LatticeXP2 sysDSP Usage Guide

February 2007 **Technical Note TN1140**

Introduction

This technical note discusses how to access the features of the LatticeXP2™ sysDSP™ (Digital Signal Processing) Block described in the LatticeXP2 Family Data Sheet. Designs targeting the sysDSP Block can offer significant improvement over traditional LUT-based implementations. Table 13-1 provides an example of the performance and area benefits of this approach:

Table 13-1. sysDSP Block vs. LUT-based Multipliers

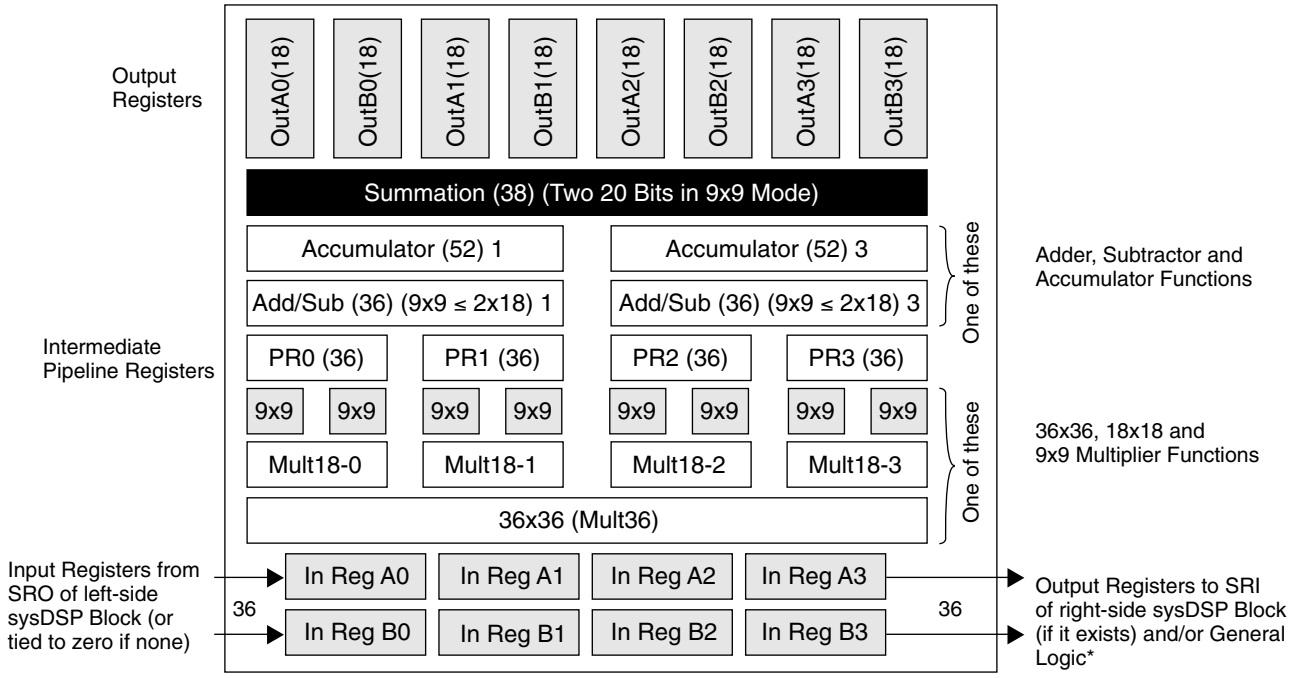
Multiplier Width	Register Pipelining	XP2-17-7 Uses One sysDSP Block		XP2-17-7 Uses LUTs	
		f_{MAX} (MHz)[1]	LUTs	f_{MAX} (MHz)[1]	LUTs
9x9	Input, Multiplier, Output	365	0	103	192
18x18	Input, Multiplier, Output	365	0	76	698
36x36	Input, Multiplier, Output	323	0	50	2732

1. These timing numbers were generated using the ispLEVER® design tool. Exact performance may vary with design and tool version.

sysDSP Block Hardware

The LatticeXP2 sysDSP Blocks are located in rows throughout device. Below is a block diagram of one of the sysDSP Blocks:

Figure 13-1. LatticeXP2 sysDSP Block



*Can only be routed to general logic routing when configured with less than three MULT18X18.

Note: Each sysDSP Block spans nine columns of PFUs.

The sysDSP Block can be configured as:

- One 36x36 Multiplier
  - Basic multiplier, no add/sub/accum/sum blocks
- Four 18x18 Multipliers
  - Two add/sub/accum blocks
  - One summation block for adding four multipliers
- Eight 9x9 Multipliers
  - Four add/sub blocks
  - Two summation blocks

Note that a sysDSP block can only be configured in one mode at a time.

sysDSP Block Software

Overview

The sysDSP Block of the LatticeXP2 device can be targeted in a number of ways.

- The IPexpress™ tool in the ispLEVER software allows the rapid creation of modules implementing sysDSP elements. These modules can then be used in HDL designs as appropriate.
- The coding of certain functions into a design's HDL and allowing the synthesis tools to Inference the use of a sysDSP block.
- The implementation of designs in The MathWorks® Simulink® tool using a Lattice block set. The ispLEVER sysDSP portion of the ispLEVER design tools will then convert these blocks into HDL as appropriate.
- Instantiation of sysDSP primitives directly in the source code

Targeting sysDSP Block Using IPexpress

IPexpress allows you to graphically specify sysDSP elements. Once the element is specified, a HDL file is generated, which can be instantiated in a design. IPexpress allows users to configure all ports and set all available parameters. The following modules target the sysDSP Block. For design examples using IPexpress, refer to EXAMPLES in the ispLEVER Software (from the Project Navigator pull down-menu, go to **File** and open **Example**). The following four types of elements can be specified via IPexpress:

- MULT (Multiplier)
- MAC (Multiplier Accumulate)
- MULTADDSUB (Multiplier Add/Subtract)
- MULTADDSUBSUM (Multiply Add/Subtract and SUM)

MULT Module

The MULT Module configures elements to be packed into the sysDSP primitives. The Basic mode screen illustrated in Figure 13-2 consists of an optional one clock, one clock enable and one reset tied to all registers. Multiple sysDSP Blocks can be spanned to accommodate large multiplications. Additional LUTs may be required if multiple sysDSP blocks are needed. Select **Area/Speed** to determine the LUT implementation. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register, which is useful in applications such as the FIR filter. The Advanced mode screen, illustrated in Figure 13-3, allows finer control over the register. In the Advanced mode, users can control each register with independent clocks, clock enables and resets. MULT inputs can be from 2 to 72 bits.

Figure 13-2. MULT Mode Basic Set-up

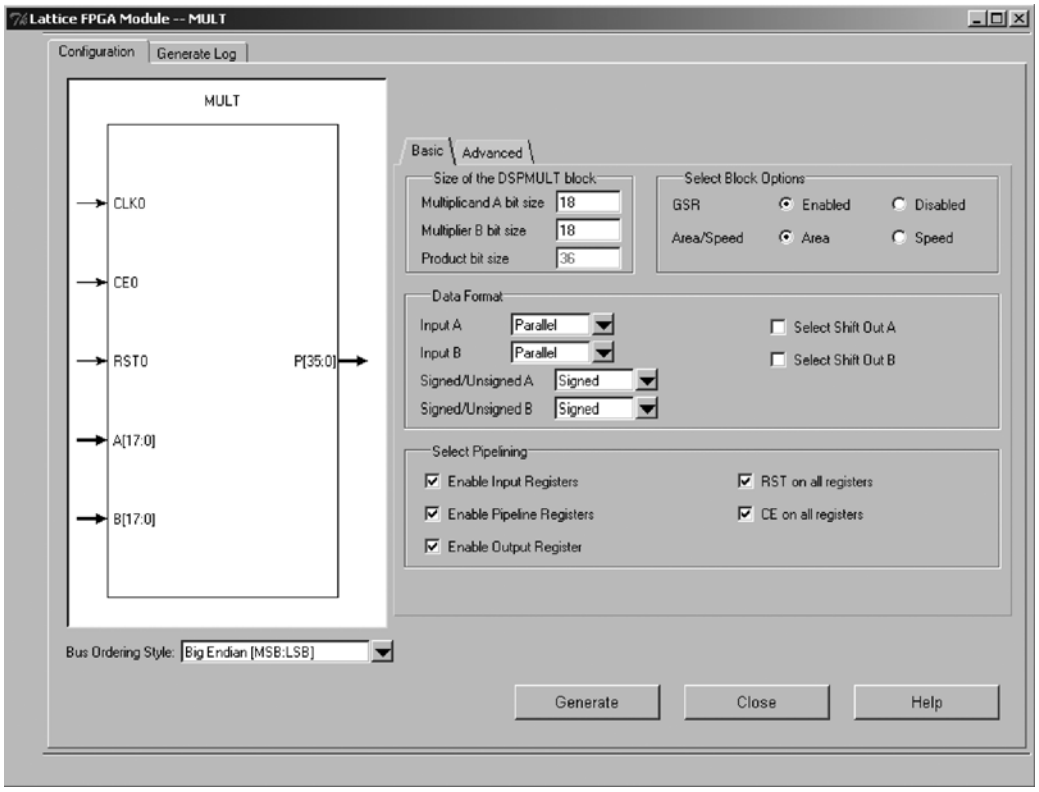
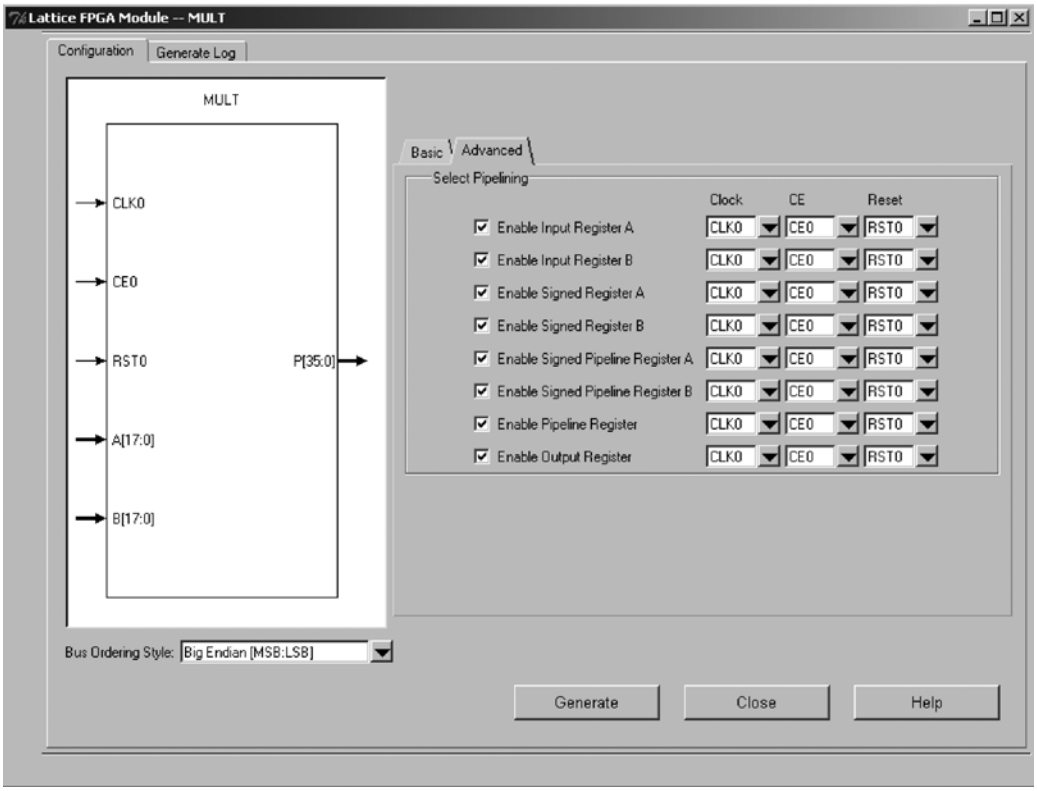


Figure 13-3. MULT Mode Advanced Set-up



MAC Module

The MAC Module configures multiply accumulate elements to be packed into the primitive MULT18X18MACB. The Basic mode, shown in Figure 13-4, consists of an optional one clock, one clock enable and one reset tied to all registers. Because of the accumulator, the output register is automatically enabled. Multiple sysDSP Blocks can be spanned to accommodate large multiplications. The accumulator of the sysDSP block is 52 bits deep and additional LUTs can be used if a larger accumulation is required. If sysDSP blocks are spanned, additional LUT logic may be required. Select **Area/Speed** to determine the LUT implementation. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register. The Accumsload loads the accumulator with the value from the LD port. This is required to initialize and load the first value of the accumulation. The Advanced mode, shown in Figure 13-5, allows finer control over the registers. In the advanced mode, users can control each register with independent clocks, clock enables and resets. MAC inputs can be from 2 to 72 bits.

Figure 13-4. MAC Mode Basic Set-up

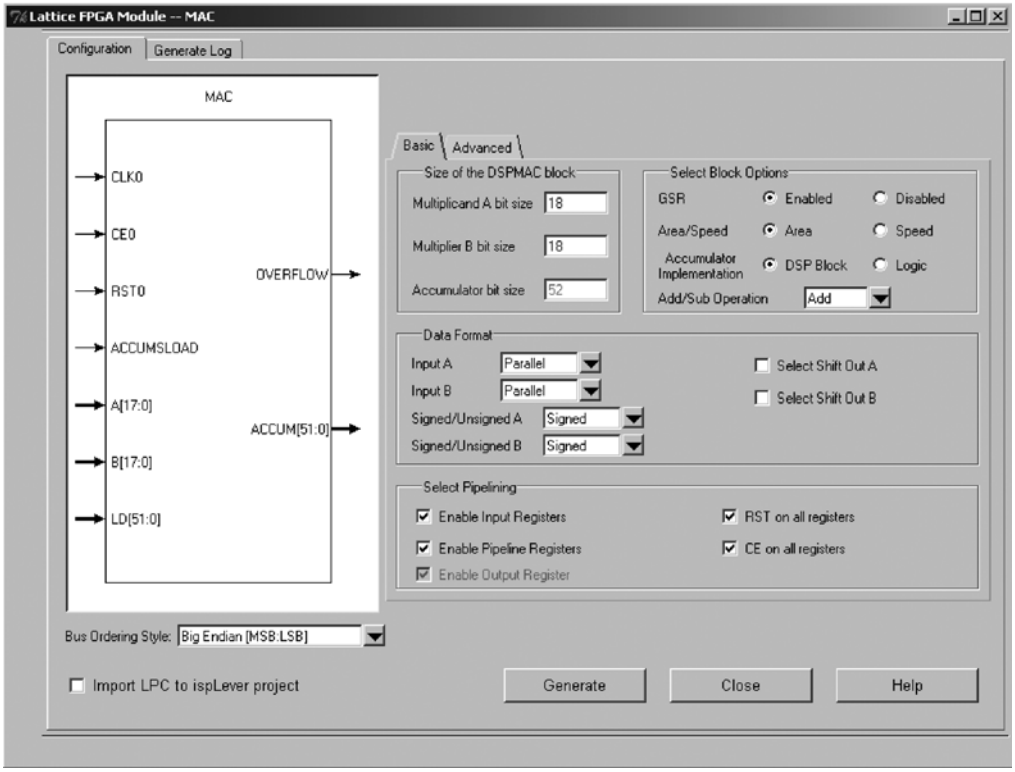
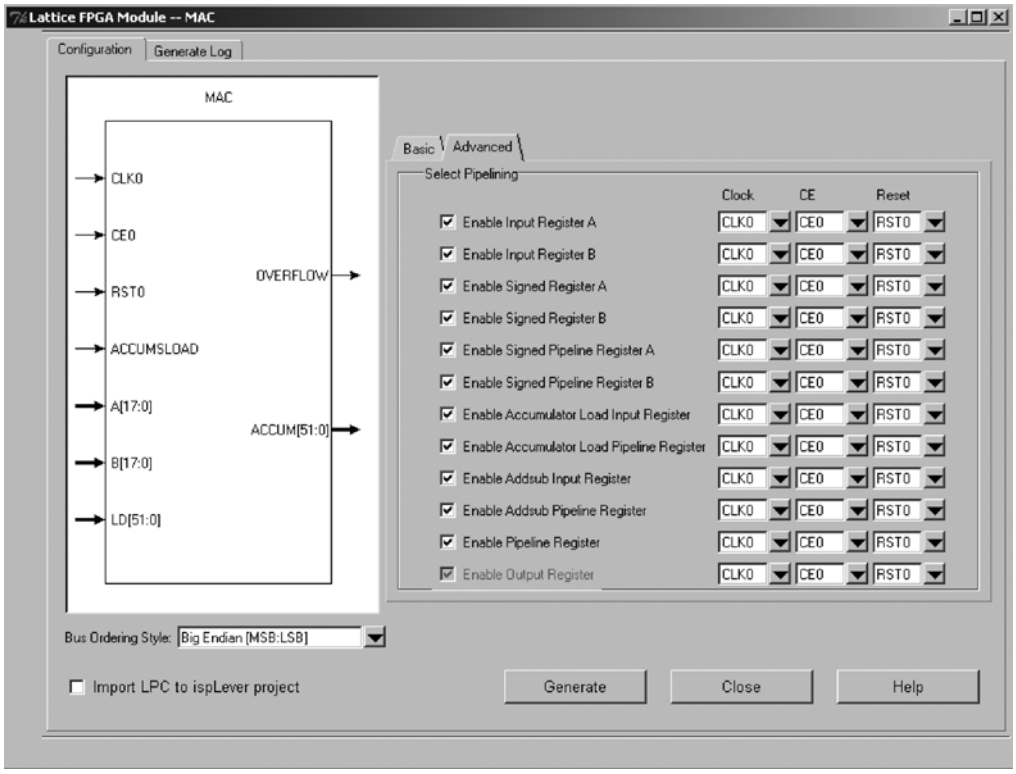


Figure 13-5. MAC Mode Advanced Set-up



MULTADDSUB Module

The MULTADDSUB GUI configures multiplier addition/subtraction elements to be packed into the primitives MULT18X18ADDSUBB or MULT9X9ADDSUBB. The Basic mode, shown in Figure 13-6, consists of an optional one clock, one clock enable and one reset tied to all registers. Multiple sysDSP Blocks can be spanned to accommodate large multiplications. If sysDSP blocks are spanned, additional LUT logic may be required. Select **Area/Speed** to determine the LUT implementation. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register, which is useful in applications such as the FIR filter. The Advanced mode, shown in Figure 13-7, provides finer control over the registers. In the advanced mode, users can control each register with independent clocks, clock enables and resets. MULTADDSUB inputs can be from 2 to 72 bits.

Figure 13-6. MULTADDSUB Mode Basic Set-up

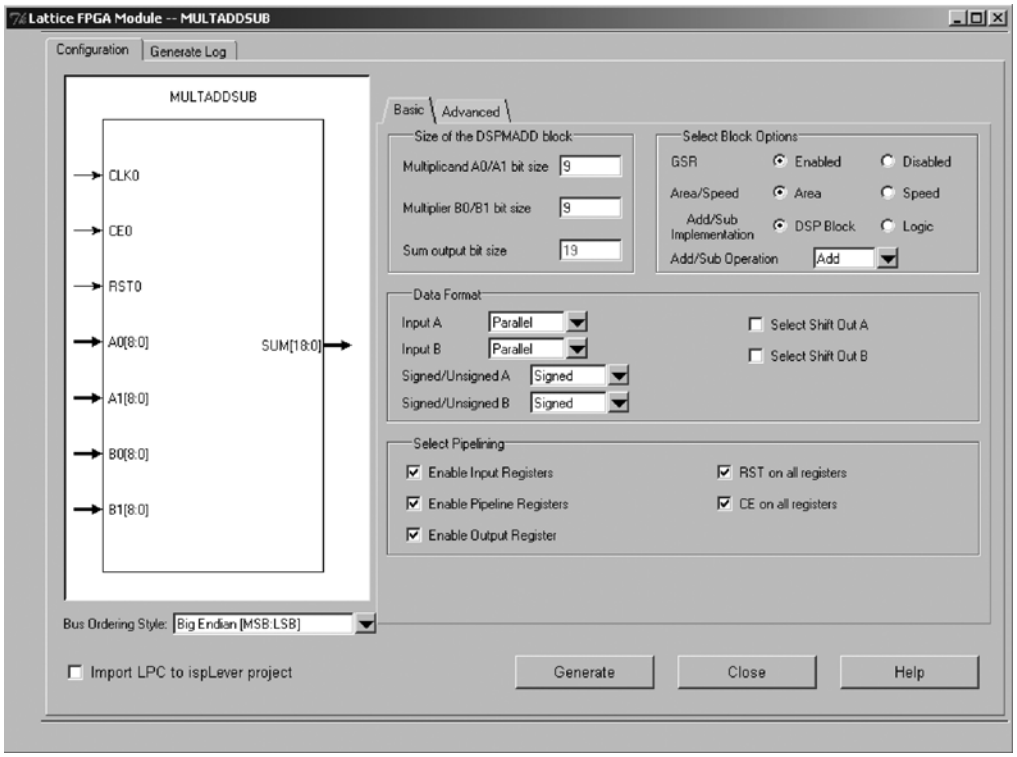
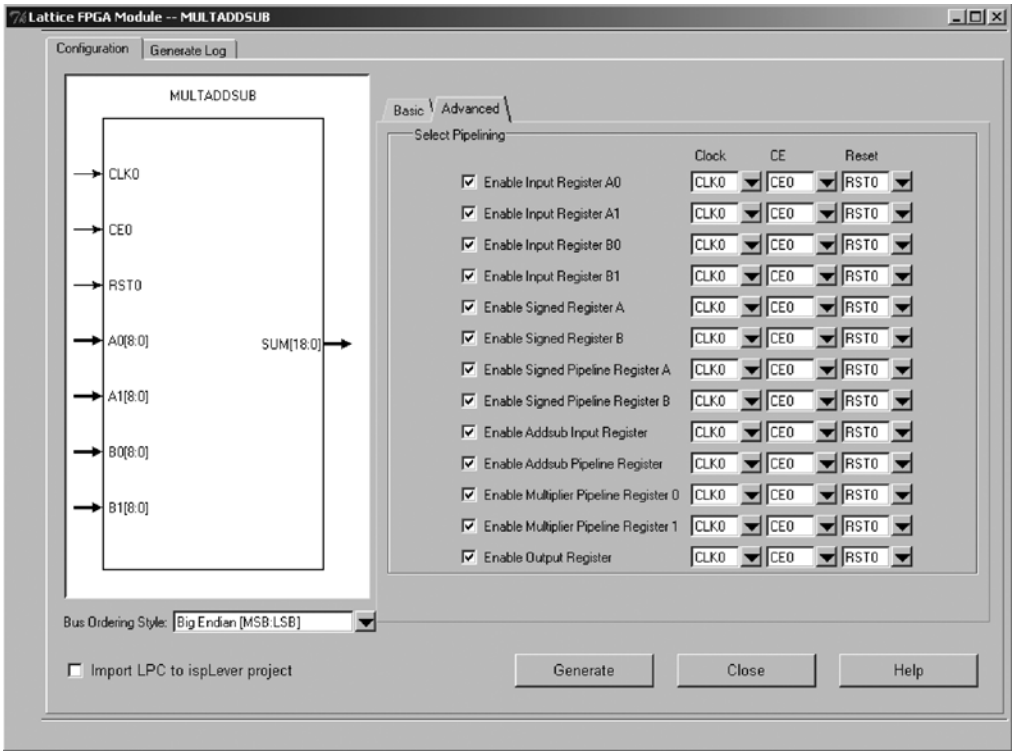


Figure 13-7. MULTADDSUB Mode Advanced Set-up



MULTADDSUBSUM Module

The MULTADDSUBSUM GUI configures Multiplier Addition/Subtraction Addition elements to be packed into the primitives MULT18X18ADDSUBSUMB or MULT9X9ADDSUBSUMB. The Basic mode, shown in Figure 13-8, consists of an optional one clock, one clock enable and one reset tied to all registers. Multiple sysDSP Blocks can be spanned to accommodate large multiplications. If sysDSP blocks are spanned, additional LUT logic may be required. Select **Area/Speed** to determine the LUT implementation. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format is can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register, which is useful in applications such as the FIR filter. The Advanced mode, shown in Figure 13-9, provides finer control over the registers. In the advanced mode, users can control each register with independent clocks, clock enables and resets. MULTADDSUBSUM inputs can be from 2 to 72 bits.

Figure 13-8. MULTADDSUBSUM Mode Basic Set-up

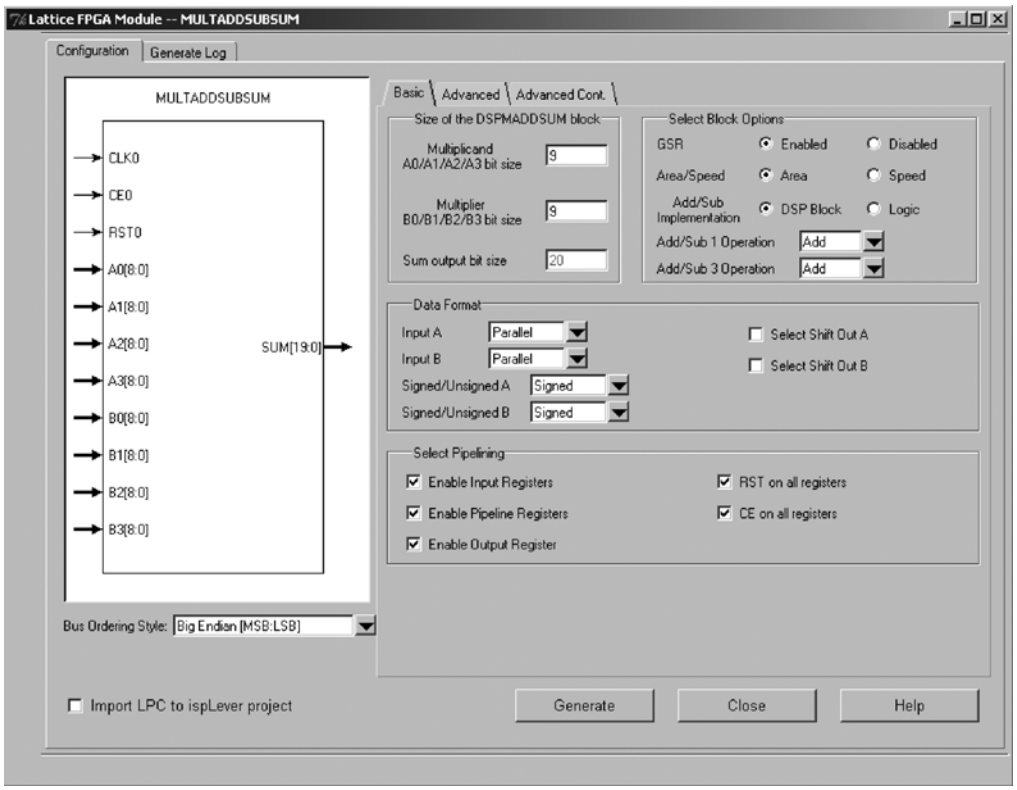


Figure 13-9. MULTADDSUBSUM Mode Advance

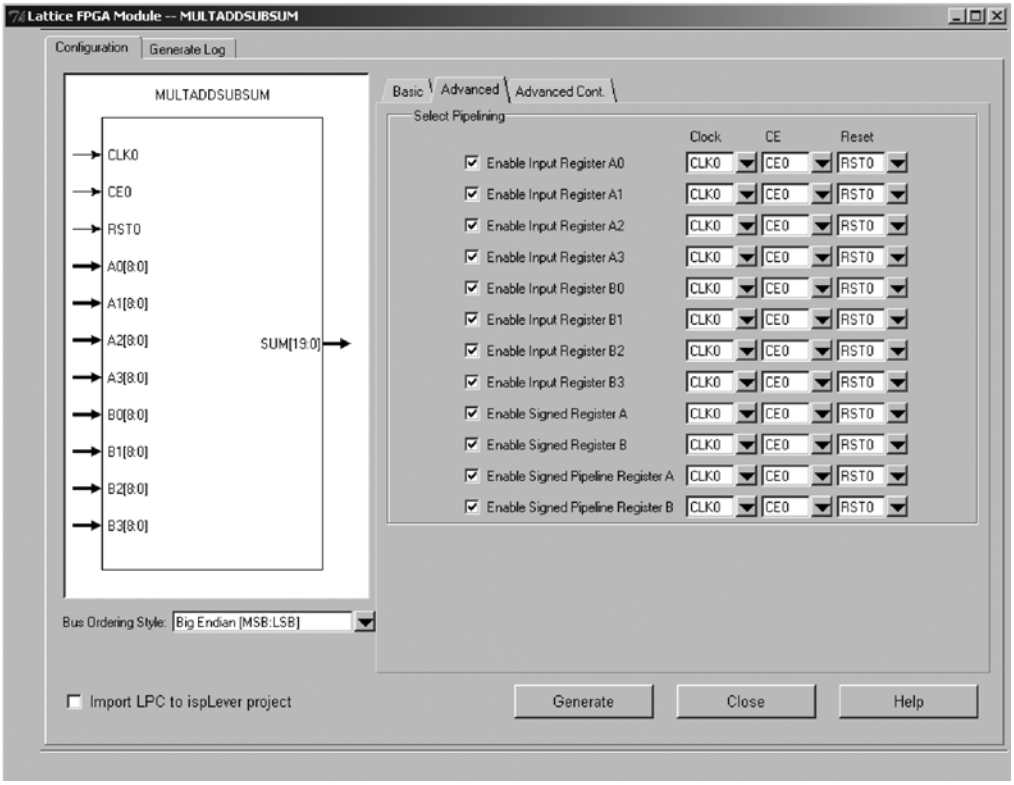
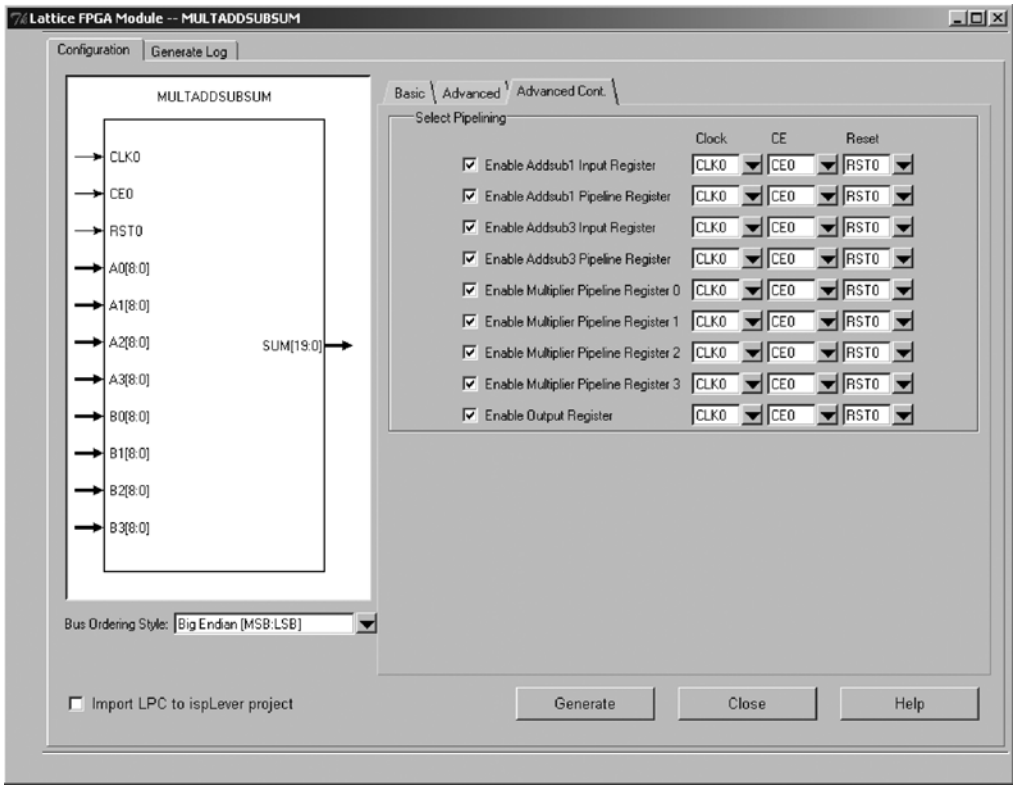


Figure 13-10. MULTADDSUBSUM Mode Advance



Targeting the sysDSP Block by Inference

The Inferencing flow enables the design tools to infer sysDSP Blocks from a HDL design. It is important to note that when using the Inferencing flow, unless the code style matches the sysDSP Block, results will not be optimal. Consider the following Verilog and VHDL examples:

```
// This Verilog example will be mapped into single MULT18X18MACB with the output register enabled
module mult_acc (dataout, dataax, dataay, clk);
   output [16:0] dataout;
   input [7:0] dataax, dataay;
   input clk;
   reg  [16:0] dataout;

   wire [15:0] multa = dataax * dataay; // 9x9 Multiplier
   wire [16:0] adder_out;
   assign adder_out = multa + dataout; // Accumulator
   always @(posedge clk)
   begin
     dataout <= adder_out; // Output Register of the Accumulator
   end
endmodule

-- This VHDL example will be mapped into single MULT18X18MACB with all the registers enabled
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity mac is
port (clk, reset : in std_logic;
      dataax, dataay : in std_logic_vector(8 downto 0);
      dataout : out std_logic_vector(17 downto 0));
end;

architecture arch of mac is
signal dataax_reg, dataay_reg : std_logic_vector(8 downto 0);
signal multout, multout_reg : std_logic_vector(17 downto 0);
signal addout : std_logic_vector(17 downto 0);
signal dataout_reg : std_logic_vector(17 downto 0);
begin

dataout <= dataout_reg;

process (clk, reset)
begin
if (reset = '1') then
  dataax_reg <= (others => '0');
  dataay_reg <= (others => '0');
elsif (clk'event and clk='1') then
  dataax_reg <= dataax;
  dataay_reg <= dataay;
end if;
end process;

multout <= dataax_reg * dataay_reg;

process (clk, reset)
begin
if (reset = '1') then
  multout_reg <= (others => '0');
elsif (clk'event and clk='1') then
  multout_reg <= multout;
end if;
```

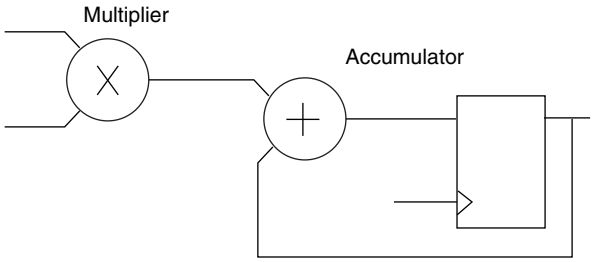
```
end process;

addout <= multout_reg + dataout_reg;

process (clk, reset)
begin
if (reset = '1') then
  dataout_reg <= (others => '0');
elsif (clk'event and clk='1') then
  dataout_reg <= addout;
end if;
end process;
end arch;
```

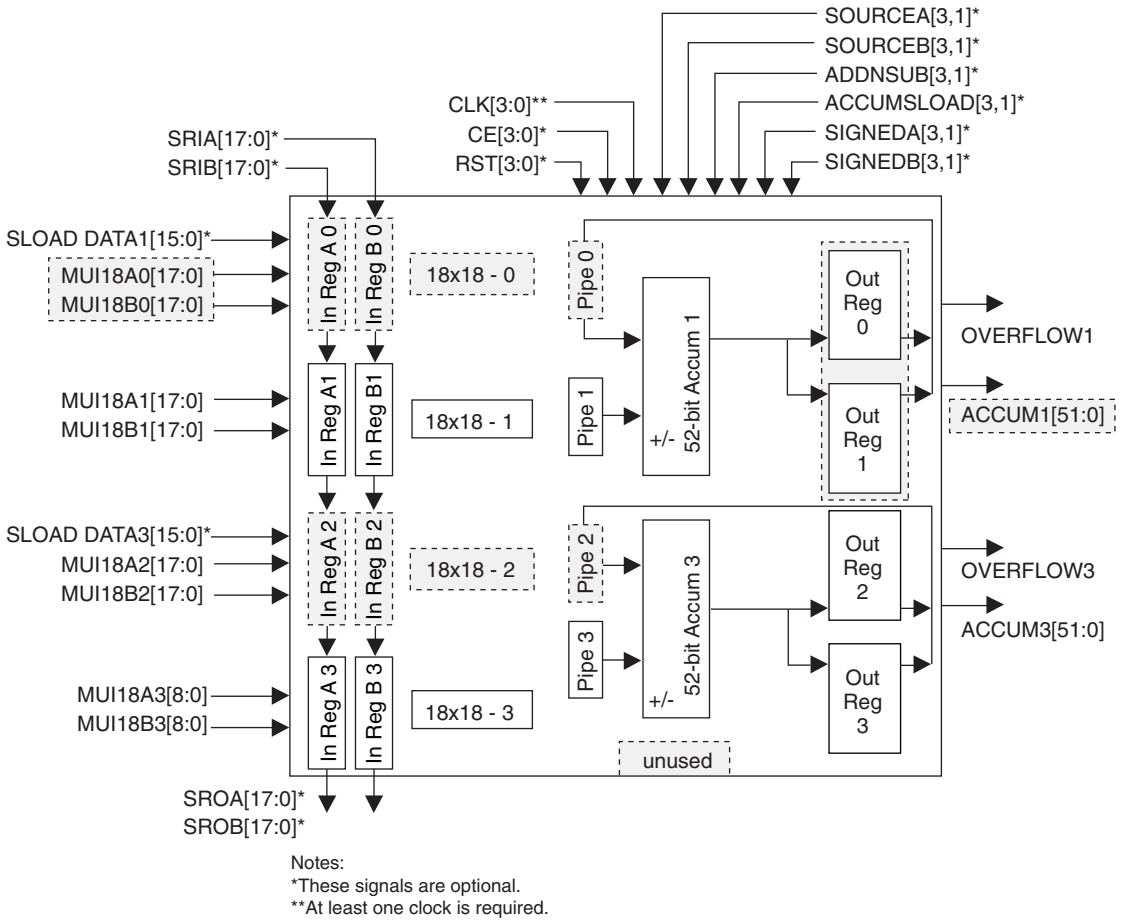
The above RTL will infer the following block diagram:

Figure 13-11. MULT18X18MACB Block Diagram



This block diagram can be mapped directly into the sysDSP primitives. Note that if a test point were added between the multiplier and the accumulator, or two output registers, etc. the code could not be mapped into a MULT18X18MACB of a sysDSP Block. Therefore, options that could be included in a design are input registers, pipeline registers, etc. For more Inferring design examples refer to EXAMPLES in the ispLEVER software.

Figure 13-12. MAC18X18MACB Packed into a sysDSP Block



Notes:
*These signals are optional.
**At least one clock is required.

sysDSP Blocks in the Report File

To check the configuration of the sysDSP Blocks in your design you can look at the MAP and Post PAR report files. The MAP report file shows the mapped sysDSP components/primitives in your design. The Post PAR report file shows the number of components in each sysDSP Block. The report files that follow show how the inferred MAC was used.

MAP Report File

```
. MULT18X18MACB  addout_17_0:

Multiplier
        Operation               Unsigned
        Operation Registers     CLK     CE      RST
        ------------------------------------------
                Input
                Pipeline
        Operation Registers     CLK     CE      RST
        ------------------------------------------
                Input
                Pipeline
```

```
AddSub
        Operation               Add
        Operation Registers     CLK     CE      RST
        ------------------------------------------
                Input
                Pipeline
Data
        Input Registers         CLK     CE      RST
        ------------------------------------------
                A               CLK0    CE0     RST0
                B               CLK0    CE0     RST0
        Pipeline Registers      CLK     CE      RST
        ------------------------------------------
                Pipe            CLK0    CE0     RST0
        Output Register         CLK     CE      RST
        ------------------------------------------
                Output          CLK0    CE0     RST0
Other
        GSR     ENABLED
   Number Of Mapped DSP Components:
   --------------------------------
   MULT36X36B          0
   MULT18X18B          0
   MULT18X18MACB       1
   MULT18X18ADDSUBB    0
   MULT18X18ADDSUBSUMB 0
   MULT9X9B            0
   MULT9X9ADDSUBB      0
   MULT9X9ADDSUBSUMB   0
   --------------------------------
```

Post PAR Report File

```
DSP Utilization Summary:
DSP Block #: 1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18
# of MULT36X36B
# of MULT18X18B
# of MULT18X18MACB                                        1
# of MULT18X18ADDSUBB
# of MULT18X18ADDSUBSUMB
# of MULT9X9B
# of MULT9X9ADDSUBB
# of MULT9X9ADDSUBSUMB
DSP Block  1    Component_Type         Instance_Name
DSP Block  2    Component_Type         Instance_Name
DSP Block  3    Component_Type         Instance_Name
DSP Block  4    Component_Type         Instance_Name
DSP Block  5    Component_Type         Instance_Name
DSP Block  6    Component_Type         Instance_Name
DSP Block  7    Component_Type         Instance_Name
DSP Block  8    Component_Type         Instance_Name
DSP Block  9    Component_Type         Instance_Name
  R45C81        MULT18X18MACB           addout_17_0
DSP Block 10    Component_Type         Instance_Name
DSP Block 11    Component_Type         Instance_Name
DSP Block 12    Component_Type         Instance_Name
DSP Block 13    Component_Type         Instance_Name
DSP Block 14    Component_Type         Instance_Name
DSP Block 15    Component_Type         Instance_Name
DSP Block 16    Component_Type         Instance_Name
DSP Block 17    Component_Type         Instance_Name
DSP Block 18    Component_Type         Instance_Name
```

Targeting the sysDSP Block Using Simulink

Simulink Overview

Simulink is a graphical add-on (similar to schematic entry) for Matlab®, which is produced by The MathWorks. For more information, refer to the Simulink web page at www.mathworks.com/products/simulink/.

Why is Simulink used?

- It allows users to create algorithms using floating point numbers.
- It helps users convert floating point algorithms into fixed point algorithms.

How does Simulink fit into the normal ispLEVER design flow?

- Once you have converted have your algorithm working in fixed point. You can use the Lattice ispDSP Block to create HDL files, which can be instantiated in your HDL design. Currently there is only support for VHDL.

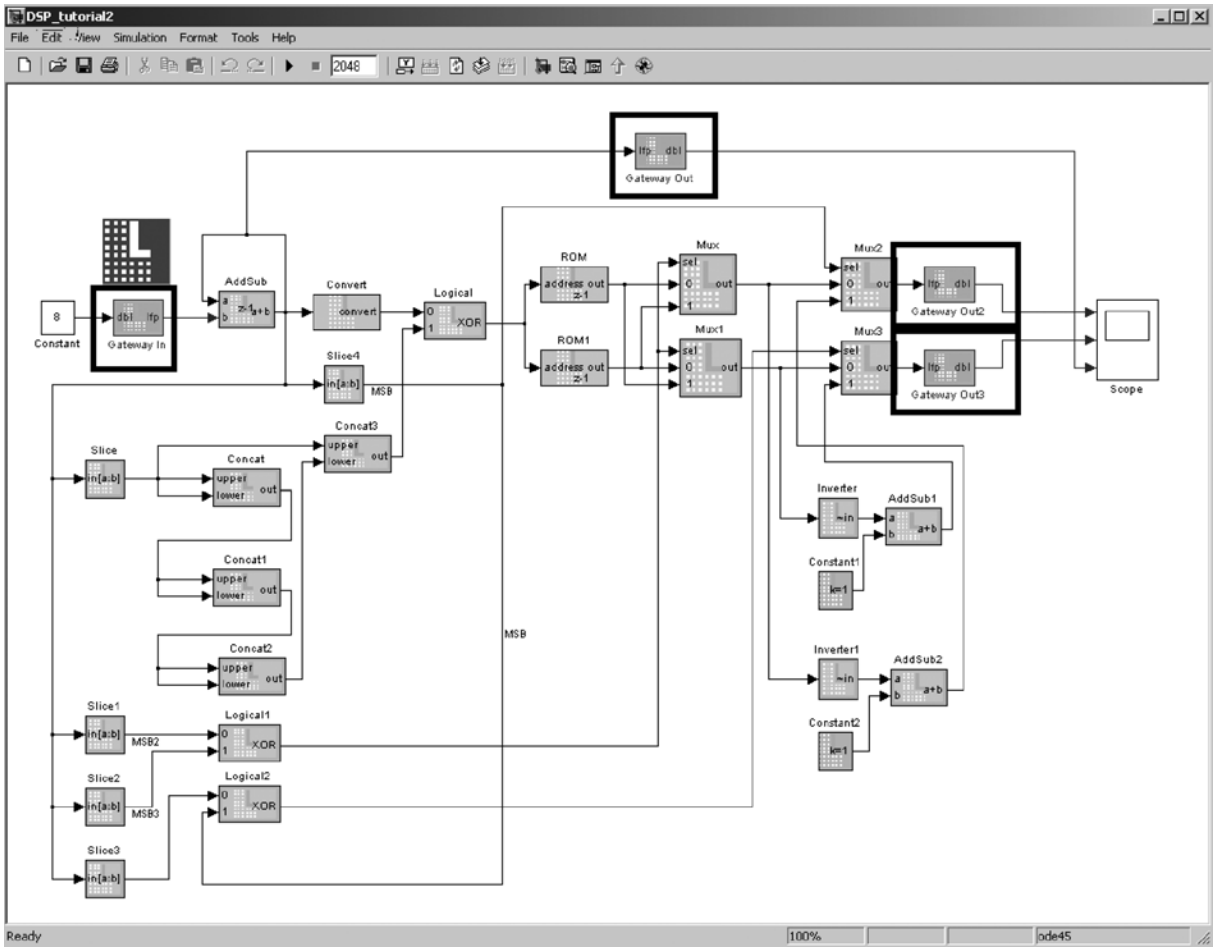
What does Lattice provide?

- Lattice provides a library of blocks for the Simulink tool, which include Multipliers, Adders, Registers, and other standard building blocks. Besides the basic building blocks there are a couple of unique Lattice blocks:

  Gateways In and Out
  Everything between Gateways In and Out represents the HDL code. Everything before a Gateway In is the stimulus your test bench. Everything after the Gateway Out are the signals you will be monitoring in the test bench. Below is an example. The box on the left contains Gateways In's and the three boxes on the right contain Gateway Outs in Figure 13-13.

  Generate
  The Generate block is used to convert Fixed point Simulink design into HDL files which can be instantiated in a HDL design. The Generate Block is identified by the Lattice logo and can be seen in Figure 13-13.

Figure 13-13. Simulink Design



Targeting the sysDSP Block by Instantiating Primitives

The sysDSP Block can be targeted by instantiating the sysDSP Block primitives into the design. The advantage of instantiating primitives is that it provides access to all ports and sets all available parameters. The disadvantage of this flow is that all this customization requires extra coding by the user. Appendix A details the syntax for the sysDSP Block primitives.

sysDSP Block Control Signal and Data Signal Descriptions

RST	Asynchronous reset of selected registers
SIGNEDA	Dynamic signal: 0 = unsigned, 1 = signed
SIGNEDB	Dynamic signal: 0 = unsigned, 1 = signed
ACCUMSLOAD	Dynamic signal: 0 = accumulate, 1 = load
ADDNSUB	Dynamic signal: 0 = subtract, 1 = add
SOURCEA	Dynamic signal: 0 = parallel input, 1 = shift input
SOURCEB	Dynamic signal: 0 = parallel input, 1 = shift input

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
February 2007	01.0	Initial release.

Appendix A. DSP Block Primitives

MULT18X18B

```
input A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0;
input SIGNEDA, SIGNEDB, SOURCEA, SOURCEB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input SRIA17,SRIA16,SRIA15,SRIA14,SRIA13,SRIA12,SRIA11,SRIA10,SRIA9;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB17,SRIB16,SRIB15,SRIB14,SRIB13,SRIB12,SRIB11,SRIB10,SRIB9;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA17,SROA16,SROA15,SROA14,SROA13,SROA12,SROA11,SROA10,SROA9;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB17,SROB16,SROB15,SROB14,SROB13,SROB12,SROB11,SROB10,SROB9;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output P35,P34,P33,P32,P31,P30,P29,P28,P27,P26,P25,P24,P23,P22,P21,P20,P19,P18;
output P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0;
parameter REG_INPUTA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB_CE = = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";
```

MULT18X18ADDSUBB

```
input A017,A016,A015,A014,A013,A012,A011,A010,A09;
input A08,A07,A06,A05,A04,A03,A02,A01,A00;
input A117,A116,A115,A114,A113,A112,A111,A110,A19;
input A18,A17,A16,A15,A14,A13,A12,A11,A10;
input B017,B016,B015,B014,B013,B012,B011,B010,B09;
input B08,B07,B06,B05,B04,B03,B02,B01,B00;
input B117,B116,B115,B114,B113,B112,B111,B110,B19;
input B18,B17,B16,B15,B14,B13,B12,B11,B10;
input SIGNEDA, SIGNEDB, SOURCEA0, SOURCEA1, SOURCEB0, SOURCEB1, ADDNSUB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input SRIA17,SRIA16,SRIA15,SRIA14,SRIA13,SRIA12,SRIA11,SRIA10,SRIA9;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB17,SRIB16,SRIB15,SRIB14,SRIB13,SRIB12,SRIB11,SRIB10,SRIB9;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA17,SROA16,SROA15,SROA14,SROA13,SROA12,SROA11,SROA10,SROA9;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB17,SROB16,SROB15,SROB14,SROB13,SROB12,SROB11,SROB10,SROB9;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output
SUM36,SUM35,SUM34,SUM33,SUM32,SUM31,SUM30,SUM29,SUM28,SUM27,SUM26,SUM25,SUM24,SUM23,SUM22,SUM21,SU
M20,SUM19,SUM18,SUM17,SUM16,SUM15,SUM14,SUM13,SUM12,SUM11,SUM10,SUM9,SUM8,SUM7,SUM6,SUM5,SUM4,SUM3
,SUM2,SUM1,SUM0;
```

```
parameter REG_INPUTA0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";
```

MULT18X18ADDSUBSUMB

```
input A017,A016,A015,A014,A013,A012,A011,A010,A09;
input A08,A07,A06,A05,A04,A03,A02,A01,A00;
input A117,A116,A115,A114,A113,A112,A111,A110,A19;
input A18,A17,A16,A15,A14,A13,A12,A11,A10;
input A217,A216,A215,A214,A213,A212,A211,A210,A29;
input A28,A27,A26,A25,A24,A23,A22,A21,A20;
input A317,A316,A315,A314,A313,A312,A311,A310,A39;
input A38,A37,A36,A35,A34,A33,A32,A31,A30;
input B017,B016,B015,B014,B013,B012,B011,B010,B09;
input B08,B07,B06,B05,B04,B03,B02,B01,B00;
input B117,B116,B115,B114,B113,B112,B111,B110,B19;
input B18,B17,B16,B15,B14,B13,B12,B11,B10;
input B217,B216,B215,B214,B213,B212,B211,B210,B29;
input B28,B27,B26,B25,B24,B23,B22,B21,B20;
input B317,B316,B315,B314,B313,B312,B311,B310,B39;
input B38,B37,B36,B35,B34,B33,B32,B31,B30;
input SIGNEDA, SIGNEDB,ADDNSUB1,ADDNSUB3;
input SOURCEA0, SOURCEA1, SOURCEA2, SOURCEA3;
input SOURCEB0, SOURCEB1, SOURCEB2, SOURCEB3;
```

```
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input SRIA17,SRIA16,SRIA15,SRIA14,SRIA13,SRIA12,SRIA11,SRIA10,SRIA9;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB17,SRIB16,SRIB15,SRIB14,SRIB13,SRIB12,SRIB11,SRIB10,SRIB9;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA17,SROA16,SROA15,SROA14,SROA13,SROA12,SROA11,SROA10,SROA9;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB17,SROB16,SROB15,SROB14,SROB13,SROB12,SROB11,SROB10,SROB9;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output
SUM37,SUM36,SUM35,SUM34,SUM33,SUM32,SUM31,SUM30,SUM29,SUM28,SUM27,SUM26,SUM25,SUM24,SUM23,SUM22,SU
M21,SUM20,SUM19,SUM18,SUM17,SUM16,SUM15,SUM14,SUM13,SUM12,SUM11,SUM10,SUM9,SUM8,SUM7,SUM6,SUM5,SUM
4,SUM3,SUM2,SUM1,SUM0;
parameter REG_INPUTA0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
```

```
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB1_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB1_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB1_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB1_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB1_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB1_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB3_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB3_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB3_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB3_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB3_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB3_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";
```

MULT18X18MACB

```
input A17,A16,A15,A14,A13,A12,A11,A10,A9;
input A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B17,B16,B15,B14,B13,B12,B11,B10,B9;
input B8,B7,B6,B5,B4,B3,B2,B1,B0;
input ADDNSUB, SIGNEDA, SIGNEDB,ACCUMSLOAD;
input SOURCEA, SOURCEB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input LD51, LD50, LD49, LD48, LD47, LD46, LD45, LD44, LD43, LD42, LD41, LD40
input LD39, LD38, LD37, LD36, LD35, LD34, LD33, LD32, LD31, LD30
input LD29, LD28, LD27, LD26, LD25, LD24, LD23, LD22, LD21, LD20
input LD19, LD18, LD17, LD16, LD15, LD14, LD13, LD12, LD11, LD10
input LD9, LD8, LD7, LD6, LD5, LD4, LD3, LD2, LD1, LD0;
input SRIA17,SRIA16,SRIA15,SRIA14,SRIA13,SRIA12,SRIA11,SRIA10,SRIA9;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB17,SRIB16,SRIB15,SRIB14,SRIB13,SRIB12,SRIB11,SRIB10,SRIB9;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA17,SROA16,SROA15,SROA14,SROA13,SROA12,SROA11,SROA10,SROA9;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB17,SROB16,SROB15,SROB14,SROB13,SROB12,SROB11,SROB10,SROB9;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output
ACCUM51,ACCUM50,ACCUM49,ACCUM48,ACCUM47,ACCUM46,ACCUM45,ACCUM44,ACCUM43,ACCUM42,ACCUM41,ACCUM40,AC
CUM39,ACCUM38,ACCUM37,ACCUM36,ACCUM35,ACCUM34,ACCUM33,ACCUM32,ACCUM31,ACCUM30,ACCUM29,ACCUM28,ACCU
M27,ACCUM26,ACCUM25,ACCUM24,ACCUM23,ACCUM22,ACCUM21,ACCUM20,ACCUM19,ACCUM18,ACCUM17,ACCUM16,ACCUM1
5,ACCUM14,ACCUM13,ACCUM12,ACCUM11,ACCUM10,ACCUM9,ACCUM8,ACCUM7,ACCUM6,ACCUM5,ACCUM4,ACCUM3,ACCUM2,
ACCUM1,ACCUM0,OVERFLOW;
parameter REG_INPUTA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
```

```
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ACCUMSLOAD_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ACCUMSLOAD_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ACCUMSLOAD_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ACCUMSLOAD_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ACCUMSLOAD_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ACCUMSLOAD_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";
```

MULT36X36B

```
input A35,A34,A33,A32,A31,A30,A29,A28,A27,A26,A25,A24,A23,A22,A21,A20,A19,A18;
input A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B35,B34,B33,B32,B31,B30,B29,B28,B27,B26,B25,B24,B23,B22,B21,B20,B19,B18;
input B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0;
input SIGNEDA, SIGNEDB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
output P71,P70,P69,P68,P67,P66,P65,P64,P63,P62,P61,P60,P59,P58,P57,P56,P55,P54;
output P53,P52,P51,P50,P49,P48,P47,P46,P45,P44,P43,P42,P41,P40,P39,P38,P37,P36;
output P35,P34,P33,P32,P31,P30,P29,P28,P27,P26,P25,P24,P23,P22,P21,P20,P19,P18;
output P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0;
parameter REG_INPUTA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";
```

MULT9X9B

```
input A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B8,B7,B6,B5,B4,B3,B2,B1,B0;
input SIGNEDA, SIGNEDB, SOURCEA, SOURCEB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0;
parameter REG_INPUTA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";
```

MULT9X9ADDSUBB

```
input A08,A07,A06,A05,A04,A03,A02,A01,A00;
input A18,A17,A16,A15,A14,A13,A12,A11,A10;
input B08,B07,B06,B05,B04,B03,B02,B01,B00;
input B18,B17,B16,B15,B14,B13,B12,B11,B10;
input SIGNEDA, SIGNEDB,ADDNSUB;
input SOURCEA0, SOURCEA1, SOURCEB0, SOURCEB1;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output
SUM18,SUM17,SUM16,SUM15,SUM14,SUM13,SUM12,SUM11,SUM10,SUM9,SUM8,SUM7,SUM6,SUM5,SUM4,SUM3,SUM2,SUM1
,SUM0;
parameter REG_INPUTA0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE0_RST = " RST0, RST1, RST2, RST3 ";
```

```
parameter REG_PIPELINE1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";
```

MULT9X9ADDSUBSUMB

```
input A08,A07,A06,A05,A04,A03,A02,A01,A00;
input A18,A17,A16,A15,A14,A13,A12,A11,A10;
input A28,A27,A26,A25,A24,A23,A22,A21,A20;
input A38,A37,A36,A35,A34,A33,A32,A31,A30;
input B08,B07,B06,B05,B04,B03,B02,B01,B00;
input B18,B17,B16,B15,B14,B13,B12,B11,B10;
input B28,B27,B26,B25,B24,B23,B22,B21,B20;
input B38,B37,B36,B35,B34,B33,B32,B31,B30;
input SIGNEDA, SIGNEDB,ADDNSUB1,ADDNSUB3;
input SOURCEA0, SOURCEA1, SOURCEA2, SOURCEA3;
input SOURCEB0, SOURCEB1, SOURCEB2, SOURCEB3;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output
SUM19,SUM18,SUM17,SUM16,SUM15,SUM14,SUM13,SUM12,SUM11,SUM10,SUM9,SUM8,SUM7,SUM6,SUM5,SUM4,SUM3,SUM
2,SUM1,SUM0;
parameter REG_INPUTA0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB0_RST = " RST0, RST1, RST2, RST3 ";
```

```
parameter REG_INPUTB1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB1_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB1_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB1_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB1_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB1_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB1_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB3_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB3_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB3_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB3_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB3_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB3_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";
```