DSP Function Usage Guide for iCE40 Devices

June 2014 Technical Note TN1295

Introduction

This technical note discusses DSP function usage for the iCE40™ device family, specifically iCE40 Ultra™. It is intended to be used as a guide to 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 MX series iCE40 Ultra devices. This block can be configured into combination of following functional units by selecting appropriate parameter values.

- Single 16x16 Multiplier (generating 32-bit product output).
- Two independent 8x8 Multiplier (generating two independent 16-bit product output).
- Single 32-bit Accumulator.
- Two independent 16-bit Accumulator.
- Single 32-bit Adder/ Subtractor.
- Two independent 16-bit Adder/ Subtractor.
- Single 32-bit Multiply-Add/ Multiply-Sub.
- Two independent 16-bit Multiply-Add/ Multiply-Sub.

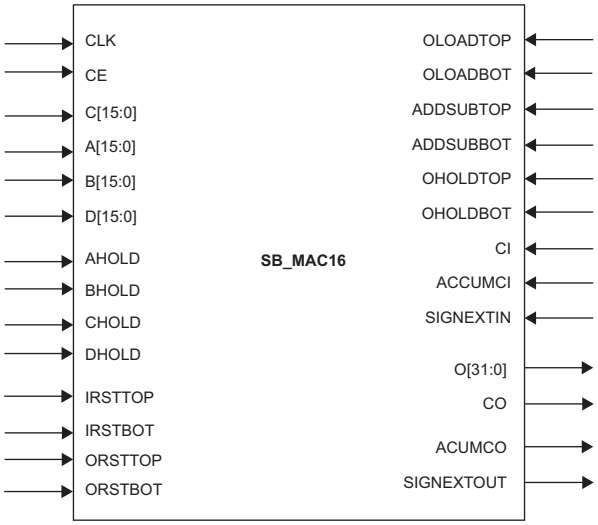
DSP Primitive – SB_MAC16

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

SB_MAC16 Primitive

Figure 1 provides an overview of the SB_MAC16 primitive with various inputs and outputs.

Figure 1. SB_MAC16 DSP Primitive Interface Diagram



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

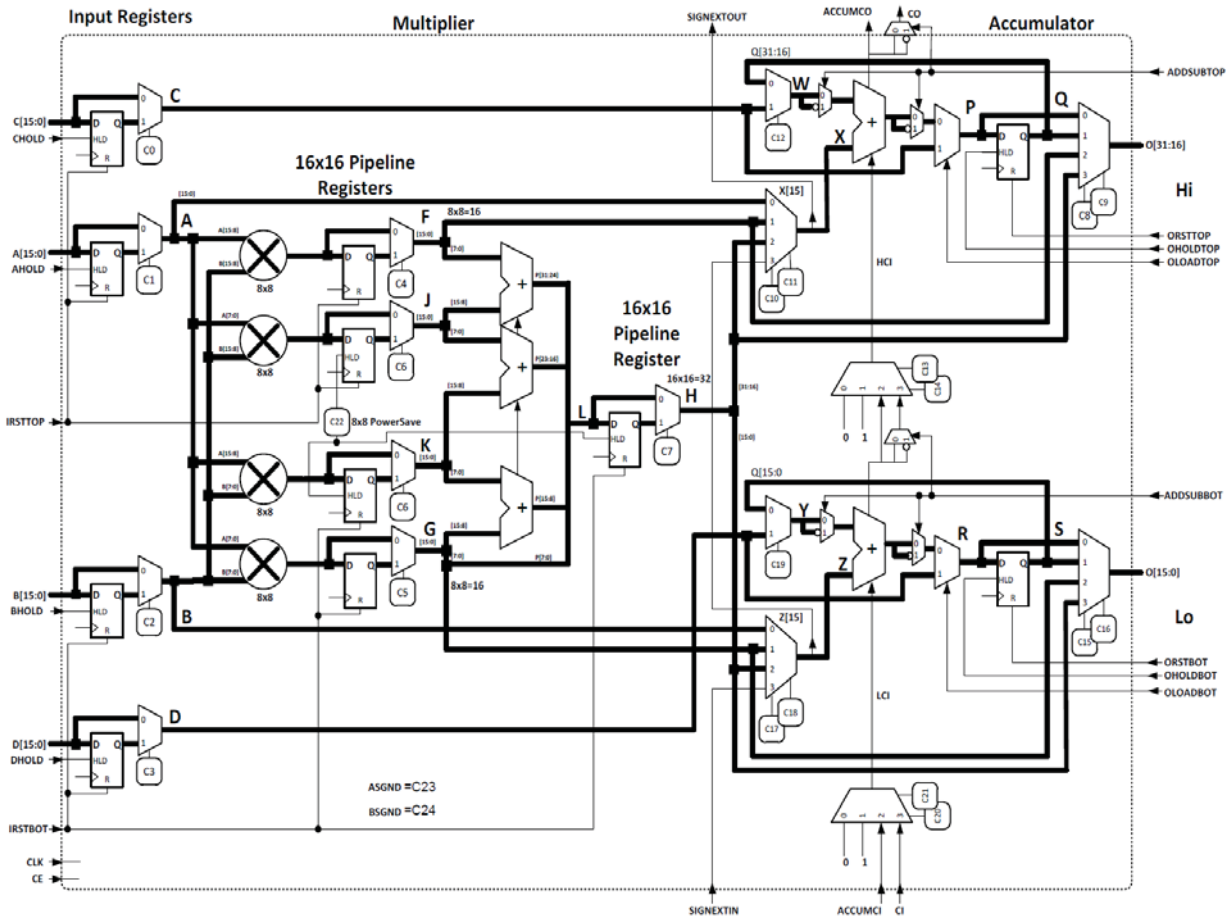
- Registered Inputs/ Outputs
  - The inputs to the functional units can be either registered or unregistered.
  - The outputs from the functional units can be either registered or unregistered.
  - The intermediate multiplier outputs can be pipelined for faster clock performance.
- Signed/ Unsigned Inputs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

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

SB_MAC16 Functional Diagram

Figure 2 shows the functional model of the SB_MAC16 primitive. The variety of functions can be implemented in this block by interfacing with portions required of the functional model that are needed for these functions.

Figure 2. SB_MAC16 DSP Functional Model



SB_MAC16 Interface Ports

The following table, Table 1, provides a list of interface ports available in SB_MAC16 and their functional description. Something of importance to note in this table is the “Default Values” of these ports; as it will be 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 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 8x8 Multiplier Output Register (8x8 MAC Pipeline Register) 0: Not reset 1: Reset	0
ORSTTOP	Input	Reset Input to top Accumulator Register (for Adder/Subtractor, Accumulator, and MAC functions) 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

Port Name	Direction	Functional Description	Default Values
OUTPUT[31:16]	Output	Upper 16 bits of Output	
IRSTBOT	Input	Reset Input to Registers A and C. Also resets upper 8x8 Multiplier Output Register (8x8 MAC Pipeline Register) and the 16x16 Multiplier Output Register (16x16 MAC Pipeline Register) 0: Not reset 1: Reset	0
ORSTBOT	Input	Reset Input to top Accumulator Register (for Adder/Subtractor, Accumulator, and MAC functions) 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 previous DSP block	
SIGNEXTIN	Input	Sign Extension Input from previous DSP block	0
SIGNEXTOUT	Output	Sign Extension Output to next DSP block	

SB_MAC16 Parameters

The parameter table below, Table 2, shows a list of parameters to configure the SV_MAC16 block. This table also maps the parameter to the configuration bits shows in the SB_MAC16 Functional Diagram in Figure 2.

Table 2. SB_MAC16 Parameter Description

Parameter Name	Configuration Bit(s)	Parameter Description & 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 8x8 Multiplier Output Register Control (Pipeline Register for MAC): 0: Not registered 1: Registered	0
BOT_8x8_MULT_REG	C5	Bottom 8x8 Multiplier Output Register Control (Pipeline Register for MAC): 0: Not registered 1: Registered	0
PIPELINE_16X16_MULT_REG1	C6	16x16 Multiplier Pipeline Register Control: 0: Not registered 1: Registered	0
PIPELINE_16x16_MULT_REG2	C7	16x16 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: 8x8 Multiplier 11: 16x16 Multiplier	00
TOPADDSUB_LOWERINPUT	C11, C10	Input X of upper Adder/Subtractor: 00: Input A 01: 8x8 Multiplier Output at Top 10: 16x16 Multiplier upper 16-bit outputs 11: Sign extension from Z15 (lower Adder/Subtractor input)	00

Parameter Name	Configuration Bit(s)	Parameter Description & Allowed Values	Default
TOPADDSUB_UPPERINPUT	C12	Input W of upper Adder/Subtractor: 0: Output of Adder/Subtractor Output Register (Accumulation Function) 1: Input C	0
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: 8x8 Multiplier 11: 16x16 Multiplier	00
BOTADDSUB_LOWERINPUT	C18, C17	Input Z of upper Adder/Subtractor: 00: Input B 01: 8x8 Multiplier Output at Top 10: 16x16 Multiplier upper 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 8x8 Multiplier Mode (Power Saving): 0: Not Selected 1: Selected	 0 --> 1
A_SIGNED	C23	Input A Sign: 0: Input A is un-signed 1: Input A is signed	 0
B_SIGNED	C24	Input B Sign: 0: Input B is un-signed 1: Input B is signed	 0

Implementing DSP Function in iCE40 Ultra Devices

There are two ways to implement the DSP function in the iCE40 Ultra devices:

- Inferencing DSP functions
  This method requires users to define the functional behavior of the DS function they wish to implement, and the tools map and place it to the DSP block SB_MAC16.
- Instantiating DSP Primitive SB_MAC16
  This method involves the 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. The detailed discussion of the methodology is discussed below.

Both these methods are discussed in details in following sections.

Inferencing DSP Functions

This method involves defining desired DSP function as a behavioral model in the standard HDL. The code does not requires you to know the details of the DSP primitive, and 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.

32-bit Accumulator with Async Data In & 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 ;
is

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 8x8 multiplier, with both inputs and outputs registered.

8x8 Multiplier, Unsigned with Sync Data In & 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);
BEGIN

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;
```

Instantiation DSP Primitive – SB_MAC16

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

Table 3 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 references 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 1.

In certain cases, the DSP block can have two independent functions, for example two 8x8 multipliers, generating two 16-bit outputs. Such cases are referenced as Top and Bottom Signals in the table below. In such cases, one of the 8x8 multipliers can be implemented using Top Signals, and other using Bottom Signals.

Table 3. Instantiation Guide

Port Name	Input/ Output	8x8 Multiplier	16x16 Multiplier	16x16 Accumulate	32x32 Accumulate	16x16 Adder / Subtractor	32x32 Adder / Subtractor	8x8 MAC	16x16 MAC
CLK	Input	Signal	Signal	Signal	Signal	Signal	Signal	Signal	Signal
CE	Input	Signal	Signal	Signal	Signal	Signal	Signal	Signal	Signal
A[7:0]	Input	Bottom	Signal	Top	Signal	Top	Signal	Bottom	Signal
A[15:8]	Input	Top	Signal	Top	Signal	Top	Signal	Top	Signal
AHOLD	Input	Signal	Signal	Top -> Signal	Signal	Top -> Signal	Signal	Signal	Signal
B[7:0]	Input	Bottom	Signal	Bottom	Signal	Bottom	Signal	Bottom	Signal
B[15:8]	Input	Top	Signal	Bottom	Signal	Bottom	Signal	Top	Signal
BHOLD	Input	Signal	Signal	Bottom - Signal	Signal	Bottom - Signal	Signal	Signal	Signal
C[15:0]	Input	Default	Default	Default -> Top	Default -> Signal	Top	Default -> Signal	Top	Signal
CHOLD	Input	Default	Default	Default -> Top	Default -> Signal	Top	Default -> Signal	Top	Signal
D[15:0]	Input	Default	Default	Default -> Bottom	Default -> Signal	Bottom	Default -> Signal	Bottom	Signal
DHOLD	Input	Default	Default	Default -> Bottom	Default -> Signal	Bottom	Default -> Signal	Bottom	Signal
IRSTTOP	Input	Top -> Signal	Signal	Top -> Signal	Signal	Top -> Signal	Signal	Top -> Signal	Signal
ORSTTOP	Input	Top	Signal	Top	Signal	Top	Signal	Top	Signal
OLOADTOP	Input	Default	Default	Signal	Signal	0	0	Signal	Signal
ADDSUBTOP	Input	Default	Default	0	0	0 = Add 1 = Sub	0 = Add 1 = Sub	0	0
OHOLDTOP	Input	Top -> default	Signal - default	Top	Signal	Top	Signal	Top	Signal
OUTPUT[31:16]	Output	Top	Signal	Top	Signal	Top	Signal	Top	Signal
IRSTBOT	Input	Bottom -> Signal	Signal	Bottom -> Signal	Signal	Bottom -> Signal	Signal	Bottom -> Signal	Signal
ORSTBOT	Input	Bottom	Signal	Bottom	Signal	Bottom	Signal	Bottom	Signal
OLOADBOT	Input	Default	Default	Signal	Signal	0	0	Signal	Signal
ADDSUBBOT	0 = Add 1 = Sub	0 = Add 1 = Sub	0	0	0 = Add 1 = Sub	0 = Add 1 = Sub	0 = Add 1 = Sub	0	0
OHOLDBOT	Input	Bottom -> default	Signal -> default	Bottom	Signal	Bottom	Signal	Bottom	Signal
OUTPUT[15:0]	Output	Bottom	Signal	Bottom	Signal	Bottom	Signal	Bottom	Signal
CI	Input	Default	Default	Default	Default	Bottom	Signal	Default	Default
CO	Output	Default	Default	Default -> Top	Default -> Signal	Top	Signal	Default -> Top	Default -> Signal
ACCUMCI	Input	Default	Default	Bottom	Signal	Default -> Bottom	Default -> Signal	Default -> Bottom	Default -> Signal
ACCUMCO	Output	Default	Default	Top	Signal	Default -> Top	Default -> Signal	Default -> Top	Default -> Signal
SIGNEXTIN	Input	Default	Default	Bottom	Signal	Bottom	Signal	Default -> Bottom	Default -> Signal
SIGNEXTOUT	Output	Default	Default	Top	Signal	Top	Signal	Default -> Top	Default -> Signal

As an example, let us look at instantiation sections for 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 instantiations will be based on the tables discussed in the sections above.

Accumulator 16x2 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 modules used for DSP applications is a multiplier. The two examples below shows the instantiation of 16-bit multipliers both signed and unsigned.

Multiplier 16x16 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
       );
```

Multiplier 16x16 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

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
June 2014	1.0	Initial release.