DSP Function Usage Guide for iCE40 Devices

Technical Note

FPGA-TN-02007-1.2

October 2020

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

1. Introduction

This technical note discusses DSP function usage for the iCE40™ device family, specifically iCE40 Ultra™ and iCE40 UltraPlus™. It is intended to be used as a guide on various modes and how to configure them for these devices.

The DSP block, referred to as SB_MAC16 primitive in this guide, is an embedded block available in the iCE40 Ultra and iCE40 UltraPlus devices. This block can be configured into combinations of the following functional units by selecting appropriate parameter values.

- Single 16 x 16 Multiplier (generating a 32-bit product output)
- Two independent 8 x 8 Multiplier (generating two independent 16-bit product outputs)
- Single 32-bit Accumulator
- Two independent 16-bit Accumulators
- Single 32-bit Adder/Subtractor
- Two independent 16-bit Adders/Subtractors

2. DSP Primitive – SB_MAC16

The SB_MAC16 primitive is the dedicated configurable DSP block for the iCE40 Ultra and iCE40 UltraPlus devices. This primitive can be configured into a multiplier, adder, subtractor, accumulator, multiply-add and multiply-sub by setting various parameters.

2.1. SB_MAC16 Primitive

Figure 2.1 provides an overview of the SB_MAC16 primitive with various inputs and outputs.

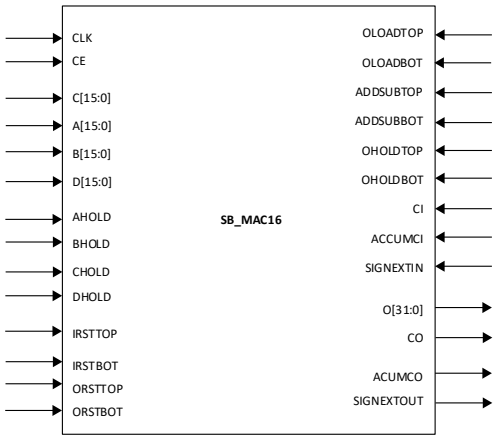

Figure 2.1. SB_MAC16 DSP Primitive Interface Diagram

The inputs and outputs of the functional units can be configured independently into:

- Registered Inputs/Outputs
  - The inputs to the functional units can be either registered or unregistered.
  - The outputs from the functional units can be either registered or unregistered.
  - The intermediate multiplier outputs can be registered (pipelined) for faster clock performance.
- Signed/Unsigned Inputs
  - Inputs to the multiplier block can be either a signed or unsigned number.

These various options and their usage is discussed in more detail in the sections that follow.

2.2. SB_MAC16 Functional Diagram

Figure 2.2 shows the functional diagram of the SB_MAC16 primitive.

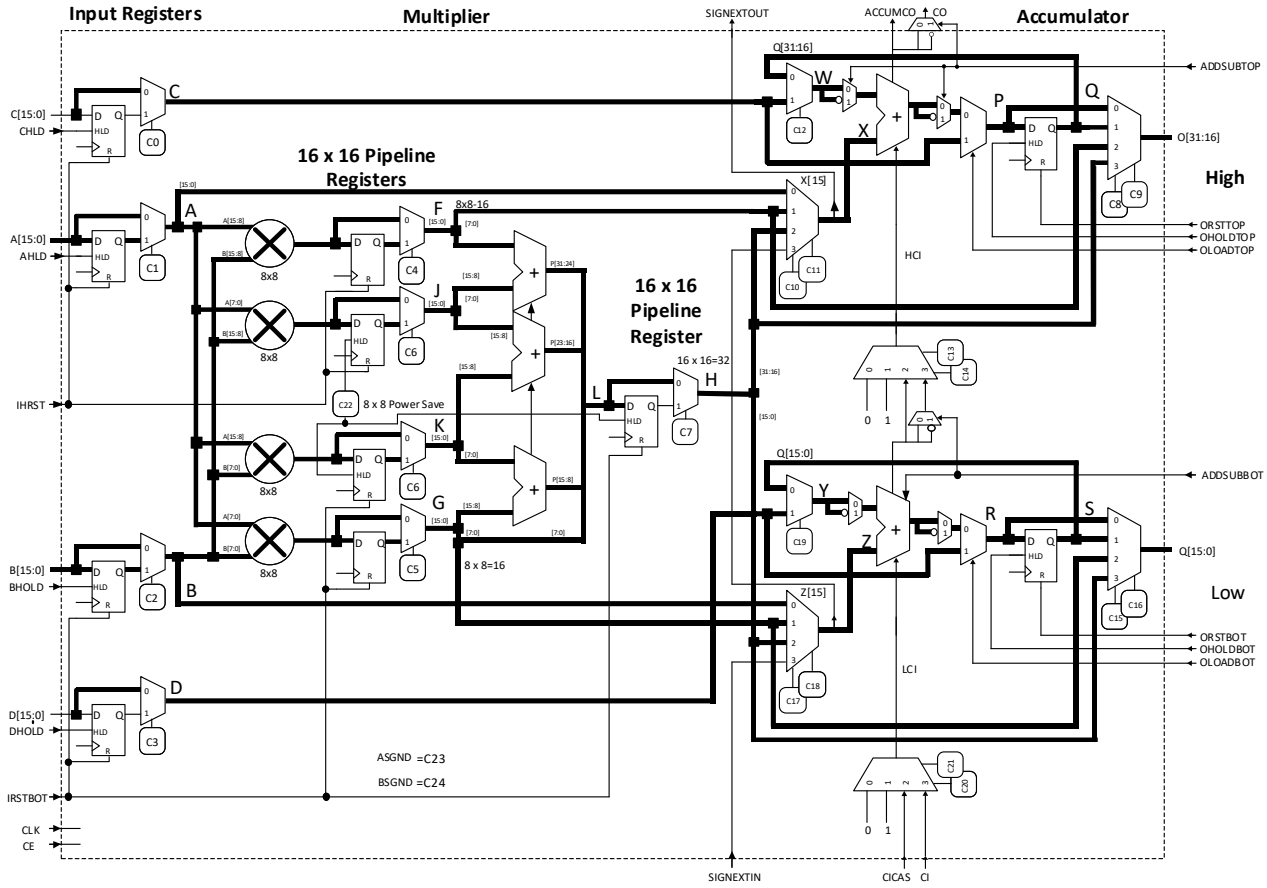

Figure 2.2. SB_MAC16 DSP Functional Diagram

2.3. SB_MAC16 Interface Ports

Table 2.1 provides a list of interface ports available in SB_MAC16 and their functional description. It is important to note the *Default Values* of these ports in this table. These values are useful in determining how to connect the ports that are not used in a particular function during instantiation. This is discussed in detail in the sections that follow.

Table 2.1. SB_MAC16 Ports and their Functional Descriptions

Port Name	Direction	Functional Description	Default Values
CLK	Input	Clock Input. Applies to all clocked elements.	—
CE	Input	Clock Enable Input. Applies to all clocked elements.	1
A[7:0]	Input	Lower 8-Bits data of Input A	8'b0
A[15:8]	Input	Upper 8-Bits data of Input A	8'b0
AHOLD	Input	Register A Hold Input. Control data flow input Register A. 0: Load 1: Hold	0
B[7:0]	Input	Lower 8-Bits data of Input B	8'b0
B[15:8]	Input	Upper 8-Bits data of Input B	8'b0
BHOLD	Input	Register B Hold Input. Control data flow input Register B. 0: Load 1: Hold	0
C[15:0]	Input	16-Bits data of Input C	16'b0
CHOLD	Input	Register C Hold Input. Control data flow input Register C. 0: Load 1: Hold	0
D[15:0]	Input	16-Bits data of Input D	16'b0
DHOLD	Input	Register D Hold Input. Control data flow input Register D. 0: Load 1: Hold	0
IRSTTOP	Input	Reset Input to Registers A and C. Also resets upper 8 x 8 Multiplier Output Register (8 x 8 MAC Pipeline Register). It is an active HIGH reset whose deassertion should be synchronized with the clock. 0: Not reset 1: Reset	0
ORSTTOP	Input	Reset Input to top Accumulator Register (for Adder/Subtractor, Accumulator, and MAC functions). It is an active HIGH reset whose deassertion should be synchronized with the clock. 0: Not reset 1: Reset	0
OLOADTOP	Input	Load Control Input to top Accumulator Register (initialize on MAC function) 0: Not load 1: Load data from Register/Input C	0
ADDSUBTOP	Input	Add/Subtract Control Input to top Accumulator 0: Add 1: Subtract	0
OHOLDTOP	Input	Top Accumulator Output Register Hold Input. Control data flow into the register. 0: Load 1: Hold	0
OUTPUT[31:16]	Output	Upper 16 bits of Output	—

Port Name	Direction	Functional Description	Default Values
IRSTBOT	Input	Reset Input to Registers A and C. Also resets upper 8 x 8 Multiplier Output Register (8 x 8 MAC Pipeline Register) and the 16 x 16 Multiplier Output Register (16 x 16 MAC Pipeline Register). It is an active HIGH reset whose deassertion should be synchronized with the clock. 0: Not reset 1: Reset	0
ORSTBOT	Input	Reset Input to top Accumulator Register (for Adder/Subtractor, Accumulator, and MAC functions). It is an active HIGH reset whose deassertion should be synchronized with the clock. 0: Not reset 1: Reset	0
OLOADBOT	Input	Load Control Input to bottom Accumulator Register (initialize on MAC function) 0: Not load 1: Load data from Register/Input D	0
ADDSUBBOT	Input	Add/Subtract Control Input to bottom Accumulator 0: Add 1: Subtract	0
OHOLDBOT	Input	Bottom Accumulator Output Register Hold Input. Control data flow into the register. 0: Load 1: Hold	0
OUTPUT[15:0]	Output	Lower 16 bits of Output	
CI	Input	Cascaded Add/Sub Carry Input from previous DSP block	0
CO	Output	Cascaded Add/Sub Carry Output to next DSP block	
ACCUMCI	Input	Cascaded Accumulator Carry Input from previous DSP block	0
ACCUMCO	Output	Cascaded Accumulator Carry Output to next DSP block	
SIGNEXTIN	Input	Sign Extension Input from previous DSP block	0
SIGNEXTOUT	Output	Sign Extension Output to next DSP block	

2.4. SB_MAC16 Parameters

The parameter table below, Table 2.2, shows a list of parameters to configure the SB_MAC16 block. This table also maps the parameters to the configuration bits shows in the SB_MAC16 Functional Diagram in Figure 2.2. For more information about the parameters, kindly refer to the iCE Technology Library.

Table 2.2. SB_MAC16 Parameter Description

Parameter Name	Configuration Bit(s)	Parameter Description and Allowed Values	Default
NEG_TRIGGER	—	Input Clock Polarity: 0: Rising edge 1: Falling edge	0
C_REG	C0	Input C Register Control: 0: Not registered 1: Registered	0
A_REG	C1	Input A Register Control: 0: Not registered 1: Registered	0
B_REG	C2	Input B Register Control: 0: Not registered 1: Registered	0
D_REG	C3	Input D Register Control: 0: Not registered 1: Registered	0
TOP_8x8_MULT_REG	C4	Top 8 x 8 Multiplier Output Register Control (Pipeline Register for MAC): 0: Not registered 1: Registered	0
BOT_8x8_MULT_REG	C5	Bottom 8 x 8 Multiplier Output Register Control (Pipeline Register for MAC): 0: Not registered 1: Registered	0
PIPELINE_16X16_MULT_REG1	C6	16 x 16 Multiplier Pipeline Register Control: 0: Not registered 1: Registered	0
PIPELINE_16x16_MULT_REG2	C7	16 x 16 Multiplier Output Register Control (Pipeline Register for MAC): 0: Not registered 1: Registered	0
TOPOUTPUT_SELECT	C9, C8	Top Output Select: 00: Adder/Subtractor, not registered 01: Adder/Subtractor, registered 10: 8 x 8 Multiplier 11: 16 x 16 Multiplier	00
TOPADDSUB_LOWERINPUT	C11, C10	Input X of upper Adder/Subtractor: 00: Input A 01: 8 x 8 Multiplier Output at Top 10: 16 x 16 Multiplier upper 16-bit Outputs 11: Sign extension from Z15 (lower Adder/Subtractor input)	00
TOPADDSUB_UPPERINPUT	C12	Input W of upper Adder/Subtractor: 0: Output of Adder/Subtractor Output Register (Accumulation Function) 1: Input C	0

Parameter Name	Configuration Bit(s)	Parameter Description and Allowed Values	Default
TOPADDSUB_CARRYSELECT	C14, C13	Carry Input Select, Top Adder/Subtractor: 00: Constant 0 01: Constant 1 10: Cascade ACCUMOUT from lower Adder/Subtractor 11: Cascade CO from lower Adder/Subtractor	00
BOTOUTPUT_SELECT	C16, C15	Bottom Output Select: 00: Adder/Subtractor, not registered 01: Adder/Subtractor, registered 10: 8 x 8 Multiplier 11: 16 x 16 Multiplier	00
BOTADDSUB_LOWERINPUT	C18, C17	Input Z of upper Adder/Subtractor: 00: Input B 01: 8 x 8 Multiplier Output at Bottom 10: 16 x 16 Multiplier lower 16-bit Outputs 11: Sign extension from SIGNEXTIN	00
BOTADDSUB_UPPERINPUT	C19	Input Y of upper Adder/Subtractor: 0: Output of Adder/Subtractor Output Register (Accumulation Function) 1: Input D	0
BOTADDSUB_CARRYSELECT	C21, C20	Carry Input Select, Bottom Adder/Subtractor: 00: Constant 0 01: Constant 1 10: Cascade ACCUMOUT from lower DSP block 11: Cascade CO from lower DSP block	00
MODE_8x8	C22	Select 8 x 8 Multiplier Mode (Power Saving): 0: Not Selected 1: Selected	0 --> 1
A_SIGNED	C23	Input A Sign: 0: Input A is unsigned 1: Input A is signed	0
B_SIGNED	C24	Input B Sign: 0: Input B is unsigned 1: Input B is signed	0

3. Implementing DSP Functions

There are two ways to implement DSP functions in the iCE40 Ultra and iCE40 UltraPlus devices:

- Inferencing DSP functions
  This method requires users to define the functional behavior of the DSP function they wish to implement, and let the software tools map it to the SB_MAC16 DSP block.
- Instantiating SB_MAC16 DSP Primitives
  This method involves instantiating the SB_MAC16 primitive in the user code. The ports discussed in the above sections need to be port-mapped for each function, or tied off to their default value.

Both of these methods are discussed in detail in the following sections.

3.1. Inferencing DSP Functions

This method involves defining the desired DSP function in behavioral HDL code. This does not require you to know the details of the DSP primitive, and the function is inferred automatically based on the code.

Here is an example of inferencing a 32-bit Accumulator with asynchronous data input and synchronous (registered) data out.

3.1.1. 32-bit Accumulator with Async Data In and Sync Data Out

Verilog

```
module accum32_syncdataout (clk, accumdata_syncout, dataAB);
input clk;
input  [31:0] dataAB;
output [31:0] accumdata_syncout;
reg    [31:0] accumdata_syncout;

always@(posedge clk)
begin
       accumdata_syncout <= accumdata_syncout + dataAB ;
end

endmodule
```

VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY accum32_syncdataout IS PORT (
clk : IN STD_LOGIC;
accumdata_syncout    : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
dataAB : IN STD_LOGIC_VECTOR(31 DOWNTO 0)
);
END accum32_syncdataout;

ARCHITECTURE arch OF accum32_syncdataout IS
-- Declare intermediate signals for referenced outputs

SIGNAL accumdata_syncout_xhdl0 : STD_LOGIC_VECTOR(31 DOWNTO 0); BEGIN
-- Drive referenced outputs

accumdata_syncout <= accumdata_syncout_xhdl0;

PROCESS (clk) BEGIN
IF (clk'EVENT AND clk = '1') THEN
accumdata_syncout_xhdl0 <= accumdata_syncout_xhdl0 + dataAB; END IF;
END PROCESS;

END arch;
```

Another example is of an 8 x 8 multiplier, with both inputs and outputs registered.

3.1.2. 8 x 8 Multiplier, Unsigned with Sync Data In and Data Out

Verilog

```
module mult8x8_inoutreg_unsigned (clk,prod, a_in, b_in);
input  [7:0]  a_in;
input  [7:0]  b_in;
input         clk;
output [15:0] prod;
reg    [15:0] prod;

reg    [7:0]  a_reg, b_reg;
wire   [15:0] mult_out;

assign mult_out = a_reg * b_reg;

always@(posedge clk)
begin
       a_reg <= a_in;
       b_reg <= b_in;
       prod <= mult_out;
end

endmodule
```

VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY mult8x8_inoutreg_unsigned IS PORT (
clk: IN STD_LOGIC;
prod   : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
a_in   : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
b_in   : IN STD_LOGIC_VECTOR(7 DOWNTO 0)
);
END mult8x8_inoutreg_unsigned;

ARCHITECTURE arch OF mult8x8_inoutreg_unsigned IS

SIGNAL a_reg : STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL b_reg : STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL mult_out : STD_LOGIC_VECTOR(15 DOWNTO 0); B
EGIN

mult_out <= ("00000000" & a_reg * b_reg);

PROCESS (clk)
BEGIN
IF (clk'EVENT AND clk = '1') THEN
a_reg  <= a_in;
b_reg  <= b_in;
prod <= mult_out;
END IF;
END PROCESS;

END arch;
```

3.2. Instantiation DSP Primitive – SB_MAC16

In order to implement various functions in the DSP block, users are required to instantiate the SB_MAC16 block in their top level HDL code. Different combinations of ports are connected to the user logic for various functions.

Table 3.1 provides a summary of port connections in instantiation based on functions that are required to be implemented. The column on the left provides various signals that are needed to be port mapped during HDL instantiation. The top row provides various functions that can be implemented. The cross referenced cells indicate whether the port connection is Signal or Default.

The term *Signal* means that this is one of the signals that user will have to port map to, while implementing the function. The *Default* implies that this port has to be connected to its default value during port mapping.

The default value of a port can be referenced from Table 2.1.

In certain cases, the DSP block can have two independent functions, for example two 8 x 8 multipliers, generating two 16-bit outputs. Such cases are referenced as Top Signals and Bottom Signals in Table 3.1. In such cases, one of the 8 x 8 multipliers can be implemented using Top Signals, and the other using Bottom Signals.

Table 3.1. Instantiation Guide

Port Name	Input/ Output	8 x 8 Multiplier	16 x 16 Multiplier	16 bit Accumulator	32 bit Accumulator	16 bit Adder/ Subtractor	32 bit Adder/ Subtractor	8 x 8 MAC
CLK	Input	Signal	Signal	Signal	Signal	Signal	Signal	Signal
CE	Input	Signal	Signal	Signal	Signal	Signal	Signal	Signal
A[7:0]	Input	Bottom	Signal	Top	Signal	Top	Signal	Bottom
A[15:8]	Input	Top	Signal	Top	Signal	Top	Signal	Top
AHOLD	Input	Signal	Signal	Top -> Signal	Signal	Top -> Signal	Signal	Signal
B[7:0]	Input	Bottom	Signal	Bottom	Signal	Bottom	Signal	Bottom
B[15:8]	Input	Top	Signal	Bottom	Signal	Bottom	Signal	Top
BHOLD	Input	Signal	Signal	Bottom - Signal	Signal	Bottom - Signal	Signal	Signal
C[15:0]	Input	Default	Default	Default -> Top	Default -> Signal	Top	Default -> Signal	Top
CHOLD	Input	Default	Default	Default -> Top	Default -> Signal	Top	Default -> Signal	Top
D[15:0]	Input	Default	Default	Default -> Bottom	Default -> Signal	Bottom	Default -> Signal	Bottom
DHOLD	Input	Default	Default	Default -> Bottom	Default -> Signal	Bottom	Default -> Signal	Bottom
IRSTTOP	Input	Top -> Signal	Signal	Top -> Signal	Signal	Top -> Signal	Signal	Top -> Signal
ORSTTOP	Input	Top	Signal	Top	Signal	Top	Signal	Top
OLOADTOP	Input	Default	Default	Signal	Signal	0	0	Signal
ADDSUBTOP	Input	Default	Default	0	0	0 = Add 1 = Sub	0 = Add 1 = Sub	0
OHOLDTOP	Input	Top -> default	Signal - default	Top	Signal	Top	Signal	Top
OUTPUT[31:16]	Output	Top	Signal	Top	Signal	Top	Signal	Top
IRSTBOT	Input	Bottom -> Signal	Signal	Bottom -> Signal	Signal	Bottom -> Signal	Signal	Bottom -> Signal
ORSTBOT	Input	Bottom	Signal	Bottom	Signal	Bottom	Signal	Bottom
OLOADBOT	Input	Default	Default	Signal	Signal	0	0	Signal

Port Name	Input/ Output	8 x 8 Multiplier	16 x 16 Multiplier	16 bit Accumulator	32 bit Accumulator	16 bit Adder/ Subtractor	32 bit Adder/ Subtractor	8 x 8 MAC
ADDSUBBOT	0 = Add	0 = Add	0	0	0 = Add	0 = Add	0 = Add	0
	1 = Sub	1 = Sub			1 = Sub	1 = Sub	1 = Sub	
OHOLDBOT	Input	Bottom -> default	Signal -> default	Bottom	Signal	Bottom	Signal	Bottom
OUTPUT[15:0]	Output	Bottom	Signal	Bottom	Signal	Bottom	Signal	Bottom
CI	Input	Default	Default	Default	Default	Bottom	Signal	Default
CO	Output	Default	Default	Default -> Top	Default -> Signal	Top	Signal	Default -> Top
ACCUMCI	Input	Default	Default	Bottom	Signal	Default -> Bottom	Default -> Signal	Default -> Bottom
ACCUMCO	Output	Default	Default	Top	Signal	Default -> Top	Default -> Signal	Default -> Top
SIGNEXTIN	Input	Default	Default	Bottom	Signal	Bottom	Signal	Default -> Bottom
SIGNEXTOUT	Output	Default	Default	Top	Signal	Top	Signal	Default -> Top

As an example, let us look at instantiating a 16-bit Accumulator with synchronous data out (registered outputs). The example below shows the port mapping and parameters that need to be set. Setting the ports and parameters is based on the tables discussed in the sections above.

3.2.1. Dual 16 bit Accumulator with Sync Data Out

Verilog

```
SB_MAC16 i_sbmac16
  (      // port interfaces
  .A(A),
  .B(B),
  .C(C),
  .D(D),
  .O(O),
  .CLK(CLK),
  .CE(CE),
  .IRSTTOP(IRSTTOP),
  .IRSTBOT(IRSTBOT),
  .ORSTTOP(ORSTTOP),
  .ORSTBOT(ORSTBOT),
  .AHOLD(AHOLD),
  .BHOLD(BHOLD),
  .CHOLD(CHOLD),
  .DHOLD(DHOLD),
  .OHOLDTOP(OHOLDTOP),
  .OHOLDBOT(OHOLDBOT),
  .OLOADTOP(OLOADTOP),
  .OLOADBOT(OLOADBOT),
  .ADDSUBTOP(ADDSUBTOP),
  .ADDSUBBOT(ADDSUBBOT),
  .CO(CO),
  .CI(CI),
  .ACCUMCI(),
  .ACCUMCO(),
  .SIGNEXTIN(),
  .SIGNEXTOUT()
  );

  defparam i_sbmac16.NEG_TRIGGER = 1'b0;
  defparam i_sbmac16.C_REG = 1'b0;
  defparam i_sbmac16.A_REG = 1'b0;
  defparam i_sbmac16.B_REG = 1'b0;
  defparam i_sbmac16.D_REG = 1'b0;

  defparam i_sbmac16.TOP_8x8_MULT_REG = 1'b0;
  defparam i_sbmac16.BOT_8x8_MULT_REG = 1'b0;
  defparam i_sbmac16.PIPELINE_16x16_MULT_REG1 = 1'b0;
  defparam i_sbmac16.PIPELINE_16x16_MULT_REG2 = 1'b0;

  defparam i_sbmac16.TOPOUTPUT_SELECT =     2'b01; // accum register output at O[31:16]
  defparam i_sbmac16.TOPADDSUB_LOWERINPUT = 2'b00;
  defparam i_sbmac16.TOPADDSUB_UPPERINPUT = 1'b0;
  defparam i_sbmac16.TOPADDSUB_CARRYSELECT = 2'b00;
  defparam i_sbmac16.BOTOUTPUT_SELECT =     2'b01; // accum regsiter output at O[15:0]
  defparam i_sbmac16.BOTADDSUB_LOWERINPUT = 2'b00;
  defparam i_sbmac16.BOTADDSUB_UPPERINPUT = 1'b0;
  defparam i_sbmac16.BOTADDSUB_CARRYSELECT = 2'b00;
  defparam i_sbmac16.MODE_8x8 = 1'b0;

  defparam i_sbmac16.A_SIGNED = 1'b0;
  defparam i_sbmac16.B_SIGNED = 1'b0;

  //defparam i_sbmac16.BOTOUTPUT_SELECT = 2'b01 ;// accum regsiter output at O[15:0].
  //defparam i_sbmac16.TOPOUTPUT_SELECT = 2'b01 ;// accum register output at O[31:16]

  Endmodule
```

VHDL

```
      i_sbmac16: SBMAC16
      GENERIC MAP (
        NEG_TRIGGER => 1'b0,
        C_REG  => 1'b0,
        A_REG  => 1'b0,
        B_REG  => 1'b0,
        D_REG  => 1'b0,

        TOP_8x8_MULT_REG => 1'b0,
        BOT_8x8_MULT_REG => 1'b0,
        PIPELINE_16x16_MULT_REG1 => 1'b0,
        PIPELINE_16x16_MULT_REG2 => 1'b0,

        TOPOUTPUT_SELECT =>  2'b01, -- accum register output at O[31:16]
        TOPADDSUB_LOWERINPUT => 2'b00,
        TOPADDSUB_UPPERINPUT => 1'b0,
        TOPADDSUB_CARRYSELECT => 2'b00,

        BOTOUTPUT_SELECT =>  2'b01, -- accum regsiter output at O[15:0]
        BOTADDSUB_LOWERINPUT => 2'b00,
        BOTADDSUB_UPPERINPUT => 1'b0,
        BOTADDSUB_CARRYSELECT => 2'b00,
        MODE_8x8 => 1'b0,

        A_SIGNED => 1'b0,
        B_SIGNED => 1'b0
)
        PORT MAP (-- port interfaces
        A => A,
        B => B,
        C => C,
        D => D,
        O => O,
        CLK     =>      CLK,
        CE      =>      CE,
        IRSTTOP         =>      IRSTTOP,
        IRSTBOT         =>      IRSTBOT,
        ORSTTOP         =>      ORSTTOP,
        ORSTBOT         =>      ORSTBOT,
        AHOLD           =>      AHOLD,
        BHOLD           =>      BHOLD,
        CHOLD           =>      CHOLD,
        DHOLD           =>      DHOLD,
        OHOLDTOP        =>      OHOLDTOP,
        OHOLDBOT        =>      OHOLDBOT,
        OLOADTOP        =>      OLOADTOP,
        OLOADBOT        =>      OLOADBOT,
        ADDSUBTOP       =>      ADDSUBTOP,
        ADDSUBBOT       =>      ADDSUBBOT,
        CO      =>      CO,
        CI      =>      CI,
        ACCUMCI         =>      Open,
        ACCUMCO         =>      Open,
        SIGNEXTIN       =>      Open,
        SIGNEXTOUT      =>      Open
        );
```

Another common function used for DSP applications is a multiplier. The two examples below show the instantiation of 16-bit multipliers both signed and unsigned.

3.2.2. Multiplier 16 x 16 Signed

Verilog

```
SB_MAC16 i_sbmac16
  (      // port interfaces
  .A(A),
  .B(B),
  .C(C),
  .D(D),
  .O(O),
  .CLK(CLK),
  .CE(CE),
  .IRSTTOP(IRSTTOP),
  .IRSTBOT(IRSTBOT),
  .ORSTTOP(ORSTTOP),
  .ORSTBOT(ORSTBOT),
  .AHOLD(AHOLD),
  .BHOLD(BHOLD),
  .CHOLD(CHOLD),
  .DHOLD(DHOLD),
  .OHOLDTOP(OHOLDTOP),
  .OHOLDBOT(OHOLDBOT),
  .OLOADTOP(OLOADTOP),
  .OLOADBOT(OLOADBOT),
  .ADDSUBTOP(ADDSUBTOP),
  .ADDSUBBOT(ADDSUBBOT),
  .CO(CO),
  .CI(CI),
  .ACCUMCI(),
  .ACCUMCO(),
  .SIGNEXTIN(),
  .SIGNEXTOUT()
  );
   defparam i_sbmac16.TOPOUTPUT_SELECT      = 2'b11; //Mult16x16 data output
   defparam i_sbmac16.BOTOUTPUT_SELECT      = 2'b11;
   defparam i_sbmac16.PIPELINE_16x16_MULT_REG2 = 1'b1;//Mult16x16 output registered
   defparam i_sbmac16.A_SIGNED       = 1'b1; //Signed Inputs
   defparam i_sbmac16.B_SIGNED       = 1'b1;
endmodule
```

VHDL

```
i_sbmac16: SB_MAC16
GENERIC MAP (
  TOPOUTPUT_SELECT     => 2'b11,
  BOTOUTPUT_SELECT     => 2'b11,
  PIPELINE_16x16_MULT_REG2     => 1'b1,
  A_SIGNED      => 1'b1,
  B_SIGNED      => 1'b1


  PORT MAP (
  A =>   A,
  B =>   B,
  C =>   C,
  D =>   D,
  O =>   O,
  CLK    =>     CLK,
  CE     =>     CE,
  IRSTTOP       =>     IRSTTOP,
  IRSTBOT       =>     IRSTBOT,
  ORSTTOP       =>     ORSTTOP,
  ORSTBOT       =>     ORSTBOT,
  AHOLD         =>     AHOLD,
  BHOLD         =>     BHOLD,
  CHOLD         =>     CHOLD,
  DHOLD         =>     DHOLD,
  OHOLDTOP      =>     OHOLDTOP,
  OHOLDBOT      =>     OHOLDBOT,
  OLOADTOP      =>     OLOADTOP,
  OLOADBOT      =>     OLOADBOT,
  ADDSUBTOP     =>      ADDSUBTOP,
  ADDSUBBOT     =>      ADDSUBBOT,
  CO     =>     CO,
  CI     =>     CI,
  ACCUMCI       =>     Open,
  ACCUMCO       =>     Open,
  SIGNEXTIN     =>     Open,
  SIGNEXTOUT    =>     Open
  );
```

3.2.3. Multiplier 16 x 16 Unsigned

Verilog

```
     SB_MAC16 i_sbmac16
       (      // port interfaces
       .A(A),
       .B(B),
       .C(C),
       .D(D),
       .O(O),
       .CLK(CLK),
       .CE(CE),
       .IRSTTOP(IRSTTOP),
       .IRSTBOT(IRSTBOT),
       .ORSTTOP(ORSTTOP),
       .ORSTBOT(ORSTBOT),
       .AHOLD(AHOLD),
       .BHOLD(BHOLD),
       .CHOLD(CHOLD),
       .DHOLD(DHOLD),
       .OHOLDTOP(OHOLDTOP),
       .OHOLDBOT(OHOLDBOT),
       .OLOADTOP(OLOADTOP),
       .OLOADBOT(OLOADBOT),
       .ADDSUBTOP(ADDSUBTOP),
       .ADDSUBBOT(ADDSUBBOT),
       .CO(CO),
       .CI(CI),
       .ACCUMCI(),
       .ACCUMCO(),
       .SIGNEXTIN(),
       .SIGNEXTOUT()
       );

defparam i_sbmac16.TOPOUTPUT_SELECT       = 2'b11;
defparam i_sbmac16.BOTOUTPUT_SELECT       = 2'b11;
defparam i_sbmac16.PIPELINE_16x16_MULT_REG2 = 1'b1;
defparam i_sbmac16.A_SIGNED = 1'b0;
defparam i_sbmac16.B_SIGNED = 1'b0;

endmodule
```

VHDL

```
i_sbmac16: SB_MAC16
GENERIC MAP (
  TOPOUTPUT_SELECT     =>     2'b11,
  BOTOUTPUT_SELECT     =>     2'b11,
  PIPELINE_16x16_MULT_REG2    =>     1'b1,
  A_SIGNED      =>     1'b0,
  B_SIGNED      =>     1'b0
PORT MAP (
  A => A,
  B => B,
  C => C,
  D => D,
  O => O,
  CLK     =>    CLK,
  CE      =>    CE,
  IRSTTOP       =>     IRSTTOP,
  IRSTBOT       =>     IRSTBOT,
  ORSTTOP       =>     ORSTTOP,
  ORSTBOT       =>     ORSTBOT,
  AHOLD         =>     AHOLD,
  BHOLD         =>     BHOLD,
  CHOLD         =>     CHOLD,
  DHOLD         =>     DHOLD,
  OHOLDTOP      =>     OHOLDTOP,
  OHOLDBOT      =>     OHOLDBOT,
  OLOADTOP      =>     OLOADTOP,
  OLOADBOT      =>     OLOADBOT,
  ADDSUBTOP     =>     ADDSUBTOP,
  ADDSUBBOT     =>     ADDSUBBOT,
  CO     =>     CO,
  CI     =>     CI,
  ACCUMCI       =>     Open,
  ACCUMCO       =>     Open,
  SIGNEXTIN     =>     Open,
  SIGNEXTOUT    =>     Open
);
```

Technical Support Assistance

Submit a technical support case via www.latticesemi.com/techsupport.

Revision History

Revision 1.2, October 2020

Section	Change Summary
All	• Changed document number from TN1295 to FPGA-TN-02007. • Updated document template.
Disclaimers	Added this section.
Introduction	Updated list of functional units.
DSP Primitive – SB_MAC16	• Added reference to iCE Technology Library in the SB_MAC16 Parameters section. • Updated IRSTTOP, ORSTTOP, IRSTBOT, ORSTBOT, and ACCUMCO descriptions in Table 2.1. SB_MAC16 Ports and their Functional Descriptions. • Updated BOTADDSUB_LOWERINPUT description in Table 2.2. SB_MAC16 Parameter Description. • Updated Figure 2.2. SB_MAC16 DSP Functional Diagram.
Implementing DSP Functions	• Updated section heading to Implementing DSP Functions. • Updated sub-section heading to Dual 16 bit Accumulator with Sync Data Out. • General update to section introduction. • Updated Inferencing DSP Functions section. • Updated Table 3.1. Instantiation Guide. • Fixed code errors.
—	• Editorial changes to correct grammar and improve readability. • Minor adjustments in style/formatting.

Revision 1.1, June 2016

Section	Change Summary
Introduction	Updated Introduction section. Added iCE40 UltraPlus and removed *MX series* to introductory paragraph.
DSP Primitive – SB_MAC16	Updated DSP Primitive – SB_MAC16 section. Added iCE40 UltraPlus and removed *MS series* to introductory paragraph
Implementing DSP Function in iCE40 Ultra and iCE40 UltraPlus Devices	Updated Implementing DSP Function in iCE40 Ultra and iCE40 UltraPlus Devices section. • Revised section heading to include iCE40 UltraPlus. • Added iCE40 UltraPlus to introductory sentence.
Technical Support Assistance	Updated Technical Support Assistance section.

Revision 1.0, June 2014

Section	Change Summary
All	Initial release.

www.latticesemi.com