Arora V DSP
ユーザーガイド

UG305-1.0J, 2023-04-20

免責事項

当社は、GOWINSEMI Terms and Conditions of Sale(GOWINSEMI取引条件)に規定されている内容を除き、(明示的か又は黙示的かに拘わらず)いかなる保証もせず、また、知的財産権や材料の使用によりあなたのハードウェア、ソフトウェア、データ、又は財産が被った損害についても責任を負いません。当社は、事前の通知なく、いつでも本文書の内容を変更することができます。本文書を参照する何れの団体及び個人も、最新の文書やエラッタ(不具合情報)については、当社に問い合わせる必要があります。

バージョン履歴

日付	バージョン	説明
2023/04/20	1.0J	初版。

目次

図一覧

表一覧

1 本マニュアルについて

1.1 マニュアル内容

本マニュアルは、主に Arora V FPGA 製品の DSP リソースの構造、信号の定義、及び呼び出し方法について説明し、ユーザーの Gowin DSP の最大限の活用と設計効率の向上を目的としています。

1.2 関連ドキュメント

GOWIN セミコンダクターの公式 Web サイト www.gowinsemi.com/ja から、以下の関連ドキュメントがダウンロード、参考できます：

- GW5AT シリーズ FPGA 製品データシート(DS981)
- GW5A シリーズ FPGA 製品データシート(DS1103)
- GW5AST シリーズ FPGA 製品データシート(DS1104)
- Gowin ソフトウェア ユーザーガイド(SUG100)

1.3 用語、略語

表 1-1 に、本マニュアルで使用される用語、略語、及びその意味を示します。

表 1-1 用語、略語

用語、略語	正式名称	意味
CFU	Configurable Function Unit	コンフィギャラブル機能ユニット
DSP	Digital Signal Processing	デジタル信号処理
FIR	Finite Impulse Response	有限インパルス応答フィルター
FFT	Fast Fourier Transformation	高速フーリエ変換
MULT	Multiplier	乗算器
PADD	Pre-adder	前置加算器
48-bit ALU	48-bit Arithmetic Logic Unit	48 ビットの算術論理演算装置

1.4 テクニカル・サポートとフィードバック

GOWIN セミコンダクターは、包括的な技術サポートをご提供しています。使用に関するご質問、ご意見については、直接弊社までお問い合わせください。

Web サイト：www.gowinsemi.com/ja

E-mail：support@gowinsemi.com

2 概要

Gowin Arora V FPGA 製品には、FIR、FFT 設計などの高性能デジタル信号処理を可能にする豊富な DSP リソースがあります。DSP ブロックは、安定したタイミングパフォーマンス、高いリソース使用率、低消費電力等の特性を備えています。このマニュアルは、ユーザーが Arora V FPGA 製品の DSP リソースを使いこなせるよう作成されています。

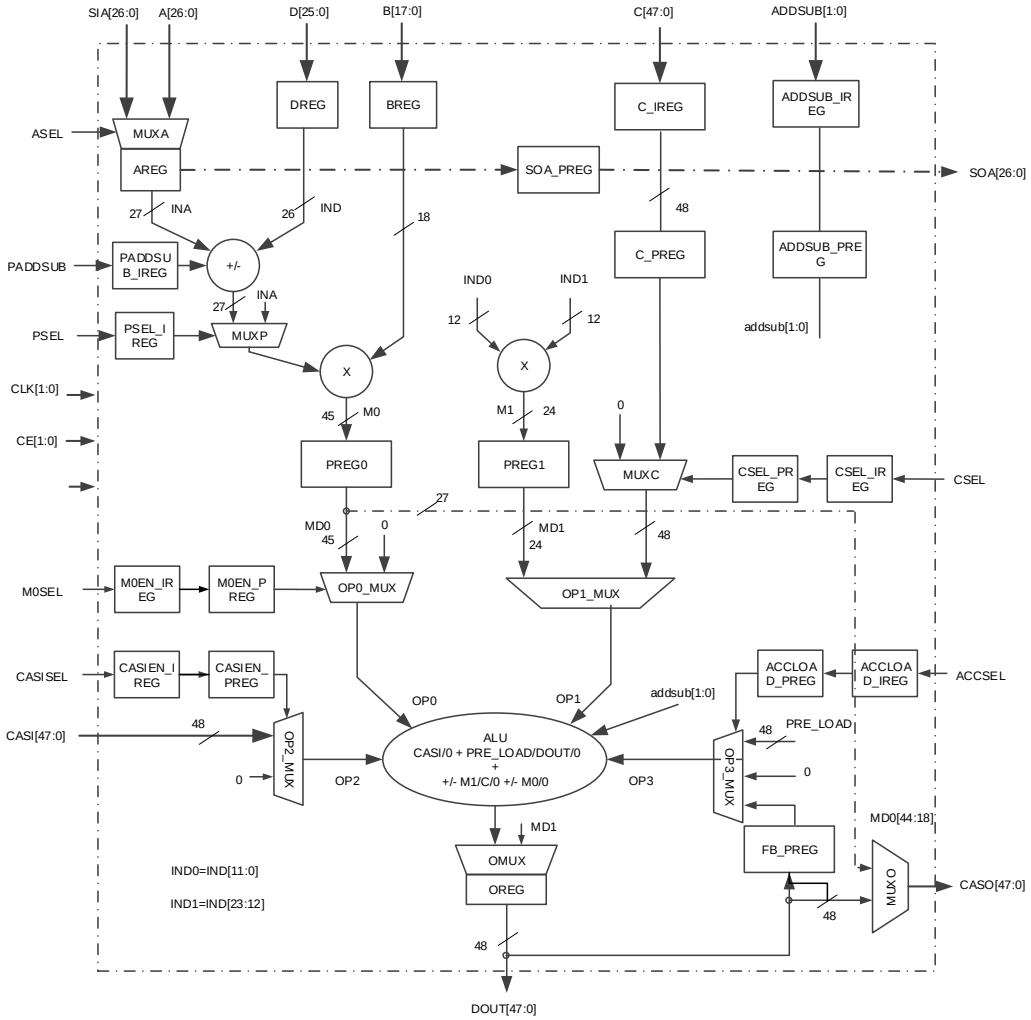
DSP ブロックの機能及び特性は以下の通りです：

- 3 つの幅（12X12，27X18，27X36）の乗算器
- 26 ビットの前置加算器
- 48 ビットの ALU
- シフト機能をポート
- 複数の乗算器のカスケード接続によるデータ幅の拡大をサポート
- 27X18 乗算器の累積、乗算加算機能をサポート
- 2 つの 12X12 乗算器の加算後の累積をサポート
- レジスタのパイプラインとバイパス機能をサポート
- 符号付きデータ操作

3 DSP の構造

Arora V FPGA 製品の DSP リソースは、FPGA のアレイに行として配置されています。DSP は、乗算、前置加算、累積、シフトなどの機能を実行することができます。 MULT、PADD、48 ビット ALU などの機能ブロックから構成されています。各 DSP は 3 つの CFU を占有します。各 DSP は、2 つの独立したクロック信号、2 つの独立したクロックイネーブル信号、2 つの独立したリセット信号を持っています。レジスタレベルは最大 4 つです(すなわち input reg, pipe reg, out reg, fb preg)。

図 3-1 DSP の構造



DSP のポートの説明及び意味は、表 3-1 に示すとおりです。内部レジスタは表 3-2 に示すとおりです。また、入力信号 CLK、CE、および RESET はレジスタを制御するために使用されます。

表 3-1 DSP のポートの説明

ポート名	I/O タイプ	説明
A[26:0]	I	27-bit データ入力 A
B[17:0]	I	18-bit データ入力 B
C[47:0]	I	48-bit データ入力 C
D[25:0]	I	26-bit データ入力 D
SIA[26:0]	I	カスケード接続に使用されるシフトデータ入力 A。入力信号 SIA は、前の隣接する DSP ブロックの出力信号 SOA に直接接続されま

ポート名	I/O タイプ	説明
		す。
CASI[47:0]	I	前の DSP ブロックの CASO からの、カスケード接続に使用される 48-bit ALU 入力
CASISEL	I	48-bit ALU 入力 CASI/0 の制御信号
ASEL	I	前置加算器の A 入力選択
PSEL	I	乗算器の A 入力選択
PADDSUB	I	前置加算器のロジック加算または減算を選択するために使用される前置加算器の操作制御信号
CLK[1:0]	I	クロック入力
CE[1:0]	I	クロックイネーブル信号、アクティブ High
RESET[1:0]	I	同期モード/非同期モードをサポートするリセット信号、アクティブ High
ADDSUB[1:0]	I	M0/0,M1/C/0 のロジック加算または減算を選択するために使用される、48-bit ALU の操作制御信号
CSEL	I	48-bit ALU 入力 C/0 制御信号
ACCSEL	I	48-bit ALU 入力 PRE_LOAD/DOUT 制御信号
M0SEL	I	48-bit ALU 入力 M0/0 制御信号
SOA[26:0]	O	シフトデータ出力 A
DOUT[47:0]	O	DSP 出力データ
CASO[47:0]	O	カスケード接続用。次の DSP ブロックに出力されます。

表 3-2 DSP ブロックの内部レジスタの説明

レジスタ	説明および関連属性
AREG	A 入力レジスタ
BREG	B 入力レジスタ
C_IREG	C 入力レジスタ
DREG	D 入力レジスタ
ADDSUB_IREG	ADDSUB 入力レジスタ
PADDSUB_IREG	PADDSUB 入力レジスタ
PSEL_IREG	PSEL 入力レジスタ

レジスタ	説明および関連属性
M0SEL_IREG	M0SEL 入力レジスタ
CASISEL_IREG	CASISEL 入力レジスタ
CSEL_IREG	CSEL 入力レジスタ
ACCSEL_IREG	ACCSEL 入力レジスタ
C_PREG	C パイプライン入力レジスタ
ADDSUB_PREG	ADDSUB パイプライン入力レジスタ
M0SEL_PREG	M0SEL パイプライン入力レジスタ
CASISEL_PREG	CASISEL パイプライン入力レジスタ
CSEL_PREG	CSEL パイプライン入力レジスタ
ACCSEL_PREG	ACCSEL パイプライン入力レジスタ
OREG	DOUT 出力レジスタ
PREG0	左乗算器パイプライン出力レジスタ
PREG1	右乗算器パイプライン出力レジスタ
FB_PREG	フィードバック出力パイプラインレジスタ
SOA_PREG	シフト出力パイプラインレジスタ

4 DSP プリミティブ

4.1 MULT

MULT(Multiplier)は DSP の乗算器です。A と B は乗算器の乗数入力信号で、DOUT は積の出力信号です。次の乗算を実現できます。

$$DOUT = A * B$$

$$DOUT = (A \pm D) * B$$

各 DSP には、乗算に使用される 2 つの乗算器があります。Multiplier は、データ幅によって 12x12、27x36 などの乗算器に構成でき、それぞれプリミティブの MULT12x12、MULT27x36 に対応します。27x36 乗算器に構成するには、2 つの DSP ブロックが必要となります。

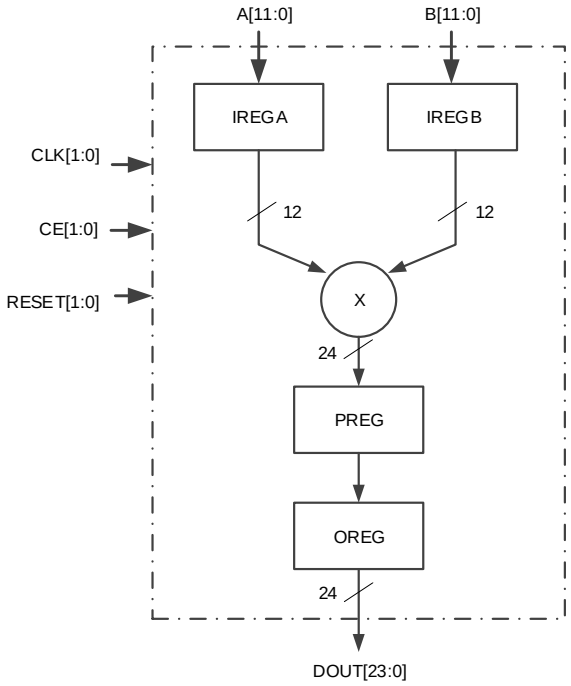
4.1.1 MULT12X12

プリミティブの紹介

MULT12X12(12x12 Multiplier)は 12 ビットの乗算を実現する 12x12 の乗算器です。

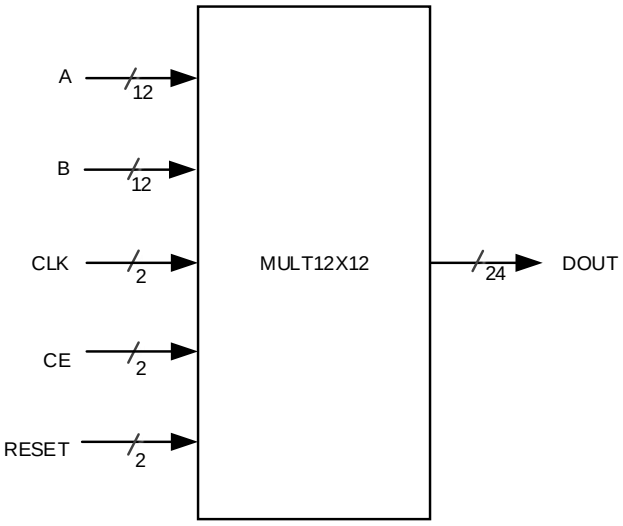
構造

図 4-1 MULT12X12 の構造



ポート図

図 4-2 MULT12X12 のポート図



ポートの説明

表 4-1 MULT12X12 のポートの説明

ポート	I/O	説明
A[11:0]	入力	12-bit データ入力信号 A
B[11:0]	入力	12-bit データ入力信号 B
CLK[1:0]	入力	クロック入力信号
CE[1:0]	入力	クロックイネーブル信号、アクティブ High
RESET[1:0]	入力	リセット入力信号、アクティブ High
DOUT[23:0]	出力	データ出力信号

パラメータの説明

表 4-2 MULT12X12 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 A レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
AREG_CE	CE0，CE1	CE0	入力 A レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
AREG_RESET	RESET0，RESET1	RESET0	入力 A レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
BREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 B レジスタのクロック制御信号 ● BYPASS：バイパスモード。

パラメータ	範囲	デフォルト	説明
			● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
BREG_CE	CE0，CE1	CE0	入力 B レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
BREG_RESET	RESET0，RESET1	RESET0	入力 B レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	Pipeline レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
PREG_CE	CE0，CE1	CE0	Pipeline レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
PREG_RESET	RESET0，RESET1	RESET0	Pipeline レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
OREG_CLK	BYPASS，CLK0，	BYPASS	出力レジスタのクロック制御信号

パラメータ	範囲	デフォルト	説明
	CLK1		● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
OREG_CE	CE0，CE1	CE0	出力レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
OREG_RESET	RESET0，RESET1	RESET0	出力レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
MULT_RESET_MODE	SYNC，ASYNC	SYNC	リセットのモード ● SYNC：同期リセット ● ASYNC：非同期リセット

プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、5 IP の呼び出しを参照してください。

Verilog でのインスタンス化：

```
MULT12X12 mult12x12_inst (
    .DOUT(dout),
    .A(a),
    .B(b),
    .CLK(clk),
    .CE(ce),
    .RESET(reset)
```

```
);

defparam mult12x12_inst.AREG_CLK = "BYPASS";
defparam mult12x12_inst.AREG_CE = "CE0";
defparam mult12x12_inst.AREG_RESET = "RESET0";
defparam mult12x12_inst.BREG_CLK = "BYPASS";
defparam mult12x12_inst.BREG_CE = "CE0";
defparam mult12x12_inst.BREG_RESET = "RESET0";
defparam mult12x12_inst.PREG_CLK = "BYPASS";
defparam mult12x12_inst.PREG_CE = "CE0";
defparam mult12x12_inst.PREG_RESET = "RESET0";
defparam mult12x12_inst.OREG_CLK = "BYPASS";
defparam mult12x12_inst.OREG_CE = "CE0";
defparam mult12x12_inst.OREG_RESET = "RESET0";
defparam mult12x12_inst.MULT_RESET_MODE = "SYNC";
```

VHDL でのインスタンス化：

```
COMPONENT MULT12X12
    GENERIC (
        AREG_CLK : string := "BYPASS";
        AREG_CE : string := "CE0";
        AREG_RESET : string := "RESET0";
        BREG_CLK : string := "BYPASS";
        BREG_CE : string := "CE0";
        BREG_RESET : string := "RESET0";
        PREG_CLK : string := "BYPASS";
        PREG_CE : string := "CE0";
        PREG_RESET : string := "RESET0";
        OREG_CLK : string := "BYPASS";
        OREG_CE : string := "CE0";
        OREG_RESET : string := "RESET0";
        MULT_RESET_MODE : string := "SYNC"
```

```
    );
    PORT (
        DOUT: out std_logic_vector(23 downto 0);
        A: in std_logic_vector(11 downto 0);
        B: in std_logic_vector(11 downto 0);
        CLK: in std_logic_vector(1 downto 0);
        CE: in std_logic_vector(1 downto 0);
        RESET: in std_logic_vector(1 downto 0)
    );
end COMPONENT;
mult12x12_inst: MULT12X12
    GENERIC MAP (
        AREG_CLK => "BYPASS",
        AREG_CE => "CE0",
        AREG_RESET => "RESET0",
        BREG_CLK => "BYPASS",
        BREG_CE => "CE0",
        BREG_RESET => "RESET0",
        PREG_CLK => "BYPASS",
        PREG_CE => "CE0",
        PREG_RESET => "RESET0",
        OREG_CLK => "BYPASS",
        OREG_CE => "CE0",
        OREG_RESET => "RESET0",
        MULT_RESET_MODE => "SYNC"
    )
    PORT MAP (
        DOUT => dout,
        A => a,
        B => b,
        CLK => CLK_i,
```

```
        CE => CE_i,
        RESET => RESET_i
);
```

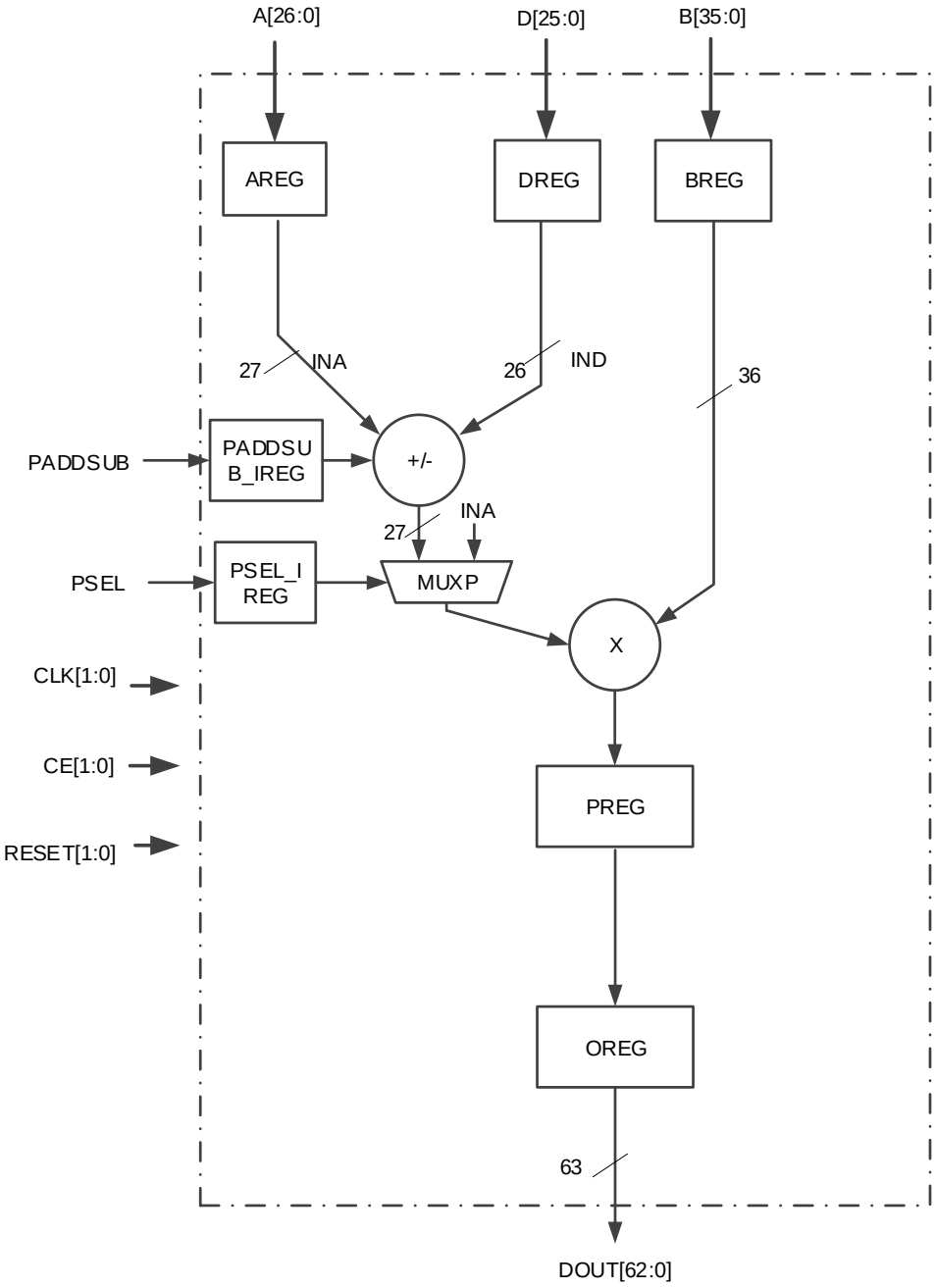
4.1.2 MULT27X36

プリミティブの紹介

MULT27X36(27x36 Multiplier)は 27 ビット X36 ビットの乗算を実現する 27x36 の乗算器です。

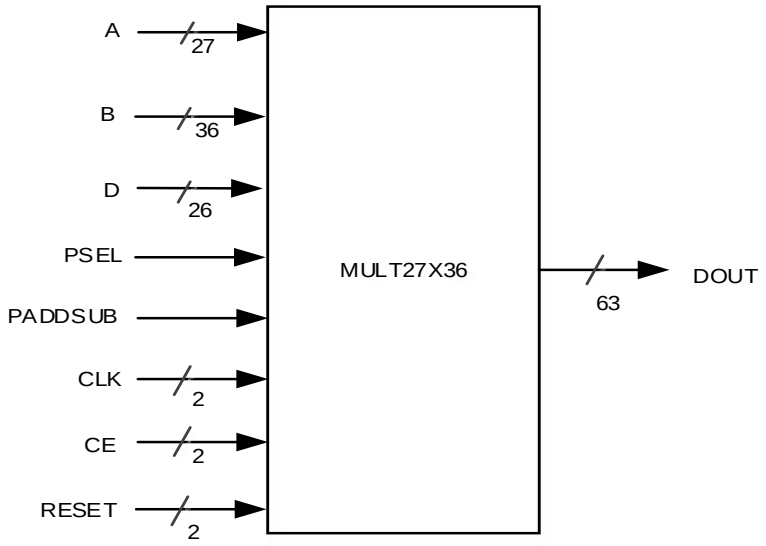
構造

図 4-3 MULT27X36 の構造



ポート図

図 4-4 MULT27X36 のポート図



ポートの説明

表 4-3 MULT27X36 のポートの説明

ポート	I/O	説明
A[26:0]	入力	27-bit データ入力信号 A
B[35:0]	入力	36-bit データ入力信号 B
D[25:0]	入力	26-bit データ入力信号 D
PSEL	入力	乗算器の A 入力ソース選択
PADDSUB	入力	前置加算器のロジック加算または減算を選択するために使用される前置加算器の操作制御信号
CLK[1:0]	入力	クロック入力信号
CE[1:0]	入力	クロックイネーブル信号、アクティブ High
RESET[1:0]	入力	リセット入力信号、アクティブ High
DOUT[62:0]	出力	データ出力信号

パラメータの説明

表 4-4 MULT27X36 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 A レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタの制御信号 clk は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタの制御信号 clk は CLK[1]から供給されます。
AREG_CE	CE0，CE1	CE0	入力 A レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
AREG_RESET	RESET0，RESET1	RESET0	入力 A レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
BREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 B レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタの制御信号 clk は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタの制御信号 clk は CLK[1]から供給されます。
BREG_CE	CE0，CE1	CE0	入力 B レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
BREG_RESET	RESET0，RESET1	RESET0	入力 B レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。

パラメータ	範囲	デフォルト	説明
DREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 D レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタの制御信号 clk は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタの制御信号 clk は CLK[1]から供給されます。
DREG_CE	CE0，CE1	CE0	入力 D レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
DREG_RESET	RESET0，RESET1	RESET0	入力 D レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
PADDSUB_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 PADDSUB レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタの制御信号 clk は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタの制御信号 clk は CLK[1]から供給されます。
PADDSUB_IREG_CE	CE0，CE1	CE0	入力 PADDSUB レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
PADDSUB_IREG_RESET	RESET0，RESET1	RESET0	入力 PADDSUB レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
PSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 PSEL レジスタのクロック制御信号 ● BYPASS：バイパスモード。

パラメータ	範囲	デフォルト	説明
			● CLK0：レジスタモード。レジスタの制御信号 clk は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタの制御信号 clk は CLK[1]から供給されます。
PSEL_IREG_CE	CE0，CE1	CE0	入力 PSEL レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
PSEL_IREG_RESET	RESET0，RESET1	RESET0	入力 PSEL レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	Mult Pipeline レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタの制御信号 clk は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタの制御信号 clk は CLK[1]から供給されます。
PREG_CE	CE0，CE1	CE0	Mult Pipeline レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
PREG_RESET	RESET0，RESET1	RESET0	Mult Pipeline レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
OREG_CLK	BYPASS，CLK0，CLK1	BYPASS	出力レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタの制御信号 clk は CLK[0]から供給されます。

パラメータ	範囲	デフォルト	説明
			● CLK1：レジスタモード。レジスタの制御信号 clk は CLK[1]から供給されます。
OREG_CE	CE0，CE1	CE0	出力レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
OREG_RESET	RESET0，RESET1	RESET0	出力レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
MULT_RESET_MODE	SYNC，ASYNC	SYNC	リセットのモード
P_SEL	1'b0，1'b1	1'b0	選択の静的制御。A または A +/- D に接続することを選択します。 ● 1'b0：INA に直接接続することを選択します。 ● 1'b1：前置加算器を選択します
DYN_P_SEL	FALSE，TRUE	FALSE	INA または INA +/- D の選択の動的制御。 ● FALSE：P_SEL により mult0 が INA または INA +/- D を選択するかを静的に制御します。 ● TRUE：入力 PSEL により mult0 が INA または INA +/- D を選択するかを動的に制御します。
P_ADDSUB	1'b0，1'b1	1'b0	前置加算器の加算または減算の選択の静的制御 ● 1’b0：加算 ● 1’b1：減算
DYN_P_ADDSUB	FALSE，TRUE	FALSE	前置加算器の加算または減算の選択の動的制御 ● FALSE：P_ADDSUB により前置加算器の加算または減算の選択を静的に制御します。 ● TRUE：入力 PSEL により前置加算器の加算または減算の選択を動的に制御しま

パラメータ	範囲	デフォルト	説明
			す。

プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、5 IP の呼び出しを参照してください。

Verilog でのインスタンス化：

```
MULT27X36 mult27x36_inst (
    .DOUT(dout),
    .A({gw_gnd,a[25:0]}),
    .B(b),
    .D(d),
    .PSEL(gw_gnd),
    .PADDSUB(gw_gnd),
    .CLK({gw_gnd,clk}),
    .CE({gw_gnd,ce}),
    .RESET({gw_gnd,reset})
);

defparam mult27x36_inst.AREG_CLK = "CLK0";
defparam mult27x36_inst.AREG_CE = "CE0";
defparam mult27x36_inst.AREG_RESET = "RESET0";
defparam mult27x36_inst.BREG_CLK = "CLK0";
defparam mult27x36_inst.BREG_CE = "CE0";
defparam mult27x36_inst.BREG_RESET = "RESET0";
defparam mult27x36_inst.DREG_CLK = "CLK0";
defparam mult27x36_inst.DREG_CE = "CE0";
defparam mult27x36_inst.DREG_RESET = "RESET0";
defparam mult27x36_inst.PADDSUB_IREG_CLK = "BYPASS";
defparam mult27x36_inst.PADDSUB_IREG_CE = "CE0";
```

```
defparam mult27x36_inst.PADDSUB_IREG_RESET = "RESET0";
defparam mult27x36_inst.PREG_CLK = "BYPASS";
defparam mult27x36_inst.PREG_CE = "CE0";
defparam mult27x36_inst.PREG_RESET = "RESET0";
defparam mult27x36_inst.PSEL_IREG_CLK = "BYPASS";
defparam mult27x36_inst.PSEL_IREG_CE = "CE0";
defparam mult27x36_inst.PSEL_IREG_RESET = "RESET0";
defparam mult27x36_inst.OREG_CLK = "CLK0";
defparam mult27x36_inst.OREG_CE = "CE0";
defparam mult27x36_inst.OREG_RESET = "RESET0";
defparam mult27x36_inst.MULT_RESET_MODE = "SYNC";
defparam mult27x36_inst.DYN_P_SEL = "FALSE";
defparam mult27x36_inst.P_SEL = "1'b1";
defparam mult27x36_inst.DYN_P_ADDSUB = "FALSE";
defparam mult27x36_inst.P_ADDSUB = "1'b0";
```

VHDL でのインスタンス化：

```
COMPONENT MULT27X36
        GENERIC (AREG_CLK:string:="CLK0";
                AREG_CE:string:="CE0";
                AREG_RESET:string:="RESET0";
                BREG_CLK:string:="CLK0";
                BREG_CE:string:="CE0";
                BREG_RESET:string:="RESET0";
                DREG_CLK:string:="CLK0";
                DREG_CE:string:="CE0";
                DREG_RESET:string:="RESET0";
                PADDSUB_IREG_CLK:string:="CLK0";
                PADDSUB_IREG_CE:string:="CE0";
                PADDSUB_IREG_RESET:string:="RESET0";
                PREG_CLK:string:="CLK0";
                PREG_CE:string:="CE0";
```

```
                PREG_RESET:string:="RESET0";
                PSEL_IREG_CLK:string:="CLK0";
                PSEL_IREG_CE:string:="CE0";
                PSEL_IREG_RESET:string:="RESET0";
                OREG_CLK:string:="CLK0";
                OREG_CE:string:="CE0";
                OREG_RESET:string:="RESET0";
                MULT_RESET_MODE:string:="ASYNC";
                DYN_P_SEL:string:="FALSE";
                P_SEL:bit:='0';
                DYN_P_ADDSUB:string:="FALSE";
                P_ADDSUB:bit:='0';
        );
        PORT(
                DOUT:OUT std_logic_vector(62 downto 0);
                A:IN std_logic_vector(26 downto 0);
                B:IN std_logic_vector(35 downto 0);
                D:IN std_logic_vector(25 downto 0);
                PSEL:IN std_logic;
                PADDSUB:IN std_logic;
                CLK:IN std_logic_vector(1 downto 0);
                CE:IN std_logic_vector(1 downto 0);
                RESET:IN std_logic_vector(1 downto 0)
        );
END COMPONENT;
uut:MULT27X36
        GENERIC MAP (AREG_CLK=>"CLK0",
                        AREG_CE=>"CE0",
                        AREG_RESET=>"RESET0",

                        BREG_CLK=>"CLK0",
```

```
                BREG_CE=>"CE0",
                BREG_RESET=>"RESET0",
                DREG_CLK=>"CLK0",
                DREG_CE=>"CE0",
                DREG_RESET=>"RESET0",
                PADDSUB_IREG_CLK=>"CLK0",
                PADDSUB_IREG_CE=>"CE0",
                PADDSUB_IREG_RESET=>"RESET0",
                PREG_CLK=>"CLK0",
                PREG_CE=>"CE0",
                PREG_RESET=>"RESET0",
                PSEL_IREG_CLK=>"CLK0",
                PSEL_IREG_CE=>"CE0",
                PSEL_IREG_RESET=>"RESET0",
                OREG_CLK=>"CLK0",
                OREG_CE=>"CE0",
                OREG_RESET=>"RESET0",
                MULT_RESET_MODE=>"ASYNC",
                DYN_P_SEL=>"FALSE",
                P_SEL=>'1',
                DYN_P_ADDSUB=>"FALSE",
                P_ADDSUB=>'0'
    )
    PORT MAP (
        DOUT=>dout,
        A=>A_i,
        B=>b,
        D=>d,
        PSEL=>gw_gnd,
        PADDSUB=>gw_gnd,
        CLK=>clk,
```

```
        CE=>ce,
        RESET=>reset
);
```

4.2 MULTALU

4.2.1 MULTALU27X18

MULTALU モードでは、乗算器の出力の 48-bit ALU 演算が実現されます。対応するプリミティブは MULTALU27X18 です。MULTALU27X18 には 16 の演算モードがあります。

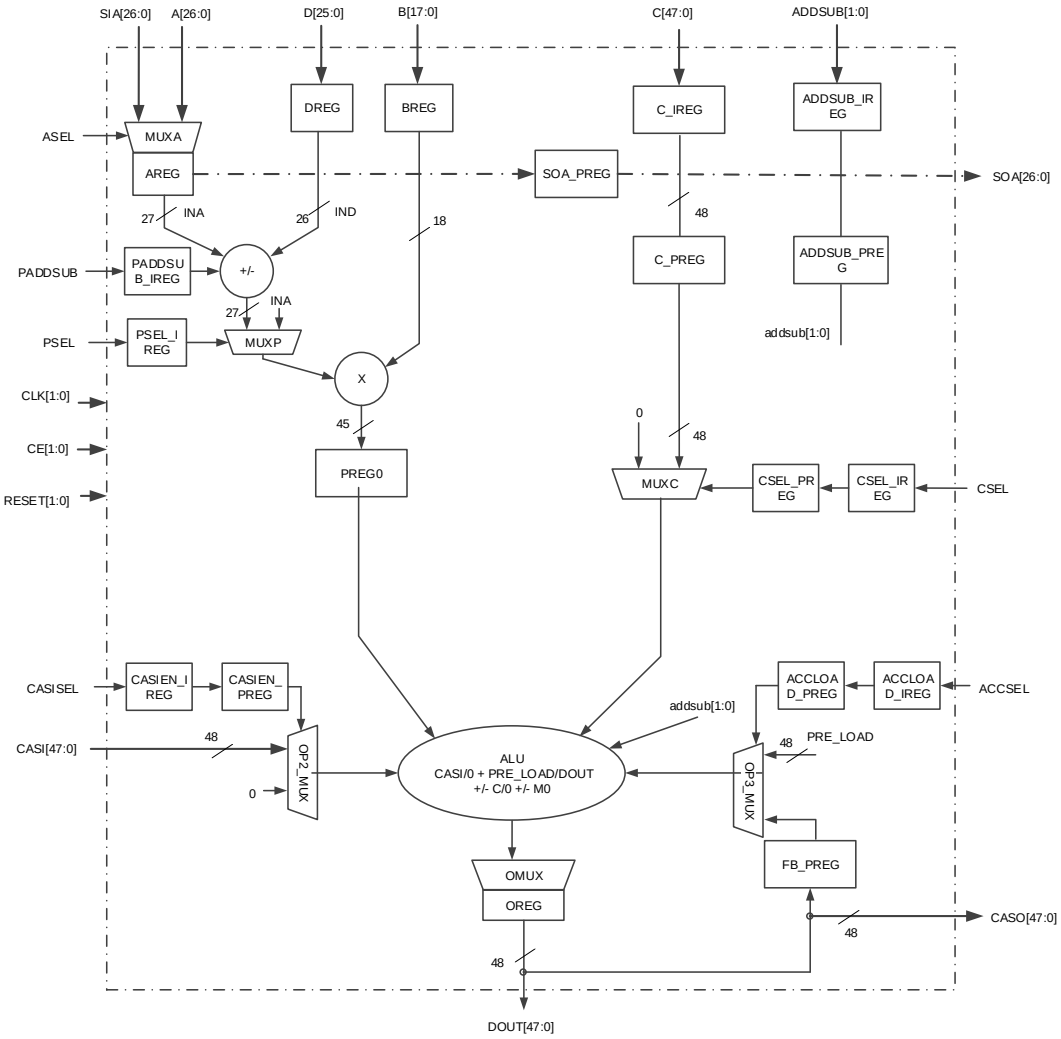
$$DOUT = \pm(A * B)$$
$$DOUT = \pm(A * B) \pm C$$
$$DOUT = \pm(A * B) + DOUT$$
$$DOUT = \pm(A * B) \pm C + DOUT$$
$$DOUT = \pm\big((A \pm D) * B\big)$$
$$DOUT = \pm\big((A \pm D) * B\big) \pm C$$
$$DOUT = \pm\big((A \pm D) * B\big) + DOUT$$
$$DOUT = \pm\big((A \pm D) * B\big) \pm C + DOUT$$
$$DOUT = \pm(A * B) + CASI$$
$$DOUT = \pm(A * B) + CASI \pm C$$
$$DOUT = \pm(A * B) + CASI + DOUT$$
$$DOUT = \pm(A * B) + CASI + DOUT \pm C$$
$$DOUT = \pm(SIA * B)$$
$$DOUT = \pm(SIA * B) \pm C$$
$$DOUT = \pm(SIA * B) + DOUT$$
$$DOUT = \pm(SIA * B) + DOUT \pm C$$

プリミティブの紹介

MULTALU 27X18 (27x18 Multiplier with ALU)は、ALU 機能付きの 27X18 の乗算器です。乗算、乗算加算、累積、乗算累積、乗算/乗算加算/累積/乗算累積に基づくシフト、乗算/乗算加算/累積/乗算累積に基づくカスケード、および乗算/乗算加算/累積/乗算累積に基づく前置加算と前置減算を実装します。

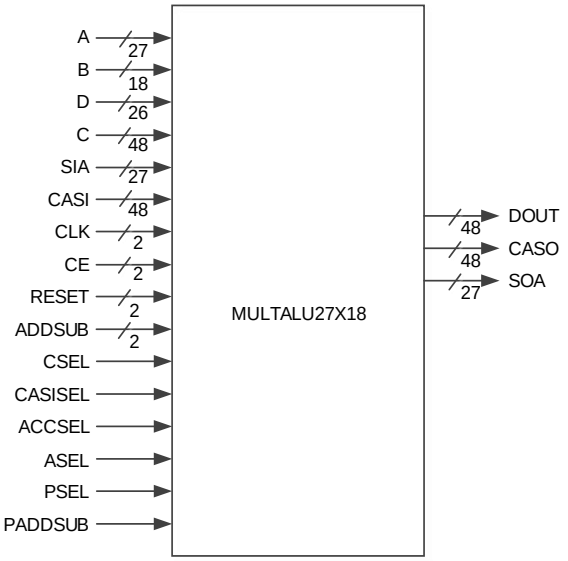
構造

図 4-5 MULTALU27X18 の構造



ポート図

図 4-6 MULTALU27X18 のポート図



ポートの説明

表 4-5 MULTALU27X18 のポートの説明

ポート	I/O	説明
A[26:0]	入力	27-bit データ入力信号 A
B[17:0]	入力	18-bit データ入力信号 B
D[25:0]	入力	26-bit 前置加算器データ入力信号 D
C[47:0]	入力	48-bit ALU データ入力信号 C
CASI[47:0]	入力	48-bit ALU カスケード接続入力信号
SIA[26:0]	入力	27-bit シフトデータ入力信号(シフト用)入力信号 SIA は、前の隣接する DSP ブロックのシフト出力信号 SOA に直接接続されます。
CLK[1:0]	入力	クロック入力信号
CE[1:0]	入力	クロックイネーブル信号、アクティブ High
RESET[1:0]	入力	リセット入力信号、アクティブ High
ADDSUB[1:0]	入力	加算/減算の動的制御信号
CSEL	入力	48-bit ALU の入力 C または 0 の選択の制御信号
CASISEL	入力	48-bit ALU の入力 CASI または 0 の選択の制御信号
ACCSEL	入力	48-bit ALU の入力 DOUT または PRE_LOAD の選択の制御信号
ASEL	入力	前置加算器または乗算器の A または SIA の選択の制御信号
PSEL	入力	乗算器の INA または INA+/-D の選択の制御信号
PADDSUB	入力	前置加算器のロジック加算または減算を選択するために使用される前置加算器の操作制御信号
SOA[26:0]	出力	シフトデータ出力信号
DOUT[47:0]	出力	データ出力信号
CASO[47:0]	出力	48-bit カスケード接続出力信号

パラメータの説明

表 4-6 MULTALU27X18 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 A(A または SIA)レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
AREG_CE	CE0，CE1	CE0	入力 A(A または SIA)レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
AREG_RESET	RESET0，RESET1	RESET0	入力 A(A または SIA)レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
BREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 B レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
BREG_CE	CE0，CE1	CE0	入力 B レジスタのクロックイネーブル制御信号

パラメータ	範囲	デフォルト	説明
			● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
BREG_RESET	RESET0，RESET1	RESET0	入力 B レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
DREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 D レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
DREG_CE	CE0，CE1	CE0	入力 D レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
DREG_RESET	RESET0，RESET1	RESET0	入力 D レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
C_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 C レジスタのクロック制御信号

パラメータ	範囲	デフォルト	説明
			● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
C_IREG_CE	CE0，CE1	CE0	入力 C レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
C_IREG_RESET	RESET0，RESET1	RESET0	入力 C レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
C_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 C pipeline レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
C_PREG_CE	CE0，CE1	CE0	入力 C pipeline レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給され

パラメータ	範囲	デフォルト	説明
			ます。
C_PREG_RESET	RESET0，RESET1	RESET0	入力 C pipeline レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
ADDSUB0_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ADDSUB[0]レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ADDSUB0_IREG_CE	CE0，CE1	CE0	入力 ADDSUB[0]レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
ADDSUB0_IREG_RESET	RESET0，RESET1	RESET0	入力 ADDSUB[0]レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
ADDSUB1_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ADDSUB[1]レジスタのクロック制御信号 ● BYPASS：バイパスモード。

パラメータ	範囲	デフォルト	説明
			● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ADDSUB1_IREG_CE	CE0，CE1	CE0	入力 ADDSUB[1]レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
ADDSUB1_IREG_RESET	RESET0，RESET1	RESET0	入力 ADDSUB[1]レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
ADDSUB0_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ADDSUB[0] pipeline レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ADDSUB0_PREG_CE	CE0，CE1	CE0	入力 ADDSUB[0] pipeline レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給され

パラメータ	範囲	デフォルト	説明
			ます。
ADDSUB0_PREG_RESET	RESET0，RESET1	RESET0	入力 ADDSUB[0] pipeline レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
ADDSUB1_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ADDSUB[1] pipeline レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ADDSUB1_PREG_CE	CE0，CE1	CE0	入力 ADDSUB[1] pipeline レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
ADDSUB1_PREG_RESET	RESET0，RESET1	RESET0	入力 ADDSUB[1] pipeline レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
PADDSUB_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 PADDSUB レジスタのクロック制御信号 ● BYPASS：バイパスモード。

パラメータ	範囲	デフォルト	説明
			● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
PADDSUB_IREG_CE	CE0，CE1	CE0	入力 PADDSUB レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
PADDSUB_IREG_RESET	RESET0，RESET1	RESET0	入力 PADDSUB レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
PSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 PSEL レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
PSEL_IREG_CE	CE0，CE1	CE0	入力 PSEL レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給され

パラメータ	範囲	デフォルト	説明
			ます。
PSEL_IREG_RESET	RESET0，RESET1	RESET0	入力 PSEL レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
CSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 CSEL レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
CSEL_IREG_CE	CE0，CE1	CE0	入力 CSEL レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
CSEL_IREG_RESET	RESET0，RESET1	RESET0	入力 CSEL レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
CSEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 CSEL pipeline レジスタのクロック制御信号 ● BYPASS：バイパスモード。

パラメータ	範囲	デフォルト	説明
			● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
CSEL_PREG_CE	CE0，CE1	CE0	入力 CSEL pipeline レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
CSEL_PREG_RESET	RESET0，RESET1	RESET0	入力 CSEL pipeline レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
CASISEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 CASISEL レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
CASISEL_IREG_CE	CE0，CE1	CE0	入力 CASISEL レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給され

パラメータ	範囲	デフォルト	説明
			ます。
CASISEL_IREG_RESET	RESET0，RESET1	RESET0	入力 CASISEL レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
CASISEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 CASISEL pipeline レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
CASISEL_PREG_CE	CE0，CE1	CE0	入力 CASISEL pipeline レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
CASISEL_PREG_RESET	RESET0，RESET1	RESET0	入力 CASISEL pipeline レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
ACCSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ACCSEL レジスタのクロック制御信号 ● BYPASS：バイパスモード。

パラメータ	範囲	デフォルト	説明
			● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ACCSEL_IREG_CE	CE0，CE1	CE0	入力 ACCSEL レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
ACCSEL_IREG_RESET	RESET0，RESET1	RESET0	入力 ACCSEL レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
ACCSEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ACCSEL pipeline レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ACCSEL_PREG_CE	CE0，CE1	CE0	入力 ACCSEL pipeline レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給され

パラメータ	範囲	デフォルト	説明
			ます。
ACCSEL_PREG_RESET	RESET0，RESET1	RESET0	入力 ACCSEL pipeline レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	M0 Pipeline レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
PREG_CE	CE0，CE1	CE0	M0 Pipeline レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
PREG_RESET	RESET0，RESET1	RESET0	M0 Pipeline レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
FB_PREG_EN	FALSE，TRUE	FALSE	フィードバック出力 pipeline レジスタの制御パラメータ ● FALSE：バイパスモード。

パラメータ	範囲	デフォルト	説明
			● TRUE：レジスタモード。制御信号 clk/ce/reset は、OREG と同じです。
SOA_PREG_EN	FALSE，TRUE	FALSE	シフト出力 SOA pipeline レジスタの制御パラメータ ● FALSE：バイパスモード。 ● TRUE：レジスタモード。制御信号 clk/ce/reset は、AREG と同じです。
OREG_CLK	BYPASS，CLK0，CLK1	BYPASS	出力レジスタのクロック制御信号 ● BYPASS：バイパスモード。 ● CLK0：レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1：レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
OREG_CE	CE0，CE1	CE0	出力レジスタのクロックイネーブル制御信号 ● CE0：レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1：レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
OREG_RESET	RESET0，RESET1	RESET0	出力レジスタのリセット制御信号 ● RESET0：レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1：レジスタのリセット制御信号は RESET[1]から供給されます。
MULT_RESET_MODE	SYNC，ASYNC	SYNC	リセットモードの構成 ● SYNC：同期リセット ● ASYNC：非同期リセット
PRE_LOAD	48’h000000000000 ~48’hFFFFFFFFFFFF	48’h0	PRE_LOAD の初期値

パラメータ	範囲	デフォルト	説明
A_SEL	1’b0，1’b1	1’b0	A または SIA の選択の静的制御 ● 1’b0：A を選択します ● 1’b1：SIA を選択します
DYN_A_SEL	FALSE，TRUE	FALSE	A または SIA の選択の動的制御 ● FALSE：A_SEL により A または SIA の選択を静的に制御します。 ● TRUE：入力 ASEL により A または SIA の選択を動的に制御します。
P_SEL	1’b0，1’b1	1’b0	INA または INA+/-D の選択の静的制御 ● 1’b0：INA を選択します ● 1’b1：INA+/-D を選択します
DYN_P_SEL	FALSE，TRUE	FALSE	INA または INA+/-D の選択の動的制御 ● FALSE：P_SEL により INA または INA+/-D の選択を静的に制御します。 ● TRUE：入力 PSEL により INA または INA+/-D の選択を動的に制御します。
P_ADDSUB	1’b0，1’b1	1’b0	前置加算器の加算または減算の選択の静的制御 ● 1’b0：加算 ● 1’b1：減算
DYN_P_ADDSUB	FALSE，TRUE	FALSE	前置加算器の加算または減算の選択の動的制御 ● FALSE：P_ADDSUB により前置加算器の加算または減算の選択を静的に制御します。 ● TRUE：入力 PADDSUB により前置加算器の加算または減算の選択を動的に制御します。
ADD_SUB_0	1’b0，1’b1	1’b0	M0/0 の加算または減算の選択の静的制御 ● 1’b0：加算 ● 1’b1：減算
DYN_ADD_SUB_	FALSE，TRUE	FALSE	M0/0 の加算または減算の選択の動的制

パラメータ	範囲	デフォルト	説明
0			御 ● FALSE：ADD_SUB_0 により M0/0 の加算または減算の選択を静的に制御します。 ● TRUE：入力 ADDSUB[0]により M0/0 の加算または減算の選択を動的に制御します。
ADD_SUB_1	1’b0，1’b1	1’b0	C/0 の加算または減算の選択の静的制御 ● 1’b0：加算 ● 1’b1：減算
DYN_ADD_SUB_1	FALSE，TRUE	FALSE	C/0 の加算または減算の選択の動的制御 ● FALSE：ADD_SUB_1 により C/0 の加算または減算の選択を静的に制御します。 ● TRUE：入力 ADDSUB[1]により M1/C/0 の加算または減算の選択を動的に制御します。
CASI_SEL	1’b0，1’b1	1’b0	CASI または 0 の選択の静的制御 ● 1’b0：0 を選択します ● 1’b1：CASI を選択します
DYN_CASI_SEL	FALSE，TRUE	FALSE	CASI または 0 の選択の動的制御 ● FALSE：CASI_SEL により CASI または 0 の選択を静的に制御します。 ● TRUE：入力 CASISEL により CASI または 0 の選択を動的に制御します。
ACC_SEL	1’b0，1’b1	1’b0	PRE_LOAD または DOUT の選択の静的制御 ● 1’b0：PRE_LOAD を選択します ● 1’b1：出力フィードバックを選択します
DYN_ACC_SEL	FALSE，TRUE	FALSE	PRE_LOAD または DOUT の選択の動的制御 ● FALSE：ACC_SEL により PRE_LOAD または出力フィードバックの選択を静的に制御します。

パラメータ	範囲	デフォルト	説明
			● TRUE：入力 ACCSEL により PRE_LOAD または出力フィードバックの選択を動的に制御します。
C_SEL	1’b0，1’b1	1’b0	C または 0 の選択の静的制御 ● 1’b0：0 を選択します ● 1’b1：C を選択します
DYN_C_SEL	FALSE，TRUE	FALSE	C または 0 の選択の動的制御 ● FALSE：C_SEL により C または 0 の選択を静的に制御します。 ● TRUE：入力 CSEL により C または 0 の選択を動的に制御します。
MULT12X12_EN	FALSE，TRUE	FALSE	M0 モードの制御 ● FALSE：27X18 モード ● TRUE：12X12 モード

プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、5 IP の呼び出しを参照してください。

Verilog でのインスタンス化：

```
MULTALU27X18 multalu27x18_inst (
    .DOUT(dout),
    .CASO(caso),
    .SOA(soa),
    .A({a[25],a[25:0]}),
    .B(b),
    .C(c),
    .D(d),
    .SIA({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .CASI(casi),
    .ACCSEL(gw_gnd),
```

```
        .CASISEL(gw_gnd),
        .ASEL(gw_gnd),
        .PSEL(gw_gnd),
        .CSEL(gw_gnd),
        .ADDSUB({gw_gnd,gw_gnd}),
        .PADDSUB(gw_gnd),
        .CLK({gw_gnd,clk}),
        .CE({gw_gnd,ce}),
        .RESET({gw_gnd,reset})
    );
    defparam multalu27x18_inst.AREG_CLK = "CLK0";
    defparam multalu27x18_inst.AREG_CE = "CE0";
    defparam multalu27x18_inst.AREG_RESET = "RESET0";
    defparam multalu27x18_inst.BREG_CLK = "CLK0";
    defparam multalu27x18_inst.BREG_CE = "CE0";
    defparam multalu27x18_inst.BREG_RESET = "RESET0";
    defparam multalu27x18_inst.DREG_CLK = "CLK0";
    defparam multalu27x18_inst.DREG_CE = "CE0";
    defparam multalu27x18_inst.DREG_RESET = "RESET0";
    defparam multalu27x18_inst.C_IREG_CLK = "CLK0";
    defparam multalu27x18_inst.C_IREG_CE = "CE0";
    defparam multalu27x18_inst.C_IREG_RESET = "RESET0";
    defparam multalu27x18_inst.PSEL_IREG_CLK = "BYPASS";
    defparam multalu27x18_inst.PSEL_IREG_CE = "CE0";
    defparam multalu27x18_inst.PSEL_IREG_RESET = "RESET0";
    defparam multalu27x18_inst.PADDSUB_IREG_CLK = "BYPASS";
    defparam multalu27x18_inst.PADDSUB_IREG_CE = "CE0";
    defparam multalu27x18_inst.PADDSUB_IREG_RESET = "RESET0";
    defparam multalu27x18_inst.ADDSUB0_IREG_CLK = "BYPASS";
    defparam multalu27x18_inst.ADDSUB0_IREG_CE = "CE0";
    defparam multalu27x18_inst.ADDSUB0_IREG_RESET = "RESET0";
```

```
defparam multalu27x18_inst.ADDSUB1_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.ADDSUB1_IREG_CE = "CE0";
defparam multalu27x18_inst.ADDSUB1_IREG_RESET = "RESET0";
defparam multalu27x18_inst.CSEL_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.CSEL_IREG_CE = "CE0";
defparam multalu27x18_inst.CSEL_IREG_RESET = "RESET0";
defparam multalu27x18_inst.CASISEL_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.CASISEL_IREG_CE = "CE0";
defparam multalu27x18_inst.CASISEL_IREG_RESET = "RESET0";
defparam multalu27x18_inst.ACCSEL_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.ACCSEL_IREG_CE = "CE0";
defparam multalu27x18_inst.ACCSEL_IREG_RESET = "RESET0";
defparam multalu27x18_inst.PREG_CLK = "BYPASS";
defparam multalu27x18_inst.PREG_CE = "CE0";
defparam multalu27x18_inst.PREG_RESET = "RESET0";
defparam multalu27x18_inst.ADDSUB0_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.ADDSUB0_PREG_CE = "CE0";
defparam multalu27x18_inst.ADDSUB0_PREG_RESET = "RESET0";
defparam multalu27x18_inst.ADDSUB1_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.ADDSUB1_PREG_CE = "CE0";
defparam multalu27x18_inst.ADDSUB1_PREG_RESET = "RESET0";
defparam multalu27x18_inst.CSEL_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.CSEL_PREG_CE = "CE0";
defparam multalu27x18_inst.CSEL_PREG_RESET = "RESET0";
defparam multalu27x18_inst.CASISEL_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.CASISEL_PREG_CE = "CE0";
defparam multalu27x18_inst.CASISEL_PREG_RESET = "RESET0";
defparam multalu27x18_inst.ACCSEL_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.ACCSEL_PREG_CE = "CE0";
defparam multalu27x18_inst.ACCSEL_PREG_RESET = "RESET0";
defparam multalu27x18_inst.C_PREG_CLK = "CLK0";
```

```
defparam multalu27x18_inst.C_PREG_CE = "CE0";
defparam multalu27x18_inst.C_PREG_RESET = "RESET0";
defparam multalu27x18_inst.FB_PREG_EN = "FALSE";
defparam multalu27x18_inst.SOA_PREG_EN = "FALSE";
defparam multalu27x18_inst.OREG_CLK = "CLK0";
defparam multalu27x18_inst.OREG_CE = "CE0";
defparam multalu27x18_inst.OREG_RESET = "RESET0";
defparam multalu27x18_inst.MULT_RESET_MODE = "SYNC";
defparam multalu27x18_inst.PRE_LOAD = 48'h000000000000;
defparam multalu27x18_inst.DYN_P_SEL = "FALSE";
defparam multalu27x18_inst.P_SEL = 1'b0;
defparam multalu27x18_inst.DYN_P_ADDSUB = "FALSE";
defparam multalu27x18_inst.P_ADDSUB = 1'b0;
defparam multalu27x18_inst.DYN_A_SEL = "FALSE";
defparam multalu27x18_inst.A_SEL = 1'b0;
defparam multalu27x18_inst.DYN_ADD_SUB_0 = "FALSE";
defparam multalu27x18_inst.ADD_SUB_0 = 1'b0;
defparam multalu27x18_inst.DYN_ADD_SUB_1 = "FALSE";
defparam multalu27x18_inst.ADD_SUB_1 = 1'b0;
defparam multalu27x18_inst.DYN_C_SEL = "FALSE";
defparam multalu27x18_inst.C_SEL = 1'b1;
defparam multalu27x18_inst.DYN_CASI_SEL = "FALSE";
defparam multalu27x18_inst.CASI_SEL = 1'b1;
defparam multalu27x18_inst.DYN_ACC_SEL = "FALSE";
defparam multalu27x18_inst.ACC_SEL = 1'b0;
defparam multalu27x18_inst.MULT12X12_EN = "FALSE";
```

VHDL でのインスタンス化：

```
COMPONENT MULTALU27X18
    GENERIC (
        AREG_CLK : string := "BYPASS";
        AREG_CE : string := "CE0";
```

```
AREG_RESET : string := "RESET0";
BREG_CLK : string := "BYPASS";
BREG_CE : string := "CE0";
BREG_RESET : string := "RESET0";
DREG_CLK : string := "BYPASS";
DREG_CE : string := "CE0";
DREG_RESET : string := "RESET0";
C_IREG_CLK : string := "BYPASS";
C_IREG_CE : string := "CE0";
C_IREG_RESET : string := "RESET0";
PSEL_IREG_CLK : string := "BYPASS";
PSEL_IREG_CE : string := "CE0";
PSEL_IREG_RESET : string := "RESET0";
PADDSUB_IREG_CLK : string := "BYPASS";
PADDSUB_IREG_CE : string := "CE0";
PADDSUB_IREG_RESET : string := "RESET0";
ADDSUB0_IREG_CLK : string := "BYPASS";
ADDSUB0_IREG_CE : string := "CE0";
ADDSUB0_IREG_RESET : string := "RESET0";
ADDSUB1_IREG_CLK : string := "BYPASS";
ADDSUB1_IREG_CE : string := "CE0";
ADDSUB1_IREG_RESET : string := "RESET0";
CSEL_IREG_CLK : string := "BYPASS";
CSEL_IREG_CE : string := "CE0";
CSEL_IREG_RESET : string := "RESET0";
CASISEL_IREG_CLK : string := "BYPASS";
CASISEL_IREG_CE : string := "CE0";
CASISEL_IREG_RESET : string := "RESET0";
ACCSEL_IREG_CLK : string := "BYPASS";
ACCSEL_IREG_CE : string := "CE0";
ACCSEL_IREG_RESET : string := "RESET0";
```

```
PREG_CLK : string := "BYPASS";
PREG_CE : string := "CE0";
PREG_RESET : string := "RESET0";
ADDSUB0_PREG_CLK : string := "BYPASS";
ADDSUB0_PREG_CE : string := "CE0";
ADDSUB0_PREG_RESET : string := "RESET0";
ADDSUB1_PREG_CLK : string := "BYPASS";
ADDSUB1_PREG_CE : string := "CE0";
ADDSUB1_PREG_RESET : string := "RESET0";
CSEL_PREG_CLK : string := "BYPASS";
CSEL_PREG_CE : string := "CE0";
CSEL_PREG_RESET : string := "RESET0";
CASISEL_PREG_CLK : string := "BYPASS";
CASISEL_PREG_CE : string := "CE0";
CASISEL_PREG_RESET : string := "RESET0";
ACCSEL_PREG_CLK : string := "BYPASS";
ACCSEL_PREG_CE : string := "CE0";
ACCSEL_PREG_RESET : string := "RESET0";
C_PREG_CLK : string := "BYPASS";
C_PREG_CE : string := "CE0";
C_PREG_RESET : string := "RESET0";
FB_PREG_EN : string := "FALSE";
SOA_PREG_EN : string := "FALSE";
OREG_CLK : string := "BYPASS";
OREG_CE : string := "CE0";
OREG_RESET : string := "RESET0";
MULT_RESET_MODE : string := "SYNC";
PRE_LOAD : bit_vector := X"000000000000";
DYN_P_SEL : string := "FALSE";
P_SEL : bit := '0';
DYN_P_ADDSUB : string := "FALSE";
```

```
        P_ADDSUB : bit := '0';
        DYN_A_SEL : string := "FALSE";
        A_SEL : bit := '0';
        DYN_ADD_SUB_0 : string := "FALSE";
        ADD_SUB_0 : bit := '0';
        DYN_ADD_SUB_1 : string := "FALSE";
        ADD_SUB_1 : bit := '0';
        DYN_C_SEL : string := "FALSE";
        C_SEL : bit := '0';
        DYN_CASI_SEL : string := "FALSE";
        CASI_SEL : bit := '0';
        DYN_ACC_SE : string := "FALSE";
        ACC_SEL : bit := '0';
        MULT12X12_EN : string := "FALSE"
);
PORT (
    DOUT: out std_logic_vector(47 downto 0);
    CASO: out std_logic_vector(47 downto 0);
    SOA: out std_logic_vector(26 downto 0);
    A: in std_logic_vector(26 downto 0);
    B: in std_logic_vector(17 downto 0);
    C: in std_logic_vector(47 downto 0);
    D: in std_logic_vector(25 downto 0);
    SIA: in std_logic_vector(26 downto 0);
    CASI: in std_logic_vector(47 downto 0);
    ACCSEL: in std_logic;
    CASISEL: in std_logic;
    ASEL: in std_logic;
    PSEL: in std_logic;
    CSEL: in std_logic;
    ADDSUB: in std_logic_vector(1 downto 0);
```

```
            PADDSUB: in std_logic;
            CLK: in std_logic_vector(1 downto 0);
            CE: in std_logic_vector(1 downto 0);
            RESET: in std_logic_vector(1 downto 0)
        );
    end COMPONENT;
begin
    gw_gnd <= '0';

    A_i <= a[25] & a(25 downto 0);
    SIA_i <= gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd &
gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd &
gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd &
gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd &
gw_gnd;
    ADDSUB_i <= gw_gnd & gw_gnd;
    CLK_i <= gw_gnd & clk;
    CE_i <= gw_gnd & ce;
    RESET_i <= gw_gnd & reset;

    multalu27x18_inst: MULTALU27X18
        GENERIC MAP (
            AREG_CLK => "CLK0",
            AREG_CE => "CE0",
            AREG_RESET => "RESET0",
            BREG_CLK => "CLK0",
            BREG_CE => "CE0",
            BREG_RESET => "RESET0",
            DREG_CLK => "CLK0",
            DREG_CE => "CE0",
            DREG_RESET => "RESET0",
            C_IREG_CLK => "CLK0",
```

```
C_IREG_CE => "CE0",
C_IREG_RESET => "RESET0",
PSEL_IREG_CLK => "BYPASS",
PSEL_IREG_CE => "CE0",
PSEL_IREG_RESET => "RESET0",
PADDSUB_IREG_CLK => "BYPASS",
PADDSUB_IREG_CE => "CE0",
PADDSUB_IREG_RESET => "RESET0",
ADDSUB0_IREG_CLK => "BYPASS",
ADDSUB0_IREG_CE => "CE0",
ADDSUB0_IREG_RESET => "RESET0",
ADDSUB1_IREG_CLK => "BYPASS",
ADDSUB1_IREG_CE => "CE0",
ADDSUB1_IREG_RESET => "RESET0",
CSEL_IREG_CLK => "BYPASS",
CSEL_IREG_CE => "CE0",
CSEL_IREG_RESET => "RESET0",
CASISEL_IREG_CLK => "BYPASS",
CASISEL_IREG_CE => "CE0",
CASISEL_IREG_RESET => "RESET0",
ACCSEL_IREG_CLK => "BYPASS",
ACCSEL_IREG_CE => "CE0",
ACCSEL_IREG_RESET => "RESET0",
PREG_CLK => "BYPASS",
PREG_CE => "CE0",
PREG_RESET => "RESET0",
ADDSUB0_PREG_CLK => "BYPASS",
ADDSUB0_PREG_CE => "CE0",
ADDSUB0_PREG_RESET => "RESET0",
ADDSUB1_PREG_CLK => "BYPASS",
ADDSUB1_PREG_CE => "CE0",
```

```
ADDSUB1_PREG_RESET => "RESET0",
CSEL_PREG_CLK => "BYPASS",
CSEL_PREG_CE => "CE0",
CSEL_PREG_RESET => "RESET0",
CASISEL_PREG_CLK => "BYPASS",
CASISEL_PREG_CE => "CE0",
CASISEL_PREG_RESET => "RESET0",
ACCSEL_PREG_CLK => "BYPASS",
ACCSEL_PREG_CE => "CE0",
ACCSEL_PREG_RESET => "RESET0",
C_PREG_CLK => "CLK0",
C_PREG_CE => "CE0",
C_PREG_RESET => "RESET0",
FB_PREG_EN => "FALSE",
SOA_PREG_EN => "FALSE",
OREG_CLK => "CLK0",
OREG_CE => "CE0",
OREG_RESET => "RESET0",
MULT_RESET_MODE => "SYNC",
PRE_LOAD => X"000000000000",
DYN_P_SEL => "FALSE",
P_SEL => '0',
DYN_P_ADDSUB => "FALSE",
P_ADDSUB => '0',
DYN_A_SEL => "FALSE",
A_SEL => '0',
DYN_ADD_SUB_0 => "FALSE",
ADD_SUB_0 => '0',
DYN_ADD_SUB_1 => "FALSE",
ADD_SUB_1 => '0',
DYN_C_SEL => "FALSE",
```

```
        C_SEL => '1',
        DYN_CASI_SEL => "FALSE",
        CASI_SEL => '1',
        DYN_ACC_SEL => "FALSE",
        ACC_SEL => '0',
        MULT12X12_EN => "FALSE"
    )
    PORT MAP (
        DOUT => dout,
        CASO => caso,
        SOA => soa,
        A => A_i,
        B => b,
        C => c,
        D => d,
        SIA => SIA_i,
        CASI => casi,
        ACCSEL => gw_gnd,
        CASISEL => gw_gnd,
        ASEL => gw_gnd,
        PSEL => gw_gnd,
        CSEL => gw_gnd,
        ADDSUB => ADDSUB_i,
        PADDSUB => gw_gnd,
        CLK => CLK_i,
        CE => CE_i,
        RESET => RESET_i
       );
```

4.3 MULTADDALU

4.3.1 MULTADDALU12X12

MULTADDALU モードでは、2 つの 12x12 乗算器の出力の 48-bit ALU 演算が実現されます。対応するプリミティブは MULTADDALU12x12 です。

MULTADDALU12x12 には 4 つの演算モードがあります。

$$DOUT = A0 * B0 \pm A1 * B1$$

$$DOUT = DOUT \pm (A0 * B0 \pm A1 * B1)$$

$$DOUT = CASI \pm A0 * B0 \pm A1 * B1$$

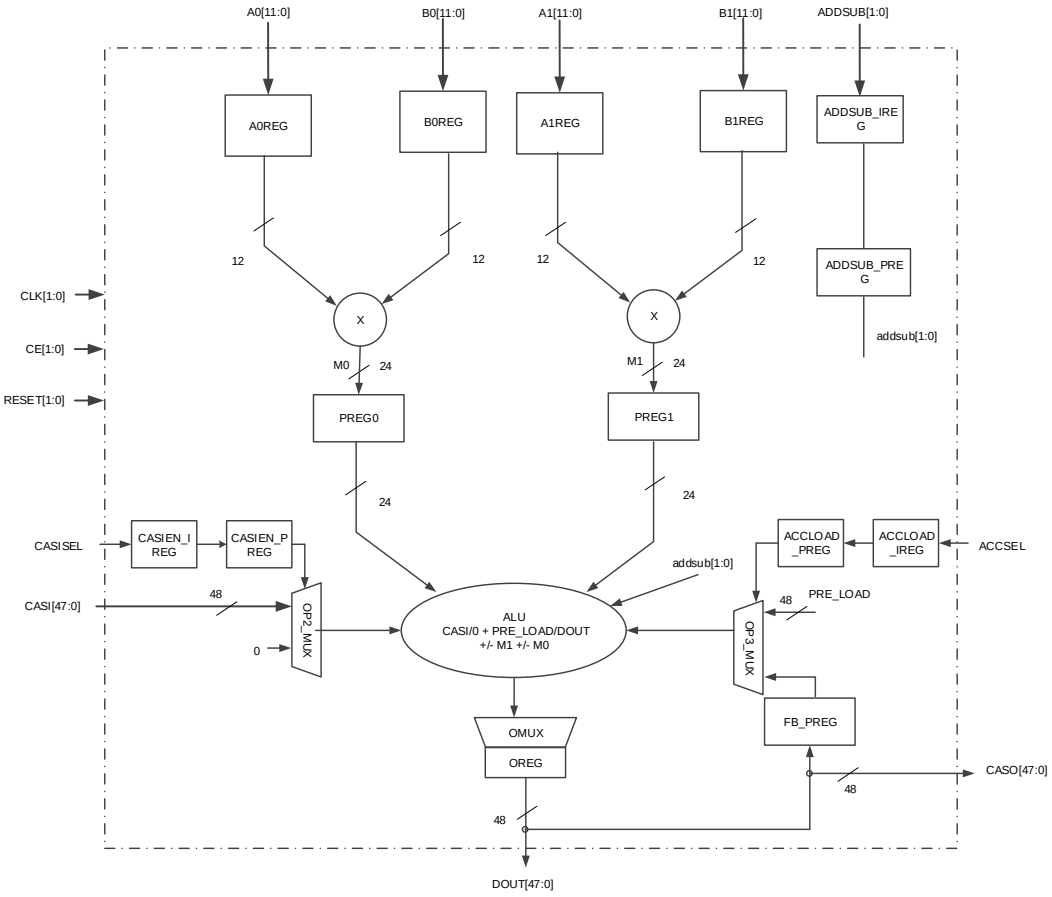
$$DOUT = CASI \pm (A0 * B0 \pm A1 * B1) + DOUT$$

プリミティブの紹介

MULTADDALU12x12（The Sum of Two 12x12 Multipliers with ALU）は、12 ビットの乗算の加算後の累積を実現する、ALU 機能付きの 12x12 乗算加算器です。

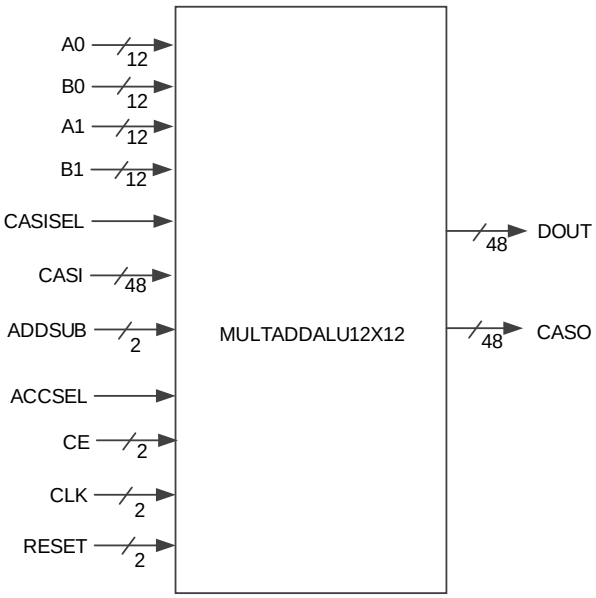
構造

図 4-7 MULTADDALU12X12 の構造



ポート図

図 4-8 MULTADDALU12X12 のポート図



ポートの説明

表 4-7 MULTADDALU12X12 のポートの説明

ポート	I/O	説明
A0[11:0]	入力	12-bit データ入力信号 A0
B0[11:0]	入力	12-bit データ入力信号 B0
A1[11:0]	入力	12-bit データ入力信号 A1
B1[11:0]	入力	12-bit データ入力信号 B1
CASI[47:0]	入力	前の DSP からの 48-bit カスケード接続入力信号
CASISEL	入力	48-bit ALU の CASI/0 の選択の制御信号
ADDSUB[1:0]	入力	加算/減算の動的制御信号
ACCSEL	入力	48-bit ALU の DOUT/PRE_LOAD の選択の制御信号
CLK[1:0]	入力	クロック入力信号
CE[1:0]	入力	クロックイネーブル信号
RESET[1:0]	入力	リセット入力信号
DOUT[53:0]	出力	データ出力信号
CASO[54:0]	出力	48-bit カスケード接続出力信号

パラメータの説明

表 4-8 MULTADDALU12X12 のパラメータの説明

パラメータ	範囲	デフォルト	説明
A0REG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 A0 レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
A0REG_CE	CE0，CE1	CE0	入力 A0 レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
A0REG_RESET	RESET0，RESET1	RESET0	入力 A0 レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
B0REG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 B0 レジスタのクロック制御信号 ● BYPASS: バイパスモード ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
B0REG_CE	CE0，CE1	CE0	入力 B0 レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
B0REG_RESET	RESET0，RESET1	RESET0	入力 B0 レジスタのリセット制御信号

パラメータ	範囲	デフォルト	説明
			● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
A1REG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 A1 レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
A1REG_CE	CE0，CE1	CE0	入力 A1 レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
A1REG_RESET	RESET0，RESET1	RESET0	入力 A1 レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
B1REG_CLK	BYPASS，CLK0，CLK1	CLK0	入力 B1 レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
B1REG_CE	CE0，CE1	CE0	入力 B1 レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。

パラメータ	範囲	デフォルト	説明
B1REG_RESET	RESET0，RESET1	RESET0	入力 B1 レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
ADDSUB0_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ADDSUB[0]レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ADDSUB0_IREG_CE	CE0，CE1	CE0	入力 ADDSUB[0]レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
ADDSUB0_IREG_RESET	RESET0，RESET1	RESET0	入力 ADDSUB[0]レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
ADDSUB1_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ADDSUB[1]レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ADDSUB1_IREG_CE	CE0，CE1	CE0	入力 ADDSUB[1]レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制

パラメータ	範囲	デフォルト	説明
			御信号は CE[1]から供給されます。
ADDSUB1_IREG_RESET	RESET0，RESET1	RESET0	入力 ADDSUB[1]レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
ADDSUB0_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ADDSUB[0] Pipeline レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ADDSUB0_PREG_CE	CE0，CE1	CE0	入力 ADDSUB[0] Pipeline レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
ADDSUB0_PREG_RESET	RESET0.RESET1	RESET0	入力 ADDSUB[0] Pipeline レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
ADDSUB1_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ADDSUB[1] Pipeline レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ADDSUB1_PRE	CE0，CE1	CE0	入力 ADDSUB[1] Pipeline レジスタのクロッ

パラメータ	範囲	デフォルト	説明
G_CE			クイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
ADDSUB1_PREG_RESET	RESET0，RESET1	RESET0	入力 ADDSUB[1] Pipeline レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
CASISEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 CASISEL レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
CASISEL_IREG_CE	CE0，CE1	CE0	入力 CASISEL レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
CASISEL_IREG_RESET	RESET0，RESET1	RESET0	入力 CASISEL レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
CASISEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 CASISEL Pipeline レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロ

パラメータ	範囲	デフォルト	説明
			ック制御信号は CLK[1]から供給されます。
CASISEL_PREG_CE	CE0，CE1	CE0	入力 CASISEL Pipeline レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
CASISEL_PREG_RESET	RESET0，RESET1	RESET0	入力 CASISEL Pipeline レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
ACCSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ACCSEL レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ACCSEL_IREG_CE	CE0，CE1	CE0	入力 ACCSEL レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
ACCSEL_IREG_RESET	RESET0，RESET1	RESET0	入力 ACCSEL レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
ACCSEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	入力 ACCSEL Pipeline レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロ

パラメータ	範囲	デフォルト	説明
			ック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
ACCSEL_PREG_CE	CE0，CE1	CE0	入力 ACCSEL Pipeline レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
ACCSEL_PREG_RESET	RESET0，RESET1	RESET0	入力 ACCSEL Pipeline レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
PREG0_CLK	BYPASS，CLK0，CLK1	BYPASS	Mult0 Pipeline レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
PREG0_CE	CE0，CE1	CE0	Mult0 Pipeline レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
PREG0_RESET	RESET0，RESET1	RESET0	Mult0 Pipeline レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
PREG1_CLK	BYPASS，CLK0，	BYPASS	Mult1 Pipeline レジスタのクロック制御信号

パラメータ	範囲	デフォルト	説明
	CLK1		● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
PREG1_CE	CE0，CE1	CE0	Mult1 Pipeline レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
PREG1_RESET	RESET0，RESET1	RESET0	Mult1 Pipeline レジスタのリセット制御信号 ● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
FB_PREG_EN	FALSE，TRUE	FALSE	フィードバック出力 Pipeline レジスタのモード制御パラメータ ● FALSE: バイパスモード。 ● TRUE: レジスタモード。制御信号 clk/ce/reset は、OREG と同じです。
OREG_CLK	BYPASS，CLK0，CLK1	BYPASS	出力レジスタのクロック制御信号 ● BYPASS: バイパスモード。 ● CLK0: レジスタモード。レジスタのクロック制御信号は CLK[0]から供給されます。 ● CLK1: レジスタモード。レジスタのクロック制御信号は CLK[1]から供給されます。
OREG_CE	CE0，CE1	CE0	出力レジスタのクロックイネーブル制御信号 ● CE0: レジスタのクロックイネーブル制御信号は CE[0]から供給されます。 ● CE1: レジスタのクロックイネーブル制御信号は CE[1]から供給されます。
OREG_RESET	RESET0，RESET1	RESET0	出力レジスタのリセット制御信号

パラメータ	範囲	デフォルト	説明
			● RESET0: レジスタのリセット制御信号は RESET[0]から供給されます。 ● RESET1: レジスタのリセット制御信号は RESET[1]から供給されます。
MULT_RESET_MODE	SYNC，ASYNC	SYNC	同期または非同期
PRE_LOAD	48bits value	48’h0	PRE_LOAD の初期値
ADD_SUB_0	1’b0，1’b1	1’b0	M0/0 の加算または減算の選択の静的制御 ● 1’b0: 加算 ● 1’b1: 減算
DYN_ADD_SUB_0	FALSE，TRUE	FALSE	M0/0 の加算または減算の選択の動的制御 ● FALSE: ADD_SUB_0 により M0/0 の加算または減算の選択を静的に制御します。 ● TRUE: 入力 ADDSUB[0]により M0/0 の加算または減算の選択を動的に制御します。
ADD_SUB_1	1’b0，1’b1	1’b0	M1/0 の加算または減算の選択の静的制御 ● 1’b0: 加算 ● 1’b1: 減算
DYN_ADD_SUB_1	FALSE，TRUE	FALSE	M1/0 の加算または減算の選択の動的制御 ● FALSE: ADD_SUB_1 により M1/0 の加算または減算の選択を静的に制御します。 ● TRUE: 入力 ADDSUB[1]により M1/0 の加算または減算の選択を動的に制御します。
CASI_SEL	1’b0，1’b1	1’b0	CASI/0 の選択の静的制御 ● 1’b0: 0 ● 1’b1: CASI
DYN_CASI_SEL	FALSE，TRUE	FALSE	CASI/0 の選択の動的制御 ● FALSE: CASI_SEL により CASI/0 の選択を静的に制御します。 ● TRUE: 入力 CASISEL により CASI/0 の選択を動的に制御します。
ACC_SEL	1’b0，1’b1	1’b0	PRE_LOAD/DOUT の選択の静的制御 ● 1’b0：PRE_LOAD ● 1’b1：フィードバックされる DOUT

パラメータ	範囲	デフォルト	説明
DYN_ACC_SEL	FALSE，TRUE	FALSE	PRE_LOAD/DOUT の選択の動的制御 ● FALSE: ACC_SEL により PRE_LOAD/DOUT の選択を静的に制御します。 ● TRUE: 入力 ACCSEL により PRE_LOAD/DOUT の選択を動的に制御します。

プリミティブのインスタンス化

プリミティブを直接インスタンス化するか、IP Core Generator で生成できます。詳しくは、5 IP の呼び出しを参照してください。

Verilog でのインスタンス化：

```
MULTADDALU12X12 multaddalu12x12_inst (
    .DOUT(dout),
    .CASO(caso),
    .A0(a0),
    .B0(b0),
    .A1(a1),
    .B1(b1),
    .CASI(casi),
    .ACCSEL(gw_gnd),
    .CASISEL(gw_gnd),
    .ADDSUB({gw_gnd,gw_gnd}),
    .CLK({gw_gnd,clk}),
    .CE({gw_gnd,ce}),
    .RESET({gw_gnd,reset})
);

defparam multaddalu12x12_inst.A0REG_CLK = "CLK0";
defparam multaddalu12x12_inst.A0REG_CE = "CE0";
defparam multaddalu12x12_inst.A0REG_RESET = "RESET0";
```

```
defparam multaddalu12x12_inst.A1REG_CLK = "CLK0";
defparam multaddalu12x12_inst.A1REG_CE = "CE0";
defparam multaddalu12x12_inst.A1REG_RESET = "RESET0";
defparam multaddalu12x12_inst.B0REG_CLK = "CLK0";
defparam multaddalu12x12_inst.B0REG_CE = "CE0";
defparam multaddalu12x12_inst.B0REG_RESET = "RESET0";
defparam multaddalu12x12_inst.B1REG_CLK = "CLK0";
defparam multaddalu12x12_inst.B1REG_CE = "CE0";
defparam multaddalu12x12_inst.B1REG_RESET = "RESET0";
defparam multaddalu12x12_inst.ACCSEL_IREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ACCSEL_IREG_CE = "CE0";
defparam multaddalu12x12_inst.ACCSEL_IREG_RESET = "RESET0";
defparam multaddalu12x12_inst.CASISEL_IREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.CASISEL_IREG_CE = "CE0";
defparam multaddalu12x12_inst.CASISEL_IREG_RESET = "RESET0";
defparam multaddalu12x12_inst.ADDSUB0_IREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ADDSUB0_IREG_CE = "CE0";
defparam multaddalu12x12_inst.ADDSUB0_IREG_RESET = "RESET0";
defparam multaddalu12x12_inst.ADDSUB1_IREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ADDSUB1_IREG_CE = "CE0";
defparam multaddalu12x12_inst.ADDSUB1_IREG_RESET = "RESET0";
defparam multaddalu12x12_inst.PREG0_CLK = "BYPASS";
defparam multaddalu12x12_inst.PREG0_CE = "CE0";
defparam multaddalu12x12_inst.PREG0_RESET = "RESET0";
defparam multaddalu12x12_inst.PREG1_CLK = "BYPASS";
defparam multaddalu12x12_inst.PREG1_CE = "CE0";
defparam multaddalu12x12_inst.PREG1_RESET = "RESET0";
defparam multaddalu12x12_inst.FB_PREG_EN = "FALSE";
defparam multaddalu12x12_inst.ACCSEL_PREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ACCSEL_PREG_CE = "CE0";
defparam multaddalu12x12_inst.ACCSEL_PREG_RESET = "RESET0";
```

```
defparam multaddalu12x12_inst.CASISEL_PREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.CASISEL_PREG_CE = "CE0";
defparam multaddalu12x12_inst.CASISEL_PREG_RESET = "RESET0";
defparam multaddalu12x12_inst.ADDSUB0_PREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ADDSUB0_PREG_CE = "CE0";
defparam multaddalu12x12_inst.ADDSUB0_PREG_RESET =
"RESET0";
defparam multaddalu12x12_inst.ADDSUB1_PREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ADDSUB1_PREG_CE = "CE0";
defparam multaddalu12x12_inst.ADDSUB1_PREG_RESET =
"RESET0";
defparam multaddalu12x12_inst.OREG_CLK = "CLK0";
defparam multaddalu12x12_inst.OREG_CE = "CE0";
defparam multaddalu12x12_inst.OREG_RESET = "RESET0";
defparam multaddalu12x12_inst.MULT_RESET_MODE = "SYNC";
defparam multaddalu12x12_inst.PRE_LOAD = 48'h000000000000;
defparam multaddalu12x12_inst.DYN_ADD_SUB_0 = "FALSE";
defparam multaddalu12x12_inst.ADD_SUB_0 = 1'b0;
defparam multaddalu12x12_inst.DYN_ADD_SUB_1 = "FALSE";
defparam multaddalu12x12_inst.ADD_SUB_1 = 1'b0;
defparam multaddalu12x12_inst.DYN_CASI_SEL = "FALSE";
defparam multaddalu12x12_inst.CASI_SEL = 1'b1;
defparam multaddalu12x12_inst.DYN_ACC_SEL = "FALSE";
defparam multaddalu12x12_inst.ACC_SEL = 1'b0;
```

VHDL でのインスタンス化：

```
COMPONENT MULTADDALU12X12
        GENERIC (
            A0REG_CLK : string := "BYPASS";
            A0REG_CE : string := "CE0";
            A0REG_RESET : string := "RESET0";
            A1REG_CLK : string := "BYPASS";
```

```
A1REG_CE : string := "CE0";
A1REG_RESET : string := "RESET0";
B0REG_CLK : string := "BYPASS";
B0REG_E : string := "CE0";
B0REG_RESET : string := "RESET0";
B1REG_CLK : string := "BYPASS";
B1REG_CE : string := "CE0";
B1REG_RESET : string := "RESET0";
ACCSEL_IREG_CLK : string := "BYPASS";
ACCSEL_IREG_CE : string := "CE0";
ACCSEL_IREG_RESET : string := "RESET0";
CASISEL_IREG_CLK : string := "BYPASS";
CASISEL_IREG_CE : string := "CE0";
CASISEL_IREG_RESET : string := "RESET0";
ADDSUB0_IREG_CLK : string := "BYPASS";
ADDSUB0_IREG_CE : string := "CE0";
ADDSUB0_IREG_RESET : string := "RESET0";
ADDSUB1_IREG_CLK : string := "BYPASS";
ADDSUB1_IREG_CE : string := "CE0";
ADDSUB1_IREG_RESET : string := "RESET0";
PREG0_CLK : string := "BYPASS";
PREG0_CE : string := "CE0";
PREG0_RESET : string := "RESET0";
PREG1_CLK : string := "BYPASS";
PREG1_CE : string := "CE0";
PREG1_RESET : string := "RESET0";
FB_PREG_EN : string := "FALSE";
ACCSEL_PREG_CLK : string := "BYPASS";
ACCSEL_PREG_CE : string := "CE0";
ACCSEL_PREG_RESET : string := "RESET0";
CASISEL_PREG_CLK : string := "BYPASS";
```

```
        CASISEL_PREG_CE : string := "CE0";
        CASISEL_PREG_RESET : string := "RESET0";
        ADDSUB0_PREG_CLK : string := "BYPASS";
        ADDSUB0_PREG_CE : string := "CE0";
        ADDSUB0_PREG_RESET : string := "RESET0";
        ADDSUB1_PREG_CLK : string := "BYPASS";
        ADDSUB1_PREG_CE : string := "CE0";
        ADDSUB1_PREG_RESET : string := "RESET0";
        OREG_CLK : string := "BYPASS";
        OREG_CE : string := "CE0";
        OREG_RESET : string := "RESET0";
        MULT_RESET_MODE : string := "SYNC";
        PRE_LOAD : bit_vector := X"000000000000";
        DYN_ADD_SUB_0 : string := "FALSE";
        ADD_SUB_0 : bit := '0';
        DYN_ADD_SUB_1 : string := "FALSE";
        ADD_SUB_1 : bit := '0';
        DYN_CASI_SEL : string := "FALSE";
        CASI_SEL : bit := '0';
        DYN_ACC_SE : string := "FALSE";
        ACC_SEL : bit := '0';
    );
    PORT (
        DOUT: out std_logic_vector(47 downto 0);
        CASO: out std_logic_vector(47 downto 0);
        A0: in std_logic_vector(11 downto 0);
        B0: in std_logic_vector(11 downto 0);
        A1: in std_logic_vector(11 downto 0);
        B1: in std_logic_vector(11 downto 0);
        CASI: in std_logic_vector(47 downto 0);
        ACCSEL: in std_logic;
```

```
            CASISEL: in std_logic;
            ADDSUB: in std_logic_vector(1 downto 0);
            CLK: in std_logic_vector(1 downto 0);
            CE: in std_logic_vector(1 downto 0);
            RESET: in std_logic_vector(1 downto 0)
        );
    end COMPONENT;
begin
    gw_gnd <= '0';

    ADDSUB_i <= gw_gnd & gw_gnd;
    CLK_i <= gw_gnd & clk;
    CE_i <= gw_gnd & ce;
    RESET_i <= gw_gnd & reset;

    multaddalu12x12_inst: MULTADDALU12X12
        GENERIC MAP (
            A0REG_CLK => "CLK0",
            A0REG_CE => "CE0",
            A0REG_RESET => "RESET0",
            A1REG_CLK => "CLK0",
            A1REG_CE => "CE0",
            A1REG_RESET => "RESET0",
            B0REG_CLK => "CLK0",
            B0REG_CE => "CE0",
            B0REG_RESET => "RESET0",
            B1REG_CLK => "CLK0",
            B1REG_CE => "CE0",
            B1REG_RESET => "RESET0",
            ACCSEL_IREG_CLK => "BYPASS",
            ACCSEL_IREG_CE => "CE0",
```

```
ACCSEL_IREG_RESET => "RESET0",
CASISEL_IREG_CLK => "BYPASS",
CASISEL_IREG_CE => "CE0",
CASISEL_IREG_RESET => "RESET0",
ADDSUB0_IREG_CLK => "BYPASS",
ADDSUB0_IREG_CE => "CE0",
ADDSUB0_IREG_RESET => "RESET0",
ADDSUB1_IREG_CLK => "BYPASS",
ADDSUB1_IREG_CE => "CE0",
ADDSUB1_IREG_RESET => "RESET0",
PREG0_CLK => "BYPASS",
PREG0_CE => "CE0",
PREG0_RESET => "RESET0",
PREG1_CLK => "BYPASS",
PREG1_CE => "CE0",
PREG1_RESET => "RESET0",
FB_PREG_EN => "FALSE",
ACCSEL_PREG_CLK => "BYPASS",
ACCSEL_PREG_CE => "CE0",
ACCSEL_PREG_RESET => "RESET0",
CASISEL_PREG_CLK => "BYPASS",
CASISEL_PREG_CE => "CE0",
CASISEL_PREG_RESET => "RESET0",
ADDSUB0_PREG_CLK => "BYPASS",
ADDSUB0_PREG_CE => "CE0",
ADDSUB0_PREG_RESET => "RESET0",
ADDSUB1_PREG_CLK => "BYPASS",
ADDSUB1_PREG_CE => "CE0",
ADDSUB1_PREG_RESET => "RESET0",
OREG_CLK => "CLK0",
OREG_CE => "CE0",
```

```
        OREG_RESET => "RESET0",
        MULT_RESET_MODE => "SYNC"
        PRE_LOAD => X"000000000000",
        DYN_ADD_SUB_0 => "FALSE",
        ADD_SUB_0 => '0',
        DYN_ADD_SUB_1 => "FALSE",
        ADD_SUB_1 => '0',
        DYN_CASI_SEL => "FALSE",
        CASI_SEL => '1',
        DYN_ACC_SEL => "FALSE",
        ACC_SEL => '0',
    )
    PORT MAP (
        DOUT => dout,
        CASO => caso,
        A0 => a0,
        B0 => b0,
        A1 => a1,
        B1 => b1,
        CASI => casi,
        ACCSEL => gw_gnd,
        CASISEL => gw_gnd,
        ADDSUB => ADDSUB_i,
        CLK => CLK_i,
        CE => CE_i,
        RESET => RESET_i
    );
```

5 IP の呼び出し

IP Core Generator は、3 種類の DSP プリミティブ(MULT、MULTALU、MULTADDALU)の生成をサポートします。

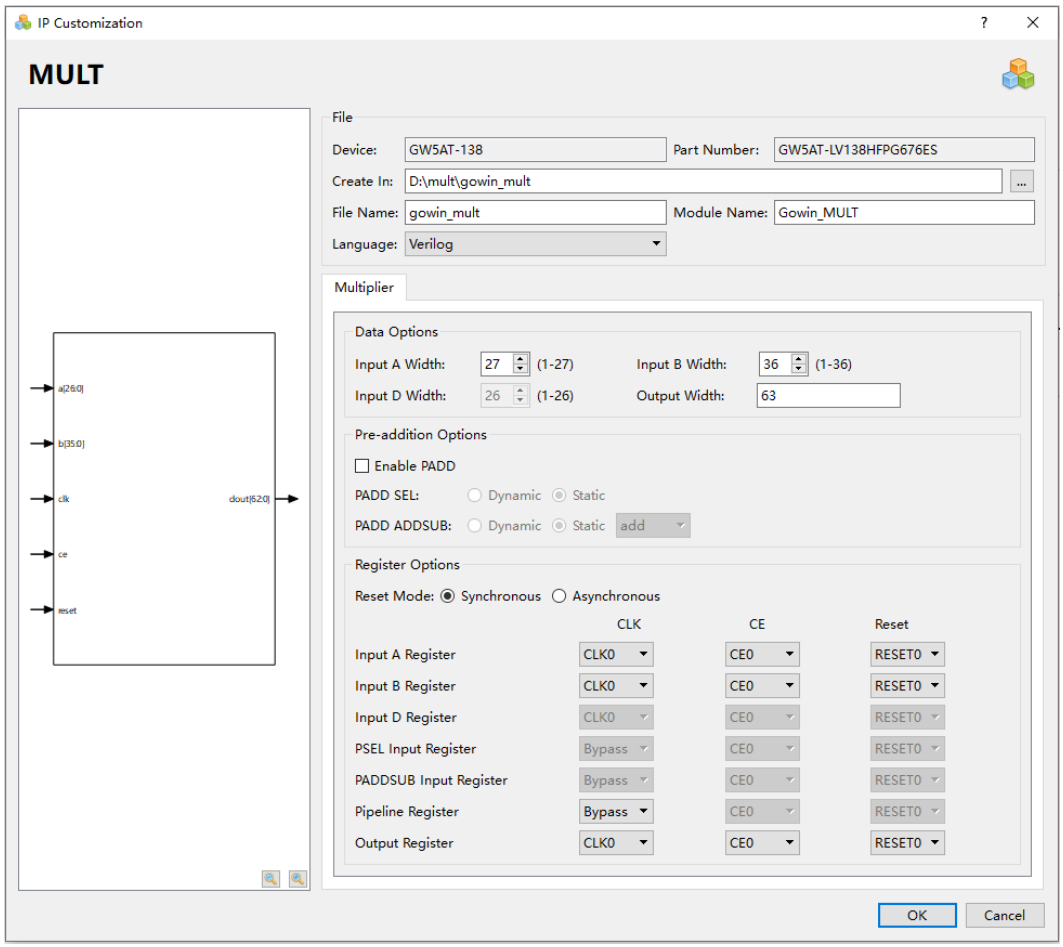
5.1 MULT

MULT は、乗算、乗算に基づく前置加算と前置減算を実現します。IP Core Generator のインターフェースで“MULT”をクリックすると、右側に MULT の概要が表示されます。

IP の構成

IP Core Generator インターフェースで MULT をダブルクリックすると、MULT の“IP Customization”ウィンドウがポップアップします(図 5-1)。このウィンドウには“File”構成タブ、“Multiplier”構成タブ、およびポート図があります。

図 5-1 MULT IP の構成ウィンドウ



1. File 構成タブは、生成される IP ファイルの構成に使用されます。

- Device：対象デバイス。
- Part Number：部品番号。
- Language：IP を実現するハードウェア記述言語。右側のドロップダウン・リストからターゲット言語(Verilog または VHDL)を選択します。
- Module Name：生成される IP ファイルのモジュール名。右側のテキストボックスで編集できます。Module Name をプリミティブ名と同じにすることはできません。同じである場合、エラーが報告されます。
- File Name：生成される IP ファイルのファイル名。右側のテキストボックスで再編集できます。
- Create In：生成される IP ファイルのパス。右側のテキストボックスでパスを直接編集するか、テキストボックスの右側にある選択ボタンを使用してパスを選択できます。

2. Multiplier 構成タブは IP のカスタマイズに使用されます(図 5-1)。

- Data Options：データオプションを構成します。
  - 入力ポート(Input A Width)の最大データ幅は 27 ビットです。
  - 入力ポート(Input B Width)の最大データ幅は 36 ビットです。
  - 入力ポート(Input D Width)の最大データ幅は 26 ビットです。
  - 出力ポートのデータ幅(Output Width)は、入力データ幅に従って自動的に調整されるため、ユーザーによる設定を必要としません。インスタンス化の際にデータ幅に従って MULT12X12 または MULT27X36 が生成されます。
- Pre-addition Options：前置加算のオプションを構成します。
  - “Enable PADD”：PADD をイネーブルします。
  - “PADD SEL”：PADD のイネーブルを動的にまたは静的に制御するかを構成します。
  - “PADD ADDSUB”：PADD の前置加減算を動的にまたは静的に制御するかを構成します。
  - 出力ポートのデータ幅(Output Width)は、入力データ幅に従って自動的に調整されるため、ユーザーによる設定を必要としません。インスタンス化の際にデータ幅に従って MULT12X12 または MULT27X36 が生成されます。
- Register Options：レジスタの動作モードを構成します。
  - “Reset Mode”：MULT のリセットモードを構成します。同期モード“Synchronous”と非同期モード“Asynchronous”がサポートされます。
  - “Input A Register”：Input A Register の CLK、CE、および Reset 信号のソースを構成します。CLK には Bypass, CLK0, および CLK1 という項目があり、CE には CE0 と CE1 という項目があり、Reset には RESET0 と RESET1 という項目があります。
  - “Input B Register”：Input B Register の CLK、CE、および Reset 信号のソースを構成します。構成項目は上記と同様です。
  - “Input D Register”：Input D Register の CLK、CE、および Reset 信号のソースを構成します。構成項目は上記と同様です。
  - “PSEL Input Register”：PSEL Input Register の CLK、CE、および Reset 信号のソースを構成します。構成項目は上記と

同様です。

- “PADDSUB Input Register”：PADDSUB Input Register の CLK、CE、および Reset 信号のソースを構成します。構成項目は上記と同様です。
- “Pipeline Register”：Pipeline Register の CLK、CE、および Reset 信号のソースを構成します。構成項目は上記と同様です。
- “Output Register”：Output Register の CLK、CE、および Reset 信号のソースを構成します。構成項目は上記と同様です。

3. ポート図：現在の IP Core の構成結果を表示します。入力・出力ポートおよびそのビット幅は Options 構成に従ってリアルタイムで更新されます(図 5-1)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_mult.v”は完全な verilog モジュールです。
- “gowin_mult_tmp.v”は IP のテンプレートファイルです。
- “gowin_mult.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

5.2 MULTALU

MULTALU は、

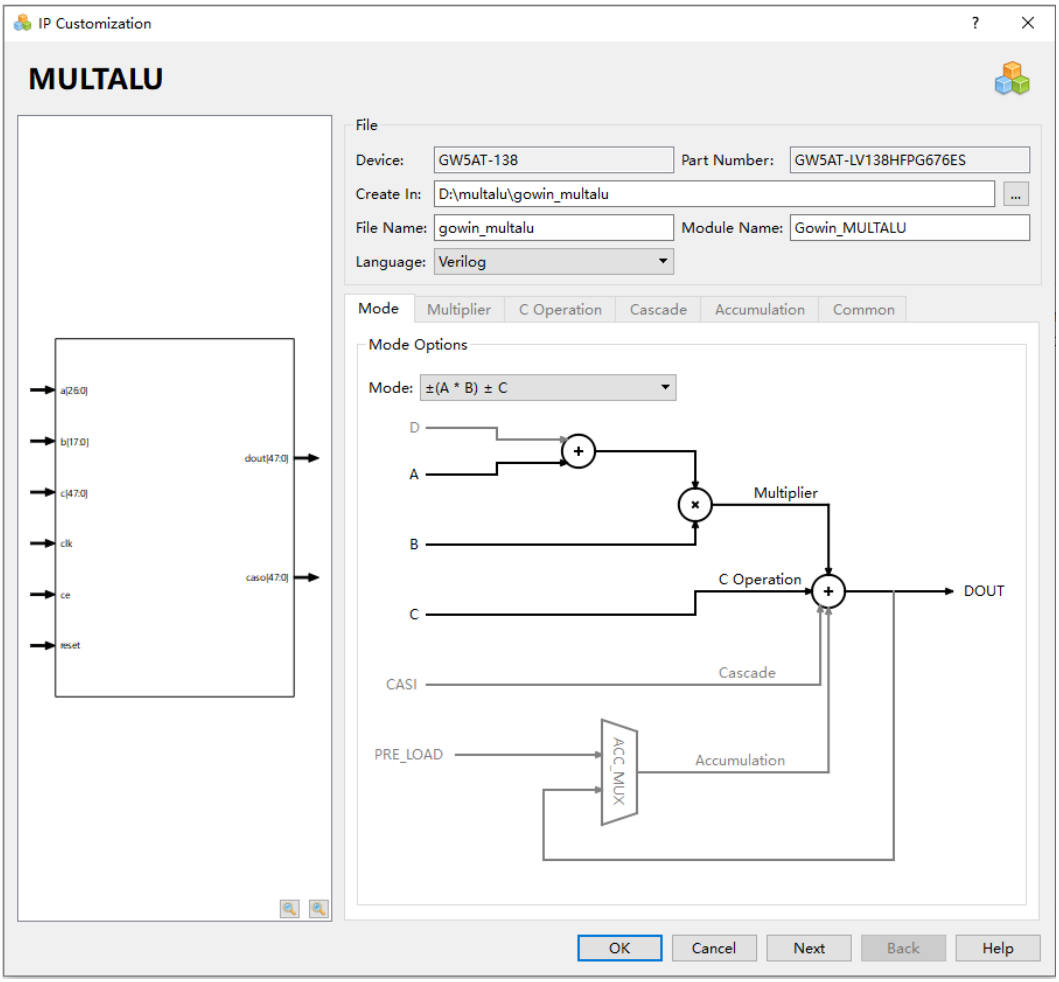
乗算、乗算加算、累積、乗算累積、乗算/乗算加算/累積/乗算累積に基づくシフト、乗算/乗算加算/累積/乗算累積に基づくカスケード、および乗算/乗算加算/累積/乗算累積に基づく前置加算と前置減算を実装します。IP Core Generator のインターフェースで MULTALU をクリックすると、右側に MULTALU の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“MULTALU”をダブルクリックすると、“IP Customization”ウィンドウがポップアップします。このウィンドウには、“File”構成タブ、“Mode”、“Multiplier”、“C Operation”、“Cascade”、“Accumulation”、“Common”構成タブがあります(図

5-2)。

図 5-2 MULTALU IP の構成ウィンドウ



1. File 構成タブは、生成される IP ファイルの構成に使用されます。MULTALU の File 構成タブの使用は MULT モジュールと同様です。5.1 MULT を参照してください。
2. Mode 構成タブ：MULTALU27X18 の演算モードを構成します。
   - Mode Options：MULTALU27X18 の演算モードを構成します。そのオプションは次のとおりです：
     - ±(A*B)
     - ±((A±D) *B)
     - ±(A*B) ± C
     - ±((A±D) *B) ± C
     - Accum ±(A*B)
     - Accum ±((A±D) *B)

- Accum ±(A*B) ± C
- Accum ±((A±D) *B) ± C
- CASI ±(A*B)
- CASI ±((A±D) *B)
- CASI ±(A*B) ± C
- CASI ±((A±D) *B) ± C
- Accum ±(A*B) +CASI
- Accum ±((A±D) *B) +CASI
- Accum ±(A*B) ± C +CASI
- Accum ±((A±D) *B) ± C +CASI

3. Multiplier 構成タブ：乗算器を構成します。Data Options、Pre-addition Options、ASEL Option、ADDSUB0 Option、Shift Option、Register Options があります。
   - MULTALU の Data Options、Pre-addition Options、Register Options の構成方法は MULT モジュールと同様です。5.1 MULT を参照してください。
   - ASEL Option：A または SIA ソースの選択の制御モードを構成します。
     - 動的制御“Dynamic”と静的制御“Static”がサポートされます。
     - 動的制御の場合、ASEL 入力ポートがイネーブルされます。
     - 静的制御の場合、“Parallel”（A を選択）または“Shift”（SIA を選択）として構成できます。
   - ADDSUB0 Option：M0/0 の加算または減算の選択の制御モードを構成します。
     - 動的制御“Dynamic”と静的制御“Static”がサポートされます。
     - 動的制御の場合、ADDSUB0 入力ポートがイネーブルされます。
     - 静的制御の場合、“add”（加算を選択）または“sub1”（減算を選択）として構成できます。
   - Shift Option：shift out 機能をイネーブルします。
4. C Opreation 構成タブ：入力 C を構成します。Data Options、CSEL

Option、ADDSUB1 Option、Register Options があります。

- MULTALU の Data Options と Register Options の構成方法は MULT モジュールと同様です。5.1 MULT を参照してください。
- CSEL Option：C または 0 ソースの選択の制御モードを構成します。
  - 動的制御“Dynamic”と静的制御“Static”がサポートされます。
- ADDSUB1 Option：M1/C/0 の加算または減算の選択の制御モードを構成します。
  - 動的制御“Dynamic”と静的制御“Static”がサポートされます。
  - 動的制御の場合、ADDSUB1 入力ポートがイネーブルされます。
  - 静的制御の場合、“add”（加算を選択）または“sub1”（減算を選択）として構成できます。

5. Cascade 構成タブ：カスケード入力 CASI を構成します。CASISEL Option と Register Options があります。
   - MULTALU の Register Options の構成方法は MULT モジュールと同様です。5.1 MULT を参照してください。
   - CASISEL Option：CASI または 0 ソースの選択の制御モードを構成します。
     - 動的制御“Dynamic”と静的制御“Static”がサポートされます。
     - 動的制御の場合、CASISEL 入力ポートがイネーブルされます。
6. Accumulation 構成タブ：ACCSEL と PRE_LOAD を構成します。ACCSEL Option、Initialization Option、および Register Options があります。
   - MULTALU の Register Options の構成方法は MULT モジュールと同様です。5.1 MULT を参照してください。
   - ACCSEL Option：PRE_LOAD または出力フィードバックの選択の制御モードを構成します。
     - 動的制御“Dynamic”と静的制御“Static”がサポートされます。
     - 動的制御の場合、ACCSEL 入力ポートがイネーブルされま

す。

- 静的制御の場合、“PRE_LOAD”（PRE_LOAD を選択）または“DOUT”（出力フィードバックを選択）として構成できます。

- Initialization Option：PRE_LOAD の初期値を設定します。
  - Preload Value の範囲は 48’h000000000000~48’hFFFFFFFFFFFF です。

7. Common 構成タブ：出力とリセットモードを構成します。Data Options と Register Options があります。
   - MULTALU の Data Options と Register Options の構成方法は MULT モジュールと同様です。5.1 MULT を参照してください。
8. ポート図：現在の IP Core の構成結果を表示します。入力・出力ポートおよびそのビット幅は Options 構成に従ってリアルタイムで更新されます(図 5-2)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_multalu.v”は完全な verilog モジュールです。
- “gowin_multalu_tmp.v”は IP のテンプレートファイルです。
- “gowin_multalu.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。

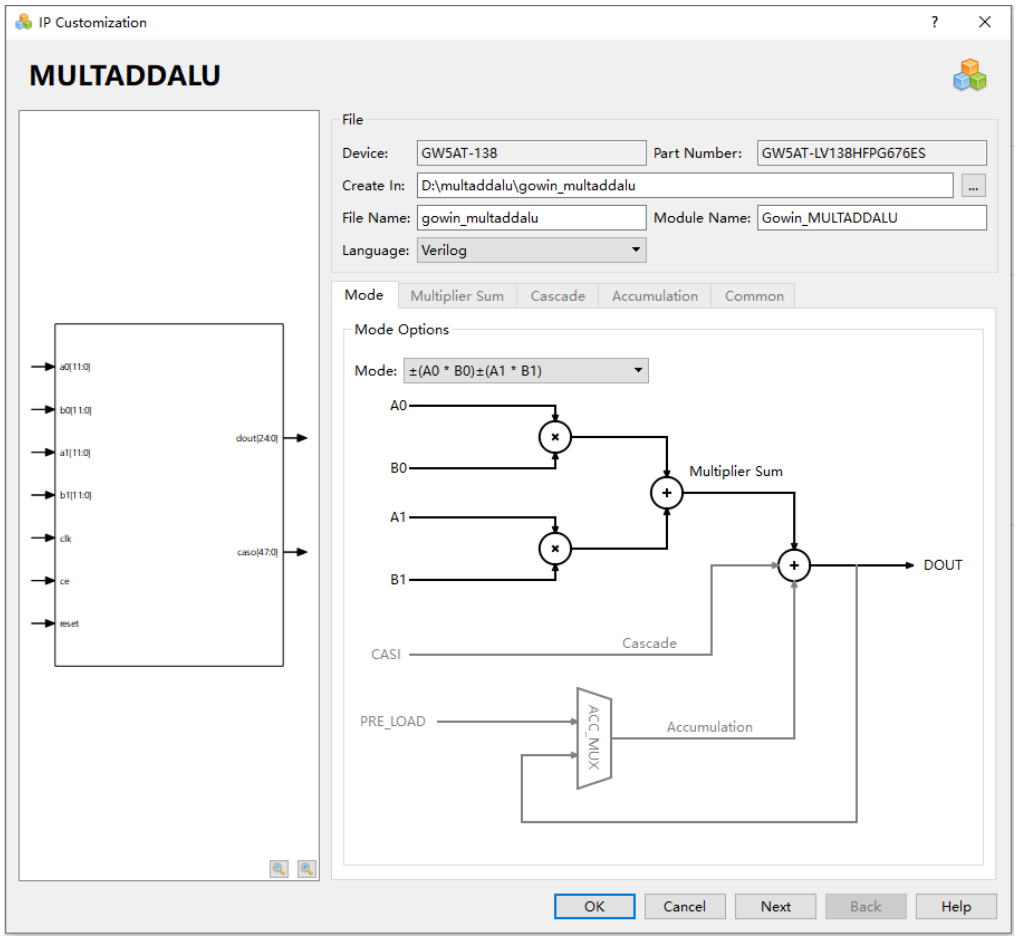
5.3 MULTADDALU

MULTADDALU は、乗算の加算後の累積を実現します。IP Core Generator のインターフェースで“MULTADDALU”をクリックすると、右側に MULTADDALU の概要が表示されます。

IP の構成

IP Core Generator インターフェースで“MULTADDALU”をダブルクリックすると、MULTADDALU の“IP Customization”ウィンドウがポップアップします。このウィンドウには“File”構成タブ、“Options”構成タブ、およびポート図があります(図 5-3)。

図 5-3 MULTADDALU IP の構成ウィンドウ



1. File 構成タブは、生成される IP ファイルの構成に使用されます。MULTADDALU の File 構成タブの使用は MULT モジュールと同様です。5.1 MULT を参照してください。

2. Mode Option：MULTADDALU の演算モードを構成します。そのオプションは次のとおりです：

   - ± (A0 * B0) ± (A1 * B1)
   - CASI ± (A0 * B0) ± (A1 * B1)
   - Accum ± (A0 * B0) ± (A1 * B1)
   - Accum ± CASI ± (A0 * B0) ± (A1 * B1)

3. Multiplier Sum，Cascade, Accumulation, Common 構成タブでのパラメータの構成は MULTALU と同様です。5.2 MULTALU を参照してください。

4. ポート図：現在の IP Core の構成結果を表示します。入力・出力ポートおよびそのビット幅は Options 構成に従ってリアルタイムで更新されます(図 5-3)。

生成されるファイル

IP の構成が完了したら、“File Name”によって命名された 3 つのファイルが生成されます：

- “gowin_multaddalu.v”は完全な verilog モジュールです。
- “gowin_multaddalu_tmp.v”は IP のテンプレートファイルです。
- “gowin_multaddalu.ipc”は IP の構成ファイルです。

注記：

VHDL が設計の言語として選択されている場合、生成される最初の 2 つのファイル名のサフィックスは.vhd になります。


GOWIN高云
智慧逻辑 定制未来