core implementation.

- **Control Ports**:
  - **CE**
    - Global check box: enables a single CE pin for all registers in the core.
    - D, A, B, CONCAT, C, M, P, SEL/CARRYIN check boxes: enable individual CE pins for all enabled registers in the core.
  - **SCLR**
    - Global check box: enables a single SCLR pin for all registers in the core.
    - D, A, B, CONCAT, C, M, P, SEL/CARRYIN checkboxes: enable an individual SCLR pin for each datapath.

### *Implementation Page*

- **Input Port Properties**: Specifies the bit-width of the D, A, B, CONCAT and C input ports. See Table 2-1 for maximum port widths.
- **Output Port Properties**: Specifies the precision of the P output port; Full Precision and User Defined.
- The Vivado IDE I automatically calculates the full precision output width given the width of the specified input ports. When P has been used as an operand, the full precision output width is set to the full DSP Slice width of 48 bits. When Full Precision is selected, the output width is set to the full precision value. When User Defined is selected, the output width can be set to any value up to 48 bits. When the specified value is less than the full precision width, the output is truncated, that is, the LSBs are removed. This option should be used when a rounding function has been specified.
  - **Width**: Specifies the actual output width of the P output port. When specified to be less than Full Precision, the DSP Slice output is truncated.
- **Additional Ports**: Specifies if the core has a CARRYOUT output port or the ACOUT, BCOUT, PCOUT or CARRYCASCOUT cascaded output ports.
- **Use DSP Slice**: Specifies if the core implementation uses an DSP Slice or FPGA logic equivalent. When a FPGA logic implementation is specified, the core is unlikely to achieve the same $F_{max}$ as DSP Slice.

Discontinued IP

## User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

*Table 4-1:* **Vivado IDE Parameter to User Parameter Relationship**

| Vivado IDE Parameter | User Parameter | Default Value |
|---|---|---|
| Show Filtered | show_filtered | False |
| Instructions: 0 | instruction1 | A*B+C |
| Instructions: 1 | instruction2 | # |
| Instructions: 2 | instruction3 | # |
| Instructions: 3 | instruction4 | # |
| Instructions: 4 | instruction5 | # |
| Instructions: 5 | instruction6 | # |
| Instructions: 6 | instruction7 | # |
| Instructions: 7 | instruction8 | # |
| Instructions: 8-63 | instruction_list | # |
| Pipeline Options | pipeline_options | Automatic |
| Custom Pipeline options: Tier: 1 | tier_1 | False |
| Custom Pipeline options: Tier: 2 | tier_2 | False |
| Custom Pipeline options: Tier: 3 | tier_3 | False |
| Custom Pipeline options: Tier: 4 | tier_4 | False |
| Custom Pipeline options: Tier: 5 | tier_5 | False |
| Custom Pipeline options: Tier: 6 | tier_6 | False |
| Custom Pipeline options: Tier: D: 1 | dreg_1 | False |
| Custom Pipeline options: Tier: D: 2 | dreg_2 | False |
| Custom Pipeline options: Tier: D: 3 | dreg_3 | True |
| Custom Pipeline options: Tier: A: 1 | areg_1 | False |
| Custom Pipeline options: Tier: A: 2 | areg_2 | False |
| Custom Pipeline options: Tier: A: 3 | areg_3 | True |
| Custom Pipeline options: Tier: A: 4 | areg_4 | True |
| Custom Pipeline options: Tier: B: 1 | breg_1 | False |
| Custom Pipeline options: Tier: B: 2 | breg_2 | False |
| Custom Pipeline options: Tier: B: 3 | breg_3 | True |
| Custom Pipeline options: Tier: B: 4 | breg_4 | True |
| Custom Pipeline options: Tier: C: 1 | creg_1 | False |
| Custom Pipeline options: Tier: C: 2 | creg_2 | False |
| Custom Pipeline options: Tier: C: 3 | creg_3 | True |

Discontinued IP

*Table 4-1:* **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

| Vivado IDE Parameter | User Parameter | Default Value |
|---|---|---|
| Custom Pipeline options: Tier: C: 4 | creg_4 | True |
| Custom Pipeline options: Tier: C: 5 | creg_5 | True |
| Custom Pipeline options: Tier: CONCAT: 3 | concatreg_3 | False |
| Custom Pipeline options: Tier: CONCAT: 4 | concatreg_4 | False |
| Custom Pipeline options: Tier: CONCAT: 5 | concatreg_5 | False |
| Custom Pipeline options: Tier: SEL: 1 | opreg_1 | False |
| Custom Pipeline options: Tier: SEL: 1 | opreg_2 | False |
| Custom Pipeline options: Tier: SEL: 1 | opreg_3 | False |
| Custom Pipeline options: Tier: SEL: 1 | opreg_4 | False |
| Custom Pipeline options: Tier: SEL: 1 | opreg_5 | False |
| Custom Pipeline options: Tier: CARRYIN: 1 | cinreg_1 | False |
| Custom Pipeline options: Tier: CARRYIN : 2 | cinreg_2 | False |
| Custom Pipeline options: Tier: CARRYIN : 3 | cinreg_3 | False |
| Custom Pipeline options: Tier: CARRYIN : 4 | cinreg_4 | False |
| Custom Pipeline options: Tier: CARRYIN : 5 | cinreg_5 | False |
| Custom Pipeline options: Tier: CARRYIN : 5 | mreg_5 | True |
| Custom Pipeline options: Tier: B: 5 | preg_6 | True |
| Input Port Properties: D | d_width | 18 |
| Input Port Properties: A | a_width | 18 |
| Input Port Properties: B | b_width | 18 |
| Input Port Properties: CONCAT | concat_width | 48 |
| Input Port Properties: C | c_width | 48 |
| Output Properties | output_properties | Full_Precision |
| Output Port Properties: Width | p_width | 48 |
| Use CARRYOUT | has_carryout | False |
| Use ACOUT | has_acout | False |
| Use BCOUT | has_bcout | False |
| Use CARRYCASCOUT | has_carrycascout | False |
| Use PCOUT | has_pcout | False |
| Control Ports: CE : Global | has_ce | False |
| Control Ports: CE : D | has_d_ce | False |
| Control Ports: CE : A | has_a_ce | False |
| Control Ports: CE : B | has_b_ce | False |
| Control Ports: CE : C | has_c_ce | False |
| Control Ports: CE : CONCAT | has_concat_ce | False |

Discontinued IP

*Table 4-1:* **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

| Vivado IDE Parameter | User Parameter | Default Value |
|---|---|---|
| Control Ports: CE : M | has_m_ce | False |
| Control Ports: CE : P | has_p_ce | False |
| Control Ports: CE : SEL/CARRYIN | has_sel_ce | False |
| Control Ports: SCLR : Global | has_sclr | False |
| Control Ports: SCLR : D | has_d_sclr | False |
| Control Ports: SCLR : A | has_a_sclr | False |
| Control Ports: SCLR : B | has_b_sclr | False |
| Control Ports: SCLR : C | has_c_sclr | False |
| Control Ports: SCLR : CONCAT | has_concat_sclr | False |
| Control Ports: SCLR : M | has_m_sclr | False |
| Control Ports: SCLR : P | has_p_sclr | False |
| Control Ports: SCLR : SEL/CARRYIN | has_sel_sclr | False |

## Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Discontinued IP

# System Generator for DSP

The DSP48 Macro core is available through Xilinx® System Generator for DSP, a design tool that enables the use of the model-based design environment Simulink® for FPGA design. The DSP48 Macro core is one of the DSP building blocks provided in the Xilinx blockset for Simulink. The core can be found in the Xilinx Blockset in the DSP section. The block is called "DSP48 Macro v3.0." See the *System Generator for DSP User Guide* [Ref 6] for more information.

## System Generator for DSP Graphical User Interface

This section describes each tab of the System Generator for DSP GUI and details the parameters that differ from the Vivado IDE.

### *Tab 1: Instructions*

The Instruction tab is used to define the operations that the core is to implement. Each instruction can be entered on a new line, or in a comma delimited list, and are enumerated from the top-down. A maximum of 64 instructions can be specified. See Instructions Page and Instruction Format of Using the DSP48 Macro IP Core for details on supported instructions and their format.

### *Tab 2: Pipeline Options*

The Pipeline Options tab is used to define the pipeline depth of the various input paths. See the Pipeline Options Page for details of all the core parameters on this tab.

### *Tab 3: Implementation*

The Implementation tab is used to define implementation options. See the Implementation Page for details of all the core parameters on this tab.

- **Output Port Properties:** Specifies the precision of the P output port; Full Precision and User Defined.
- The core automatically calculates the full precision output width and binary point position given the width and binary point of the specified input ports. When P has been used as an operand, the full precision output width is set to the full DSP Slice width of 48 bits.
- When Full Precision is selected, the output is set to full precision width and binary point.
- When User Defined is selected, the output width can be set to any value up to 48 bits. The output formatting has two modes of operation:

  Full precision binary point is calculated to be zero.

  - When the output width is specified to be less than the full precision width, the output is truncated, that is, the LSBs are removed.

  Full precision binary point is calculated to be greater than zero.

  - The binary point is anchored. The output of the core behaves in the same manner as a System Generator Convert block with the following settings: **Quantization > Truncate and Overflow > Wrap**. Some restrictions on the binary point values are enforced; it cannot be greater than the full precision binary point value and its permitted minimum value will be modified to ensure that when the binary point value and output width are combined, the resulting MSB value does not exceed 48 bits.

  ◦ **Width**: Specifies the user-defined output width of the P output port.

  ◦ **Binary Point: Specifies the** user-defined **binary point of the P output port.**
- **ce:** Selects either a global clock enable pin, or separate clock enable pins for each register. When separate clock enable pins are selected, these are managed within System Generator to correctly handle multirate constraints.
- **rst:** Selects either a global reset pin, or separate reset pins for each datapath. In a similar way to the separate clock enable pins, System Generator manages the situation where datapaths with separate reset controls have different rates.

Discontinued IP

**FPGA Area Estimation:** See the System Generator for DSP documentation for detailed information about this section.

# Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

## Required Constraints

This section is not applicable for this IP core.

## Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

## Clock Frequencies

This section is not applicable for this IP core.

## Clock Management

This section is not applicable for this IP core.

## Clock Placement

This section is not applicable for this IP core.

## Banking

This section is not applicable for this IP core.

## Transceiver Placement

This section is not applicable for this IP core.

## I/O Standard and Placement

This section is not applicable for this IP core.

Discontinued IP

## Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7].

## Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Discontinued IP

# Migrating and Upgrading

This appendix contains information about migrating a design from the ISE® Design Suite to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Discontinued IP

## Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, *see the ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 8].

The Vivado IP upgrade functionality can be used to upgrade an existing XCO/XCI file from v2.1 to DSP48 Macro v3.0. There are no changes of functionality, port or configuration from v2.1 to v3.0.

### Parameter Changes

There are no parameter differences between DSP48 Macro versions v2.0 and v2.1 and DSP48 Macro v3.0.

### Port Changes

There are no port changes between DSP48 Macro versions v2.0 and v2.1 and DSP48 Macro v3.0.

### Functionality Changes

There are no functionality changes between DSP48 Macro versions v2.0 and v2.1 and DSP48 Macro v3.0.

# Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

## Parameter Changes

No change.

## Port Changes

No change.

## Other Changes

No change.

### *Simulation*

Starting with DSP48 Macro v3.0 (2013.3 version), behavioral simulation models have been replaced with IEEE P1735 Encrypted VHDL. The resulting model is bit and cycle accurate with the final netlist. For more information on simulation, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7].

Discontinued IP

# Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Discontinued IP

## Finding Help on Xilinx.com

To help in the design and debug process when using the DSP48 Macro, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support WebCase.

### Documentation

This product guide is the main document associated with the DSP48 Macro. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx® Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

### Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available. DSP48 Macro

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

**Master Answer Record for the DSP48 Macro**

AR: 54500

## Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

# Debug Tools

There are many tools available to address DSP48 Macro design issues. It is important to know which tools are useful for debugging various situations.

## Vivado Design Suite Debug Feature

The Vivado Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 9].

Discontinued IP

# Additional Resources and Legal Notices

## Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

## References

These documents provide supplemental material useful with this product guide:

1. *Vivado® Design Suite User Guide, Designing with IP* (UG896)
2. *UltraScale™ Architecture DSP Slice User Guide* (UG579)
3. *7 Series FPGA DSP48E1 Slice User Guide* (UG479)
4. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Getting Started* (UG910)
6. *System Generator for DSP User Guide* (UG640)
7. *Vivado Design Suite User Guide - Logic Simulation* (UG900)
8. *ISE® to Vivado Design Suite Migration Guide* (UG911)
9. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

Discontinued IP

# Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 11/18/2015 | 3.0 | UltraScale+ device support added. |
| 04/02/2014 | 3.0 | • Added link to resource utilization numbers.<br>• Added User Parameter table (Table 4-1). |
| 12/18/2013 | 3.0 | • Added UltraScale architecture support.<br>• Template updated. |
| 10/02/2013 | 3.0 | Minor updates to IP Facts table and Migrating appendix. Document version number advanced to match the core version number. |
| 03/20/2013 | 1.0 | Initial release as a product guide; replaces DS754. Added Artix®-7 support. There are no other documentation changes for this release. |

Discontinued IP

# Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.

© Copyright 2013–2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. AMBA, AMBA Designer, ARM, ARM1176JZ-S, CoreSight, Cortex, and PrimeCell are trademarks of ARM in the EU and other countries. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. All other trademarks are the property of their respective owners.