



LatticeECP2/M sysDSP 使用ガイド

はじめに

このテクニカルノートはLatticeECP2/M sysDSP(デジタル信号処理)ブロックの機能にアクセスする方法について議論します。sysDSPブロックを対象とする設計は、伝統的なLUTベースの実装に対してかなりの改善を与えます。表12-1は、このアプローチの性能と使用エリアの恩恵についての例を示します。

表12-1 sysDSPブロック対LUTベースの乗算器

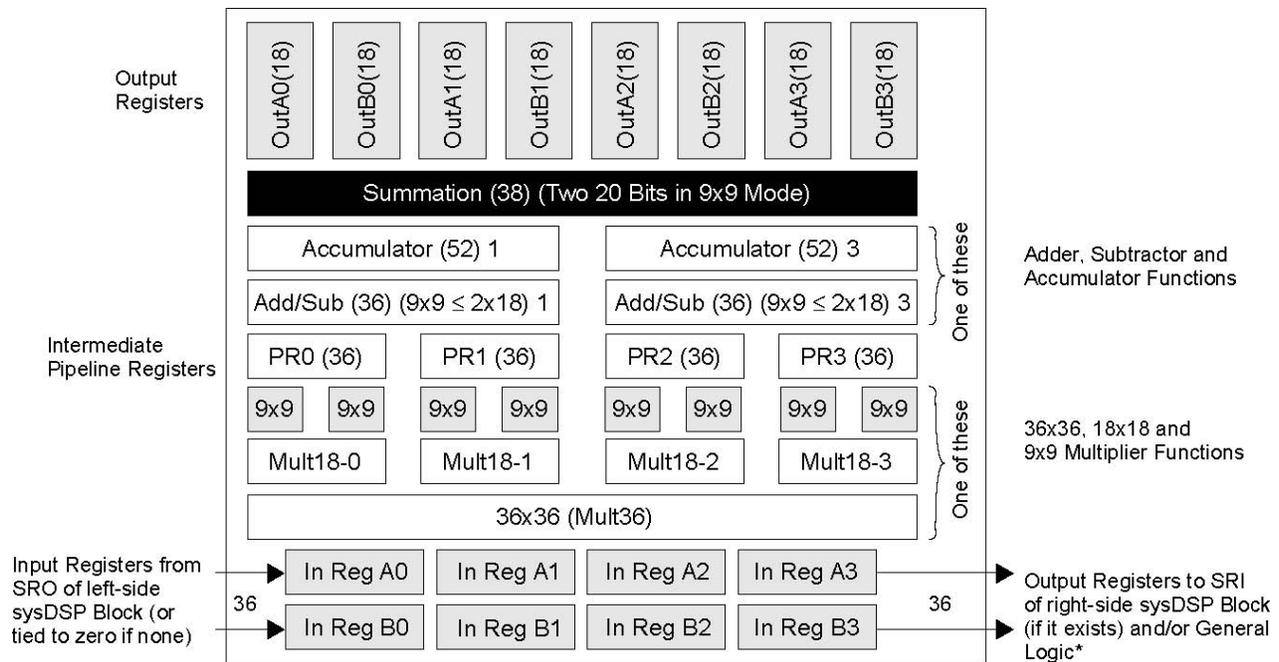
乗算器ビット幅	パイプライン・レジスタ	ECP2-50-7 DSPブロック1個使用		ECP2-50-7 LUT使用	
		f _{MAX}	LUT数	f _{MAX}	LUT数
9x9	入力、乗算器、出力	404	0	124	192
18x18	入力、乗算器、出力	420	0	95	698
36x36	入力、乗算器、出力	371	0	61	2732

sysDSPブロックのハードウェア

概要

sysDSPブロックはLatticeECP2/Mデバイスの中央の列に位置しています。sysDSPのブロック図は図12-1で示されます。

図12-1 LatticeECP2/M sysDSPブロック図



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

Note: Each sysDSP Block spans nine columns of PFUs.

sysDSPブロックは、次のように構成できます。

36x36 乗算器1個

- 基本的な乗算器。加減算・アキュミュレーション・総和(add/sub/accum/sum)ブロックはなし

18x18 乗算器4個

- 加減算・アキュミュレーション・ブロックが2個
- 乗算器4個の総和(sum)ブロックが1個

9x9 乗算器8個

- 加減算ブロックが4個
- 総和ブロックが2個

sysDSPブロックは一度にいずれか一つのモードにのみ構成できます。

sysDSPブロック・ソフトウェア

概要

幾つかの方法でLatticeECP2/MデバイスのsysDSPブロックをターゲットとすることができます。

- ラティスispLEVERデザインツールのIPexpressで、sysDSP要素を実装するモジュールを迅速に生成できます。そして、HDLデザインでこれらのモジュールを適宜インスタンス化して用いることができます。
- HDL設計で特定機能をコード記述することで、論理合成ツールにsysDSPブロックの使用を推論させることができます。
- Mathworks社のSimulinkツールでラティス・ブロックセットを用いて設計を実装します。ispLEVERツールのispLEVER sysDSP部が、適切にこれらのブロックをHDLに変換します。
- 直接ソースコードのsysDSPプリミティブでインスタンス化します。

IPexpressを用いたsysDSPブロックのターゲット

IPexpressでグラフィカルにsysDSP要素を指定できます。要素がいったん指定されると、HDLファイルが生成され、設計内でインスタンスして使用することができます。ユーザは、IPexpressで全てのポートを構成し、全ての利用できるパラメータを設定できます。以下はsysDSPブロックをターゲットとするモジュールを示します。

IPexpressを用いる設計例については、ispLEVERソフトウェアの例 (EXAMPLES) を参照してください。プロジェクト・ナビゲータ(Project Navigator)プルダウンメニューで、**File->Open Example**を選択してください。IPexpressでは以下の4つの要素タイプを指定することができます。

- MULT (乗算器)
- MAC (積和；乗算と累積)
- MULTADDSUB (乗算と、加算/減算)
- MULTADDSUBSUM (乗算、加算/減算、そして総和)

MULTモジュール

MULTモジュールは、sysDSPプリミティブに詰め込まれる要素を構成します。図12-2に図示されたベーシックモード・スクリーンは、クロック(オプション)、クロックイネーブル、リセットそれぞれ一本よりなり、全てのレジスタに接続されています。多ビットの乗算に対応するために複数のsysDSPブロックにまたがるすることができます。複数のsysDSPブロックが必要な場合、付加的なLUTが必要かもしれません。Area/Speedの選択がLUT実装を決定します。入力データ形式はパラレル、シフト、またはダイナミックが選定できます。シフト形式は入力が18ビット未満である場合にだけイネーブルできます。シフト形式はサンプル、或いはシフトレジスタで、FIRフィルタなどのアプリケーションで役立ちます。図12-3で示されるアドバンストモー

ド・スクリーンは、レジスタのより詳細な制御を可能にします。アドバンストモードで、ユーザは各レジスタを独立したクロック、クロックイネーブル、およびリセットで制御することができます。MULT入力は2〜72ビットをサポートします。

図12-2 MULTモードのベーシック・セットアップ

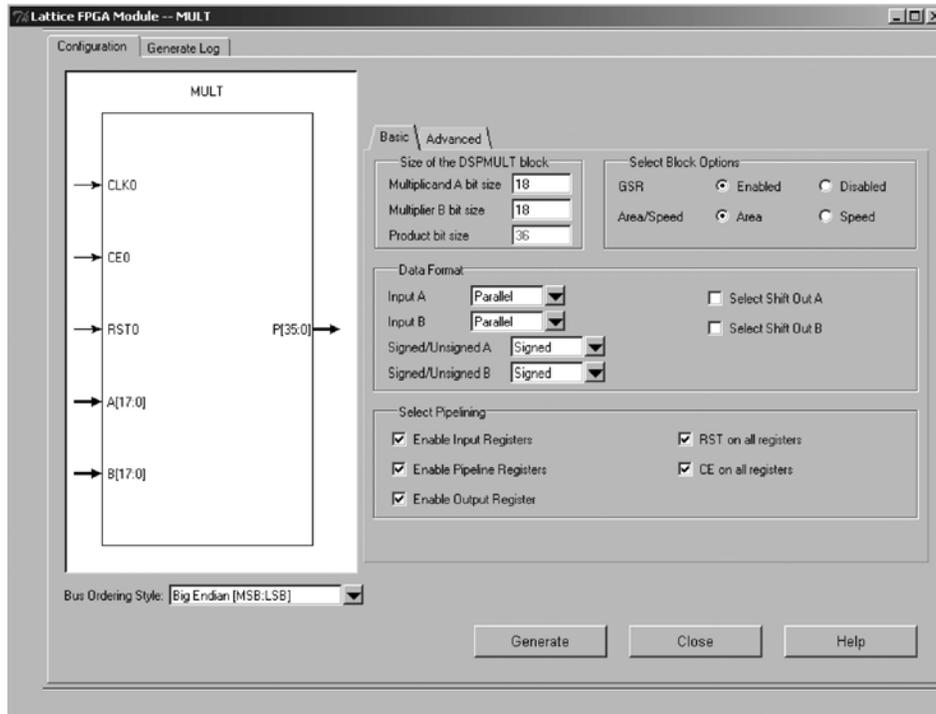
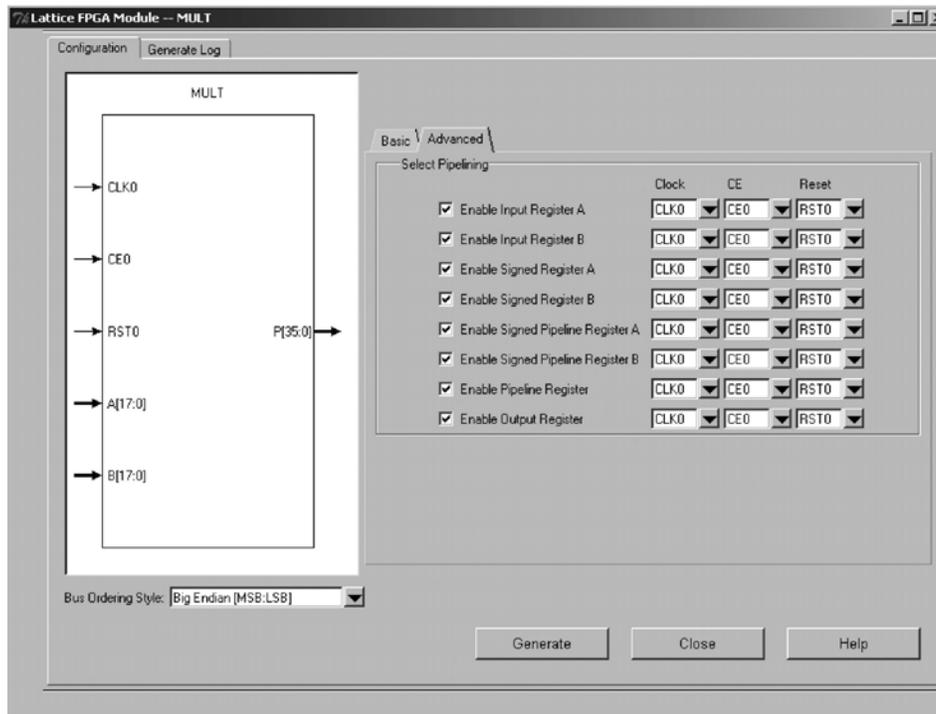


図12-3 MULTモードのアドバンスト・セットアップ



MAC モジュール

MACモジュールは、プリミティブにパッキングされる積和要素を構成します。図12-4で示されたベーシックモードは、クロック(オプション)、クロックイネーブル、およびリセットそれぞれ1本からなり、全てのレジスタに接続されています。アキュムレータがあるため、出力レジスタは自動的にイネーブルされます。多ビットの乗算に対応するために複数のsysDSPブロックにまたがることができます。sysDSPブロックのアキュムレータは深さ52ビットですが、より多ビットの累積(アキュムレート)が必要な場合、追加LUTを用いて実現することができます。複数のsysDSPブロックにまたがる場合、追加LUTロジックが必要です。Area/Speedを選択してLUT実装を決定します。入力データ形式は平行、シフト、またはダイナミックが選定できます。シフト形式は入力が18ビット未満である場合にのみイネーブルできます。シフト形式はサンプル/シフトレジスタをイネーブルします。AccumsloadはアキュムレータをLDポートからの値と共にロードします。これはアキュムレートの最初の値を初期化してロードするために必要です。図12-5で示すアドバンスドモードはレジスタのより詳細の制御を可能にします。アドバンスドモードで、各レジスタをユーザは独立したクロック、クロックイネーブル、およびリセットでそれぞれ制御することができます。MAC入力は2~72ビットをサポートします。

図12-4 MACモードのベーシック・セットアップ

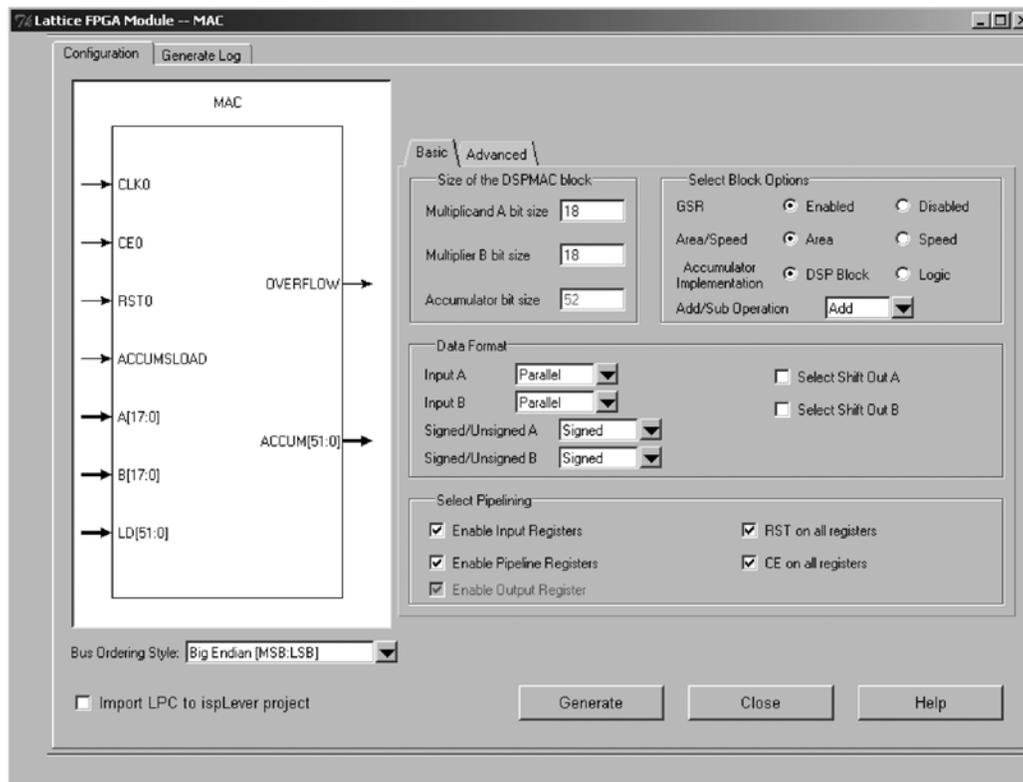
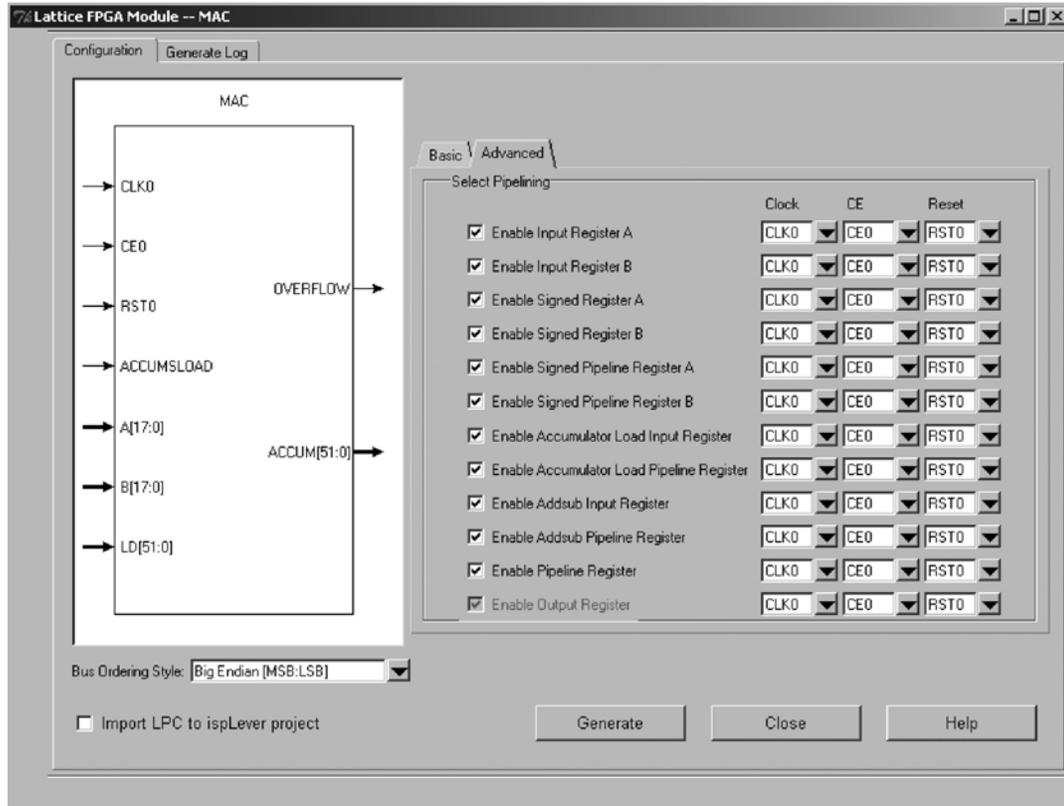


図12-5 MACモードのアドバンスト・セットアップ



MULTADDSUB モジュール

MULTADDSUB GUIは、プリミティブMULT18X18ADDSUB、またはMULT9X9ADDSUBにパッキングされる乗算・加減算要素を構成します。図12-6に示すベーシックモードは、クロック(オプション)、クロックイネーブル、およびリセットそれぞれ1本からなり、全てのレジスタに接続されています。多ビットの乗算に対応するために複数のsysDSPブロックにまたがることができます。複数のsysDSPブロックにまたがる場合、追加LUTロジックが必要です。Area/Speedを選択してLUT実装を決定します。入力データ形式は平行、シフト、またはダイナミックが選定できます。シフト形式は入力が18ビット未満である場合にのみイネーブルできます。シフト形式はサンプル/シフトレジスタをイネーブルし、FIRフィルタなどのアプリケーションで役に立ちます。図12-7で示すアドバンストモードは、レジスタのより詳細な制御を可能にします。アドバンストモードで、各レジスタをユーザは独立したクロック、クロックイネーブル、およびリセットでそれぞれ制御することができます。MULTADDSUB入力は2~72ビットにできます。

図12-6 MULTADDSUBモードのベーシック・セットアップ

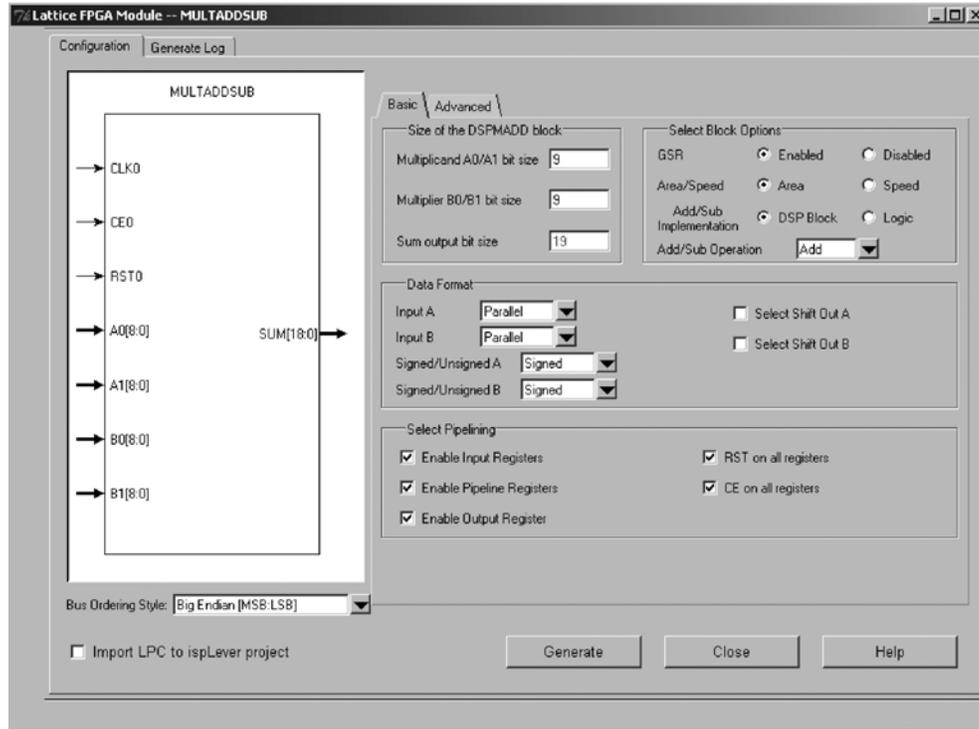
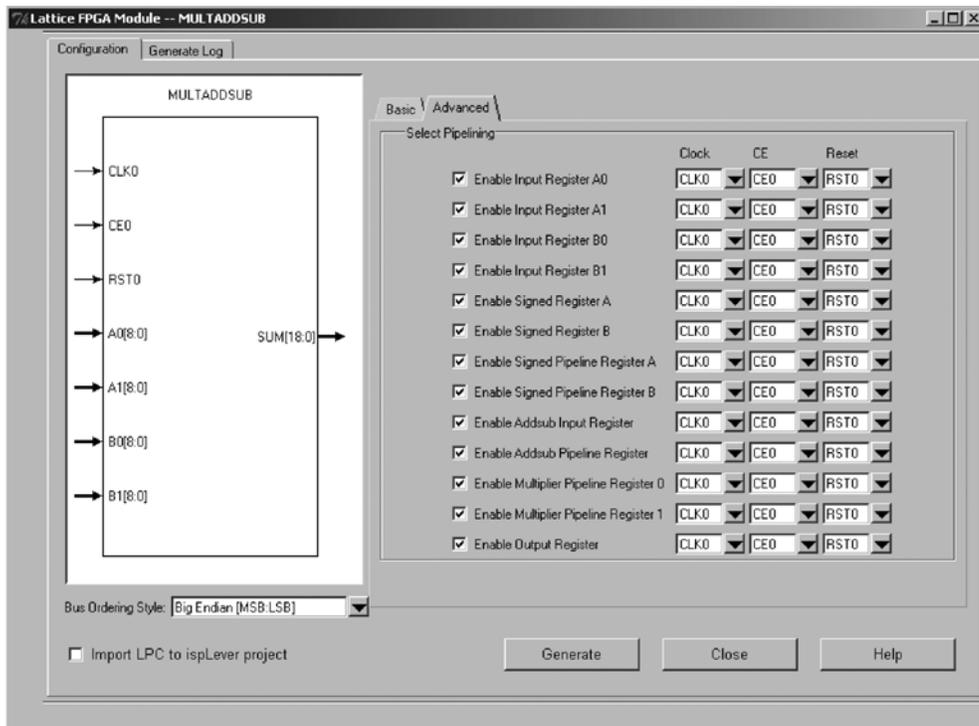


図12-7 MULTADDSUBモードのアドバンスド・セットアップ



MULTADDSUBSUM モジュール

MULTADDSUBSUM GUIは、プリミティブMULT18X18ADDSUBSUM、またはMULT9X9ADDSUBSUMにパッキングされる乗算・加減算・総和要素を構成します。図12-8に示すベーシックモードは、クロック(オプション)、クロックイネーブル、およびリセットそれぞれ1本からなり、全てのレジスタに接続されています。多ビットの乗算に対応するために複数のsysDSPブロックにまたがることができます。複数のsysDSPブロックにまたがる場合、追加LUTロジックが必要です。Area/Speedを選択してLUT実装を決定します。入力データ形式は平行、シフト、またはダイナミックが選定できます。シフト形式は入力が18ビット未満である場合にのみイネーブルできます。シフト形式はサンプル/シフトレジスタをイネーブルし、FIRフィルタなどのアプリケーションで役に立ちます。図12-9で示すアドバンスド・モードは、レジスタのより詳細な制御を可能にします。アドバンスド・モードで、各レジスタをユーザは独立したクロック、クロックイネーブル、およびリセットでそれぞれ制御することができます。MULTADDSUBSUM入力は2~72ビットにできます。

図12-8 MULTADDSUBSUMモードのベーシック・セットアップ

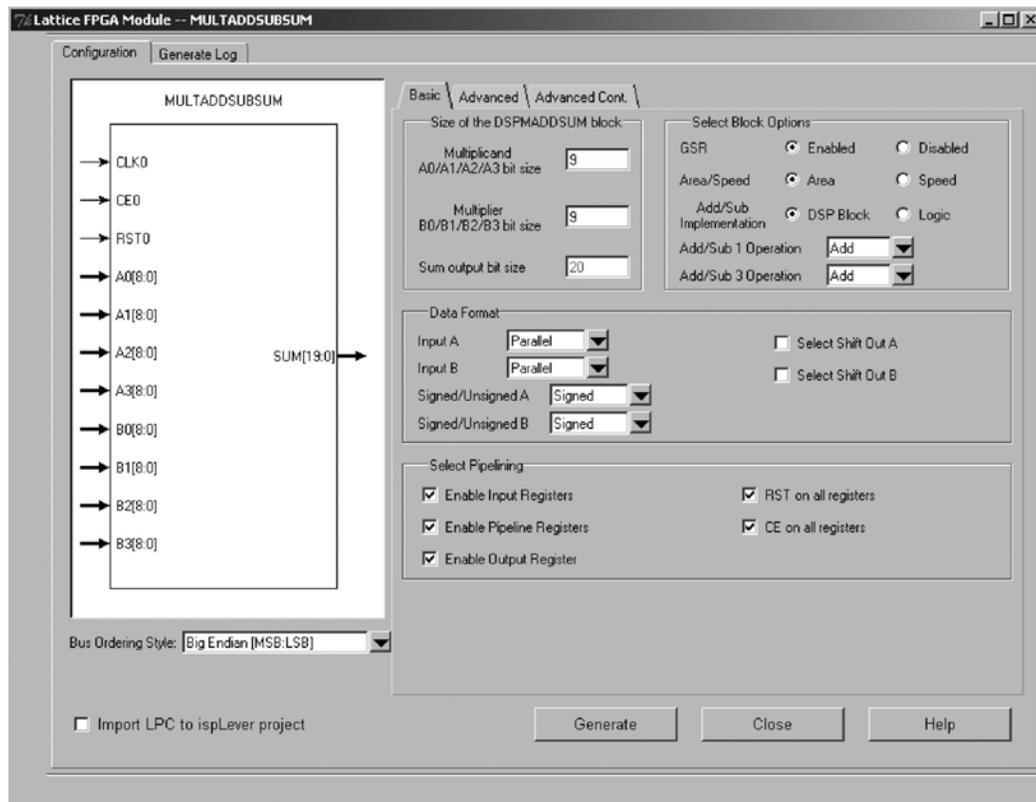
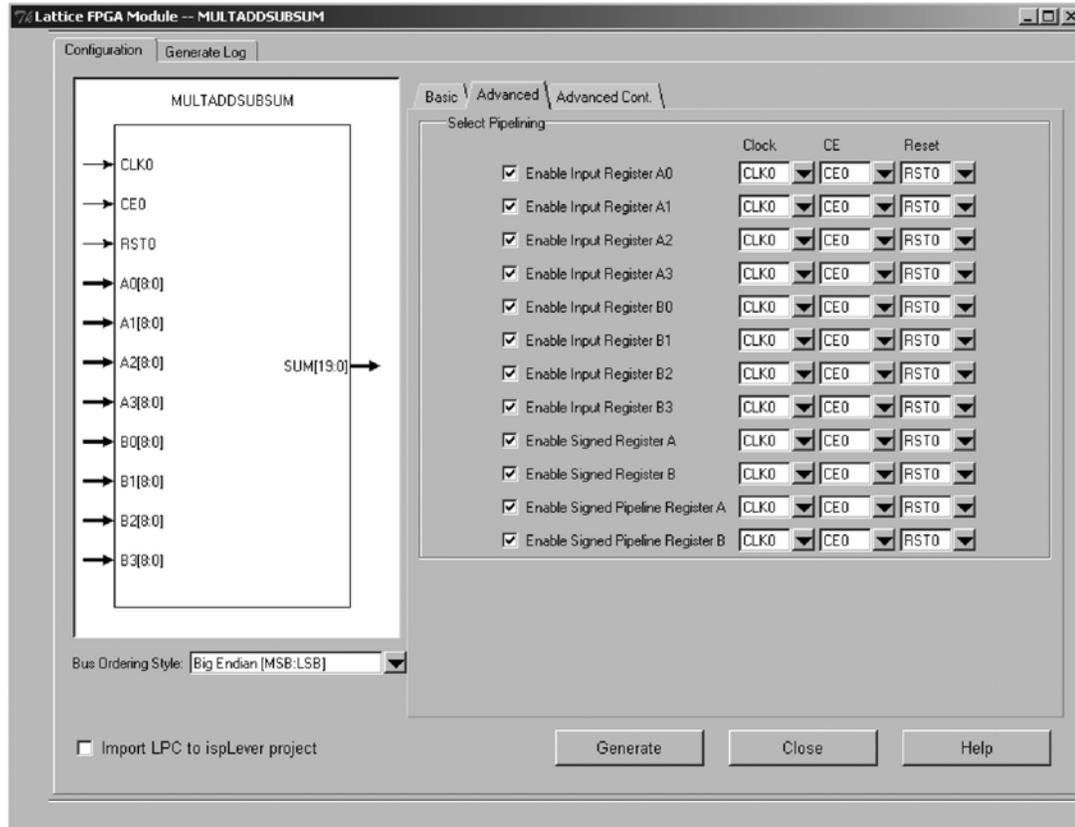


図12-9 MULTADDSUMモードのアドバンスド・セットアップ



推論によるsysDSPブロックのターゲット

推論フローは、デザインツールにHDL設計からsysDSPブロックを推論させることができます。推論フローを用いる場合、コーディング・スタイルが合致しないとsysDSPブロック推論結果が最適にならないのに注意することは重要です。以下のVerilogとVHDLの例を考えます。

```
// This Verilog example will be mapped into single MULT9X9MAC with the output register
// enabled
```

```
// This will be mapped into single MULT9X9MAC 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] mult_a = dataax * dataay; // 9x9 Multiplier
    wire [16:0] adder_out;
    assign adder_out = mult_a + 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;
```

```

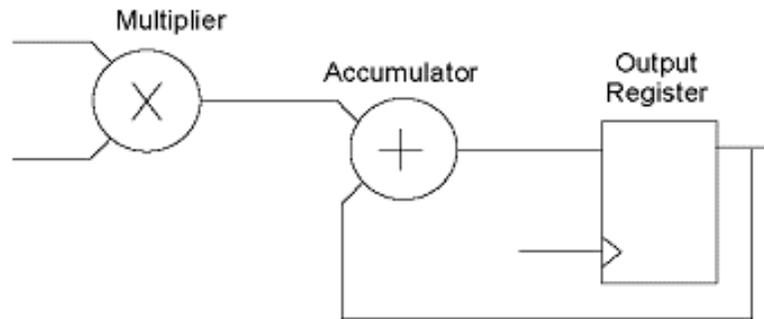
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

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;

```

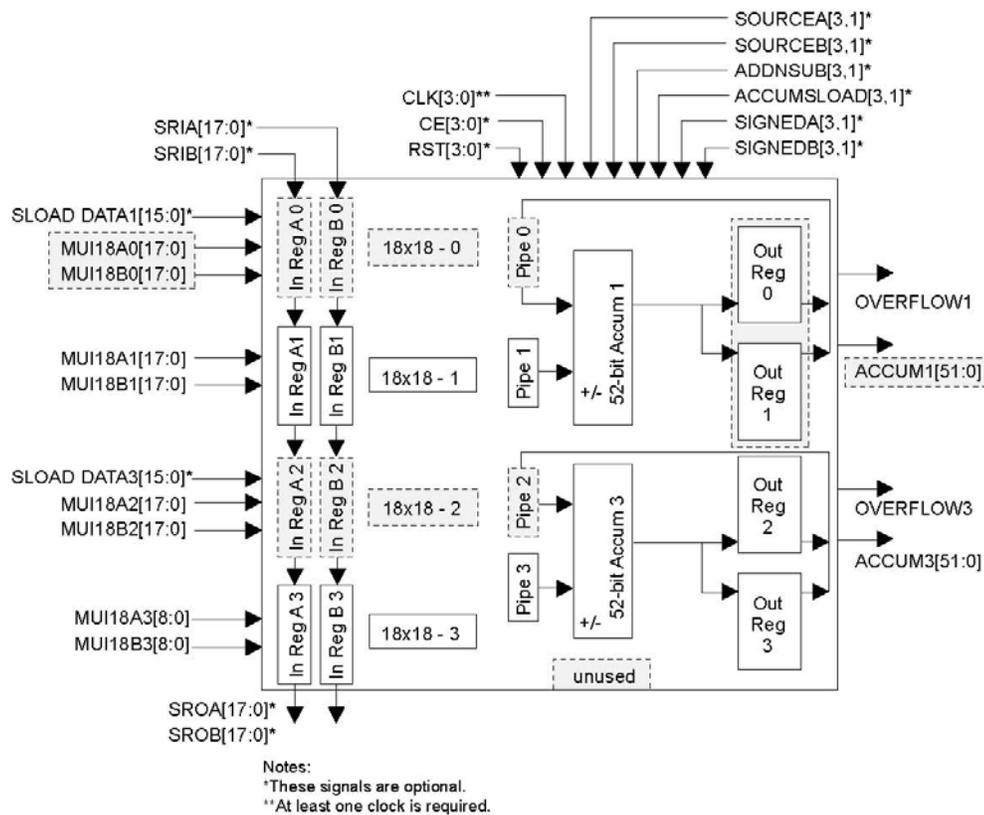
上のRTLは次に示すブロック図を推論するでしょう。

図12-11 MULT18X18MACBブロック図



このように、直接sysDSPプリミティブにこのブロック図を配置することができます。仮に乗算器とアキュムレータの間にテストポイントを加えたり、出力レジスタを2つにすると、そうしたコードではsysDSPブロックのMULT18X18MACBに配置することができないでしょう。設計中に含むことができるオプションは、入力レジスタ、パイプラインレジスタなどです。これ以上の推論設計については、ispLEVERソフトウェアの例 (EXAMPLE) を参照して下さい。

図12-12 sysDSPブロックにパッキングされたMAC18X18MACB



レポートファイルにおけるsysDSPブロック

デザインにおけるsysDSPブロックのコンフィグレーションは、MAPレポートとポストPARレポート両ファイルを見ることでチェックできます。MAPレポートファイルではマッピングされたsysDSPコンポーネント/プリミティブを示します。ポストPARレポートファイルは各sysDSPブロック内のコンポーネント数を示します。以下のレポートファイルは推論されたMACがどのように用いられたかを示します

MAPレポートファイル

. 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 (配置配線後の) レポートファイル

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
# 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

Simulinkを用いたsysDSPブロックのターゲット

Simulink概要

SimulinkはMatlabのためのグラフィカルなアドオン(回路図入力と類似)で、Mathworks社によって提供されます。詳しい情報については、次のウェブサイトを参照してください。
<http://www.mathworks.com/products/simulink/>

なぜSimulinkを用いるか?

- Simulinkは、ユーザに浮動小数点を用いたアルゴリズム作成を可能にします。
- ユーザが浮動小数点アルゴリズムを固定小数点アルゴリズムに変換する手助となります。

通常のispLEVER設計フローにSimulinkはどうフィットするか?

いったん固定小数点で動作するアルゴリズムに変換できてしまうと、設計内でインスタンスするHDLファイルを作成するためにLattice ispDSPブロックを用いることができます。現在はVHDLのみサポートしています。

ラティスの提供するものは?

ラティスはSimulinkツールのためのブロック・ライブラリを提供し、これには乗算器、加算器、レジスタ、および他の標準のビルディング・ブロックが含まれます。この基本的なビルディング・ブロックに加えて、2~3のユニークなラティス・ブロックがあります。

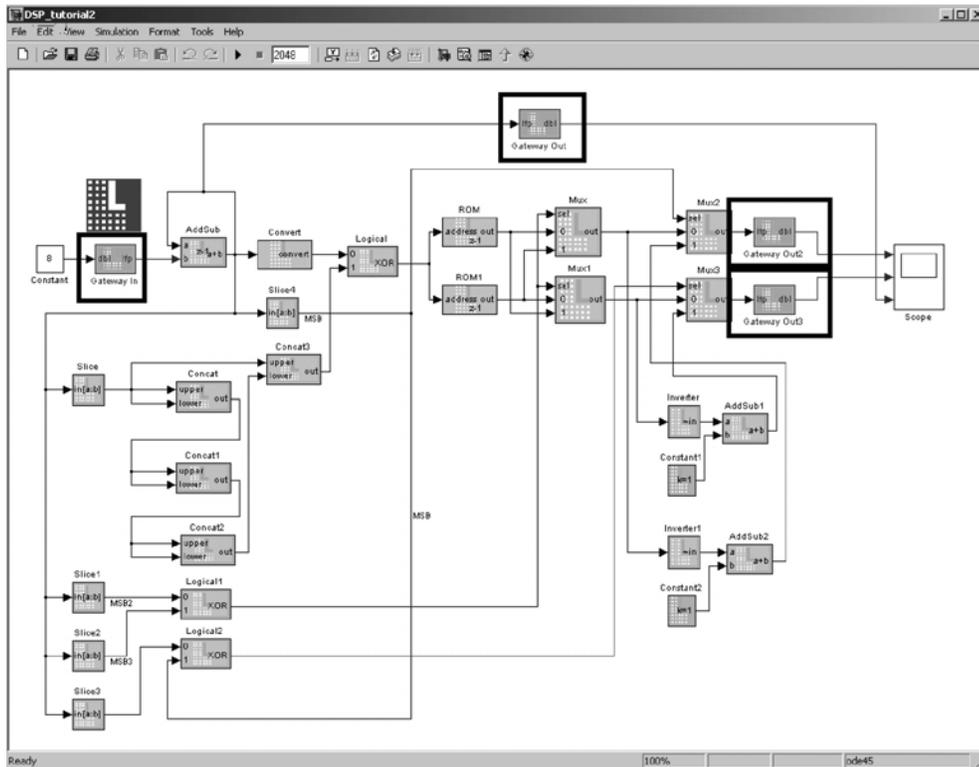
入出力ゲートウェイ

入力ゲートウェイ (Gateway In) と出力ゲートウェイ (Gateway Out)の間のすべてがHDLコードになります。入力ゲートウェイ前のすべては設計者によるテストベンチのステイミュラスです。出力ゲートウェイ (Gateway Out) の後すべては、設計者がテストベンチでモニターする信号です。図12-13では、左側の枠の中に入力ゲートウェイがあり、右側にある3つの枠に出力ゲートウェイがあります。

Generator

Generatorブロックは、固定小数点のSimulink設計を、HDL設計内にインスタンスすることができるHDLファイルに変換するのに用いられます。Generatorブロックはラティス・ロゴで特定され、図12-13の左上にあります。

図12-13 Simulinkによる設計



プリミティブをインスタンスすることによるsysDSPブロックのターゲット

sysDSPブロック・プリミティブを設計内でインスタンスすることで、sysDSPブロックをターゲットにすることが出来ます。プリミティブをインスタンスする利点は、全ポートへのアクセスが可能になり、利用できる全パラメータ・セットが設定できるということです。このフローの不都合は、このすべてのカスタム化が、ユーザにとっては余分なコーディングのコストになるということです。付録AはsysDSPブロック・プリミティブのための構文を示します（日本語版では省略。英語版を参照）。

SysDSPブロックの制御信号とデータ信号の記述

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

テクニカル・サポート支援

ホットライン: 1-800-LATTICE (North America)
 +1-503-268-8001 (Outside North America)
 e-mail: techsupport@latticesemi.com
 インターネット: www.latticesemi.com