Gowin DSP
ユーザーガイド

UG287-1.3.3J, 2023-01-05

著作権について(2023)
著作権に関する全ての権利は、Guangdong Gowin Semiconductor Corporation に留保されています。

GOWIN高云、Gowin、及び GOWINSEMI は、当社により、中国、米国特許商標庁、及びその他の国において登録されています。商標又はサービスマークとして特定されたその他全ての文字やロゴは、それぞれの権利者に帰属しています。何れの団体及び個人も、当社の書面による許可を得ず、本文書の内容の一部もしくは全部を、いかなる視聴覚的、電子的、機械的、複写、録音等の手段によりもしくは形式により、伝搬又は複製をしてはなりません。

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
2016/05/16	1.05J	初版。
2016/07/04	1.06J	PADD18 のブロック図を変更。
2016/07/11	1.07J	図面を更新。
2016/08/16	1.08J	GW2A-18 デバイスの乗算器数を変更。
2016/11/08	1.09J	乗算器のブロック図を変更。
2017/10/09	1.10J	新らしいプリミティブに基づき関連内容を変更。
2020/08/18	1.2J	● マニュアルの構造を最適化。 ● 第 5 章“IP の呼び出し”の内容を最適化。
2021/06/21	1.3J	● IP 呼び出しの一部の図面を更新。 ● IP 呼び出しの"Help"情報を削除。
2021/10/12	1.3.1J	RESET、CE などの説明を更新。
2022/07/14	1.3.2J	第 2 章の注記を削除。
2023/01/05	1.3.3J	IP 呼び出しの一部の図面を更新、"Device Version"オプションを追加。

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 Gowin 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：

- GW1N シリーズ FPGA 製品データシート(DS100)
- GW1NR シリーズ FPGA 製品データシート(DS117)
- GW1NS シリーズ FPGA 製品データシート(DS821)
- GW1NZ シリーズ FPGA 製品データシート(DS841)
- GW1NSR シリーズ FPGA 製品データシート(DS861)
- GW1NSE シリーズ安全 FPGA 製品データシート(DS871)
- GW1NSER シリーズ安全 FPGA 製品データシート(DS881)
- GW1NRF シリーズ Bluetooth FPGA 製品データシート(DS891)
- GW2A シリーズ FPGA 製品データシート(DS102)
- GW2AR シリーズ FPGA 製品データシート(DS226)
- GW2ANR シリーズ FPGA 製品データシート(DS961)
- GW2AN-18X & 9X FPGA 製品データシート(DS971)

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
ALU54	54-bit Arithmetic Logic Unit	54 ビットの算術論理演算装置
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
FFT	Fast Fourier Transformation	高速フーリエ変換
FIR	Finite Impulse Response	有限インパルス応答フィルター
MULT	Multiplier	乗算器
PADD	Pre-adder	前置加算器

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 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 DSP を使いこなせるよう作成されています。

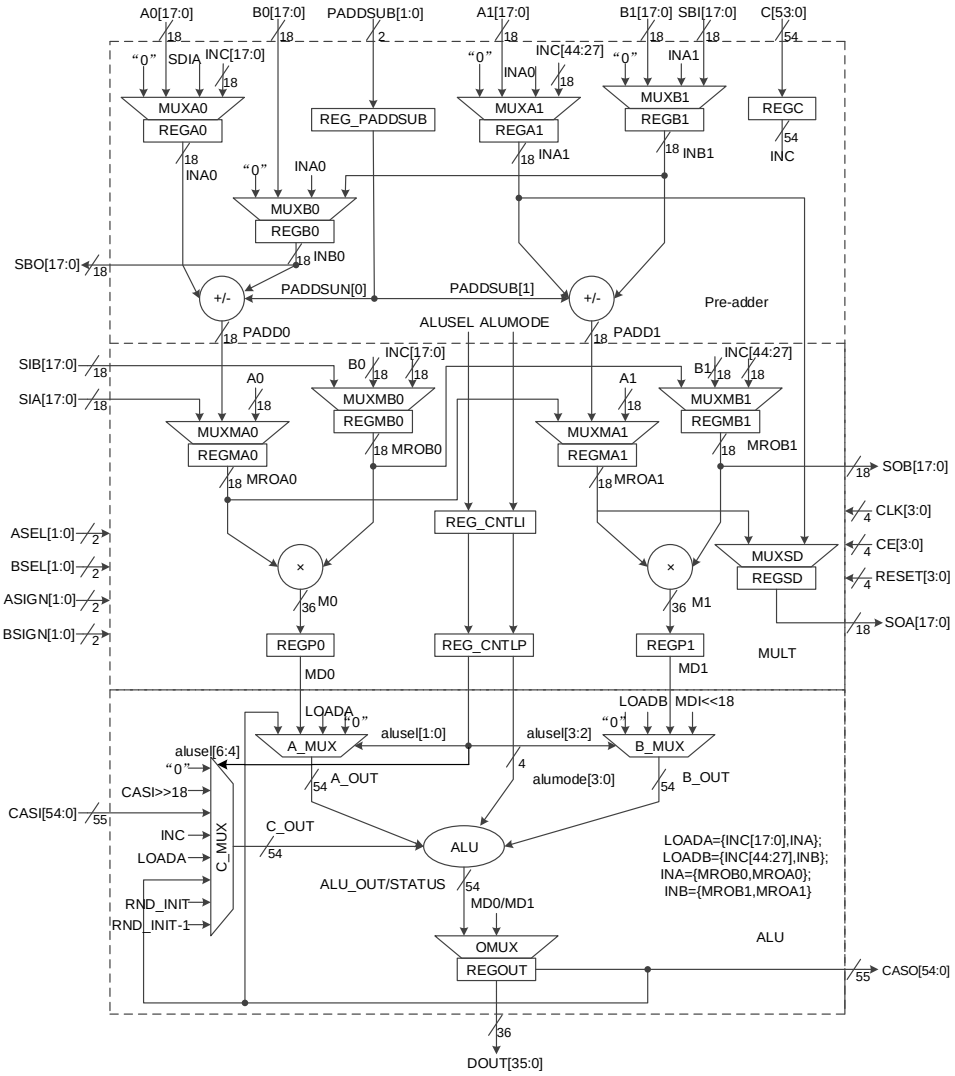
DSP ブロックの機能及び特性は以下の通りです：

- 3 つの幅(9 ビット、18 ビット、36 ビット)の乗算器
- 54 ビットの ALU
- 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
- バレルシフタ
- フィードバック信号による適応フィルタリング
- レジスタのパイプラインとバイパス機能をサポート

3 DSP の構造

GOWIN セミコンダクターFPGA 製品の DSP ブロックは、FPGA アレイ内に行として配置されています。DSP ブロックは 2 つのマクロセルから構成されます。各マクロセルには、2 つの前置加算器(Pre-adder)、2 つの 18-bit 乗算器(MULT18X18)、および 1 つの 3 入力算術論理演算装置(ALU54)が含まれます。マクロセルの構造は、図 3-1 に示す通りです。

図 3-1 マクロセルの構造



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
A0[17:0]	I	18-bit データ入力 A0
B0[17:0]	I	18-bit データ入力 B0
A1[17:0]	I	18-bit データ入力 A1
B1[17:0]	I	18-bit データ入力 B1
C[53:0]	I	54-bit データ入力 C
SIA[17:0]	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

ポート名	I/O タイプ	説明
		クの出力信号 SOA に直接接続されます。
SIB[17:0]	I	カスケード接続に使用されるシフトデータ入力 B。入力信号 SIB は、前の隣接する DSP ブロックの出力信号 SOB に直接接続されます。
SBI[17:0]	I	前置加算器のシフト入力、逆方向
CASI[54:0]	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 ALU 入力
ASEL[1:0]	I	前置加算器または乗算器の A 入力ソース選択
BSEL[1:0]	I	乗算器の B 入力ソース選択
ASIGN[1:0]	I	入力信号 A の符号ビット
BSIGN[1:0]	I	入力信号 B の符号ビット
PADDSUB[1:0]	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[3:0]	I	クロック入力
CE[3:0]	I	クロックイネーブル信号、アクティブ High
RESET[3:0]	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
SOA[17:0]	O	シフトデータ出力 A
SOB[17:0]	O	シフトデータ出力 B
SBO[17:0]	O	前置加算器のシフト出力、逆方向
DOUT[35:0]	O	DSP 出力データ
CASO[54:0]	O	カスケード接続用。最上位ビットは符号ビット。

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
REGA0	A0 入力レジスタ
REGA1	A1 入力レジスタ
REGB0	B0 入力レジスタ
REGB1	B1 入力レジスタ
REGC	C 入力レジスタ
REGMA0	左乗数 A0 入力レジスタ
REGMA1	右乗数 A1 入力レジスタ
REGMB0	左乗数 B0 入力レジスタ
REGMB1	右乗数 B1 入力レジスタ
REGP0	左乗算器パイプライン出力レジスタ
REGP1	右乗算器パイプライン出力レジスタ
REGOUT	DOUT 出力レジスタ
REG_CNTLI	制御信号の初段レジスタ
REG_CNTLP	制御信号の二段目レジスタ
REGSD	SOA シフト出力レジスタ

4 DSP プリミティブ

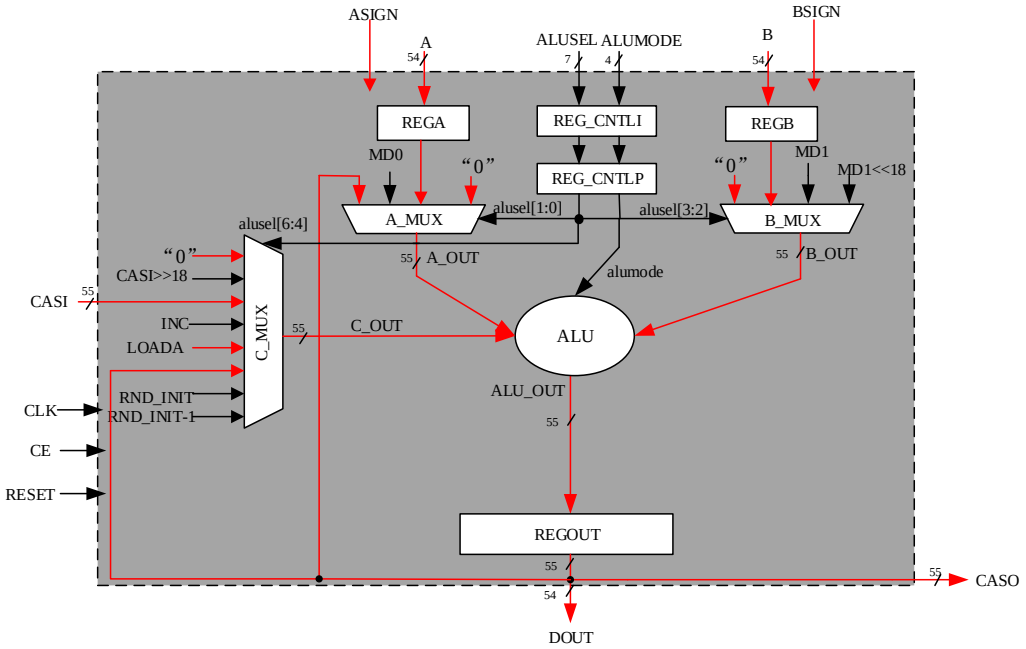
4.1 ALU54

プリミティブの紹介

ALU54D(54-bit Arithmetic Logic Unit)は 54 ビットの算術論理演算を実現する 54 ビットの算術論理演算装置です。

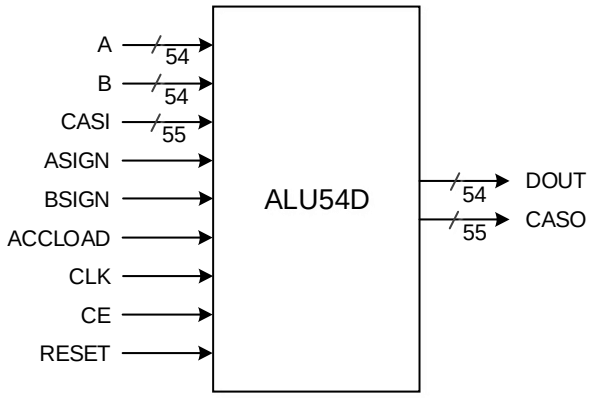
構造

図 4-1 ALU54D の構造



ポート図

図 4-2 ALU54D のポート図



ポートの説明

表 4-1 ALU54D のポート図

ポート	I/O	説明
A[53:0]	入力	54-bit データ入力信号 A
B[53:0]	入力	54-bit データ入力信号 B
CASI[54:0]	入力	55-bit カスケード接続入力信号
ASIGN	入力	A 符号ビット入力信号
BSIGN	入力	B 符号ビット入力信号
ACCLOAD	入力	アキュムレータ Reload モード選択信号。値が 0 の場合は 0 をリロードし、値が 1 の場合は累加します
CLK	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
RESET	入力	リセット入力、アクティブ High
DOUT[53:0]	出力	ALU54D データ出力信号
CASO[54:0]	出力	55-bit カスケード接続出力信号

パラメータの説明

表 4-2 ALU54D のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG	1'b0,1'b1	1'b0	入力 A レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BREG	1'b0,1'b1	1'b0	入力 B レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ACCLOAD_REG	1'b0,1'b1	1'b0	ACCLOAD レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
OUT_REG	1'b0,1'b1	1'b0	出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT 加減算モード選択 1’b0：加算 1’b1：減算
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT 加減算モード選択 1’b0：加算 1’b1：減算
ALUMODE	0,1,2	0	ALU54 動作モードおよび入力選択 0:ACC/0 +/- B +/- A; 1:ACC/0 +/- B + CASI; 2:A +/- B + CASI;
ALU_RESET_MODE	“SYNC”,” ASYNC”	“SYNC”	リセットモードの構成 SYNC：同期リセット

パラメータ	範囲	デフォルト	説明
			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：

```
ALU54D alu54_inst (
  .A(a[53:0]),
  .B(b[53:0]),
  .CASI(casi[54:0]),
       .ASIGN(asign),
  .BSIGN(bsign),
  .ACCLOAD(accload),
  .CE(ce),
  .CLK(clk),
  .RESET(reset),
  .DOUT(dout[53:0]),
  .CASO(caso[54:0])
);
defparam alu54_inst.AREG=1'b1;
defparam alu54_inst.BREG=1'b1;
defparam alu54_inst.ASIGN_REG=1'b0;
defparam alu54_inst.BSIGN_REG=1'b0;
defparam alu54_inst.ACCLOAD_REG=1'b1;
defparam alu54_inst.OUT_REG=1'b0;
defparam alu54_inst.B_ADD_SUB=1'b0;
defparam alu54_inst.C_ADD_SUB=1'b0;
defparam alu54_inst.ALUMODE=0;
defparam alu54_inst.ALU_RESET_MODE="SYNC";
```

VHDL でのインスタンス化：

```
COMPONENT ALU54D
        GENERIC (AREG:bit:='0';
                        BREG:bit:='0';
                        ASIGN_REG:bit:='0';
                        BSIGN_REG:bit:='0';
                        ACCLOAD_REG:bit:='0';
                        OUT_REG:bit:='0';
                        B_ADD_SUB:bit:='0';
                        C_ADD_SUB:bit:='0';
                        ALUD_MODE:integer:=0;
                        ALU_RESET_MODE:string:="SYNC"
          );
        PORT(
              A:IN std_logic_vector(53 downto 0);
                B:IN std_logic_vector(53 downto 0);
                ASIGN:IN std_logic;
                BSIGN:IN std_logic;
                CE:IN std_logic;
                CLK:IN std_logic;
                RESET:IN std_logic;
                ACCLOAD:IN std_logic;
                CASI:IN std_logic_vector(54 downto 0);
                CASO:OUT std_logic_vector(54 downto 0);
                   DOUT:OUT std_logic_vector(53 downto 0)
          );
END COMPONENT;
uut:ALU54D
      GENERIC MAP (AREG=>'1',
                           BREG=>'1',
                           ASIGN_REG=>'0',
                           BSIGN_REG=>'0',
```

```
                    ACCLOAD_REG=>'1',
                    OUT_REG=>'0',
                    B_ADD_SUB=>'0',
                    C_ADD_SUB=>'0',
                    ALUD_MODE=>0,
                    ALU_RESET_MODE=>"SYNC"
    )
PORT MAP (
        A=>a,
        B=>b,
        ASIGN=>asign,
        BSIGN=>bsign,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        CASO=>caso,
        DOUT=>dout
);
```

4.2 MULT

MULT(Multiplier)は乗算器です。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。$DOUT = A * B$ という乗算を実現できます。

DSP マクロセルは、内蔵の 2 つの乗算器で乗算を行います。Multiplier はデータ幅によって 9x9、18x18、36x36 などの乗算器に構成でき、それぞれプリミティブの MULT9X9、MULT18X18、MULT36X36 に対応します。36x 36 乗算器に構成するには、1 つの DSP ブロック(即ち 2 つのマクロセル)が必要となります。

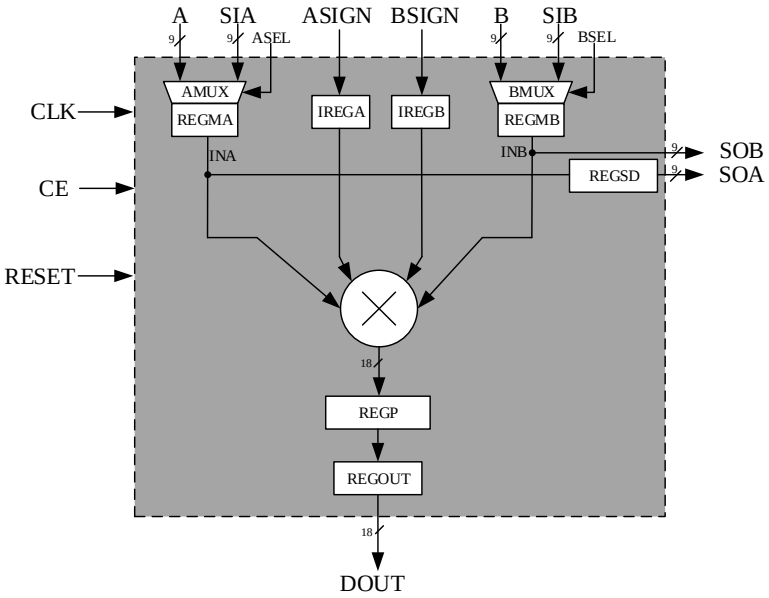
4.2.1 MULT9X9

プリミティブの紹介

MULT9X9(9x9 Multiplier)は 9 ビットの乗算を実現する 9x9 の乗算器です。

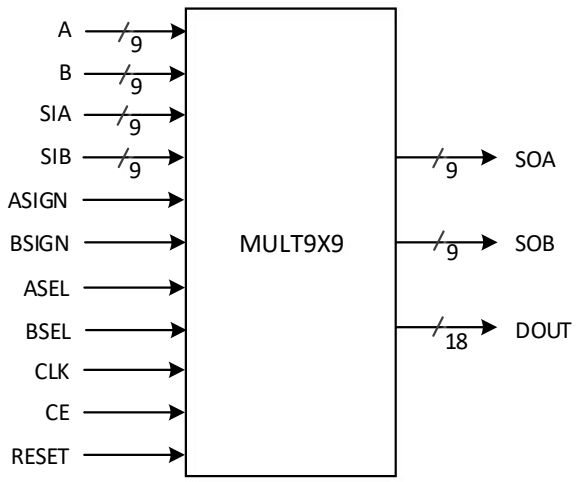
構造

図 4-3 MULT9X9 の構造



ポート図

図 4-4 MULT9X9 のポート図



ポートの説明

表 4-3 MULT9X9 のポートの説明

ポート	I/O	説明
A[8:0]	入力	9-bit データ入力信号 A
B[8:0]	入力	9-bit データ入力信号 B
SIA[8:0]	入力	9-bit シフトデータ入力信号 A
SIB[8:0]	入力	9-bit シフトデータ入力信号 B
ASIGN	入力	A 符号ビット入力信号
BSIGN	入力	B 符号ビット入力信号
ASEL	入力	ソース選択(SIA または A)
BSEL	入力	ソース選択(SIB または B)
CLK	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
RESET	入力	リセット入力、アクティブ High
DOUT[17:0]	出力	データ出力
SOA[8:0]	出力	シフトデータ出力信号 A
SOB[8:0]	出力	シフトデータ出力信号 B

パラメータの説明

表 4-4 MULT9X9 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG	1'b0,1'b1	1'b0	入力 A(SIA または A)レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BREG	1'b0,1'b1	1'b0	入力 B(SIB または B)レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
OUT_REG	1'b0,1'b1	1'b0	出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
PIPE_REG	1'b0,1'b1	1'b0	Pipeline レジスタ 1’b0：バイパスモード

パラメータ	範囲	デフォルト	説明
			1’b1：レジスタモード
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
SOA_REG	1'b0,1'b1	1'b0	SOA レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
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

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：

```
MULT9X9 uut(
        .DOUT(dout[17:0]),
    .SOA(soa[8:0]),
    .SOB(sob[8:0]),
      .A(a[8:0]),
      .B(b[8:0]),
      .SIA(sia[8:0]),
      .SIB(sib[8:0]),
      .ASIGN(asign),
      .BSIGN(bsign),
      .ASEL(asel),
      .BSEL(bsel),
      .CE(ce),
      .CLK(clk),
```

```
        .RESET(reset)
  );
defparam uut.AREG=1'b1;
defparam uut.BREG=1'b1;
defparam uut.OUT_REG=1'b1;
defparam uut.PIPE_REG=1'b0;
defparam uut.ASIGN_REG=1'b0;
defparam uut.BSIGN_REG=1'b0;
defparam uut.SOA_REG=1'b0;
defparam uut.MULT_RESET_MODE="ASYNC";
```

VHDL でのインスタンス化：

```
COMPONENT MULT9X9
        GENERIC (AREG:bit:='0';
                        BREG:bit:='0';
                        OUT_REG:bit:='0';
                        PIPE_REG:bit:='0';
                        ASIGN_REG:bit:='0';
                        BSIGN_REG:bit:='0';
                        SOA_REG:bit:='0';
                        MULT_RESET_MODE:string:="SYNC"
          );
        PORT(
                A:IN std_logic_vector(8 downto 0);
                  B:IN std_logic_vector(8 downto 0);
                  SIA:IN std_logic_vector(8 downto 0);
                  SIB:IN std_logic_vector(8 downto 0);
                  ASIGN:IN std_logic;
                  BSIGN:IN std_logic;
                  ASEL:IN std_logic;
                  BSEL:IN std_logic;
                  CE:IN std_logic;
```

```
                CLK:IN std_logic;
                RESET:IN std_logic;
                SOA:OUT std_logic_vector(8 downto 0);
                SOB:OUT std_logic_vector(8 downto 0);
                DOUT:OUT std_logic_vector(17 downto 0)
            );
END COMPONENT;
uut:MULT9X9
    GENERIC MAP (AREG=>'1',
                    BREG=>'1',
                    OUT_REG=>'1',
                    PIPE_REG=>'0',
                    ASIGN_REG=>'0',
                    BSIGN_REG=>'0',
                    SOA_REG=>'0',
                    MULT_RESET_MODE=>"ASYNC"
     )
    PORT MAP (
        A=>a,
        B=>b,
        SIA=>sia,
        SIB=>sib,
        ASIGN=>asign,
        BSIGN=>bsign,
        ASEL=>asel,
        BSEL=>bsel,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        SOA=>soa,
        SOB=>sob,
```

```
        DOUT=>dout
    );
```

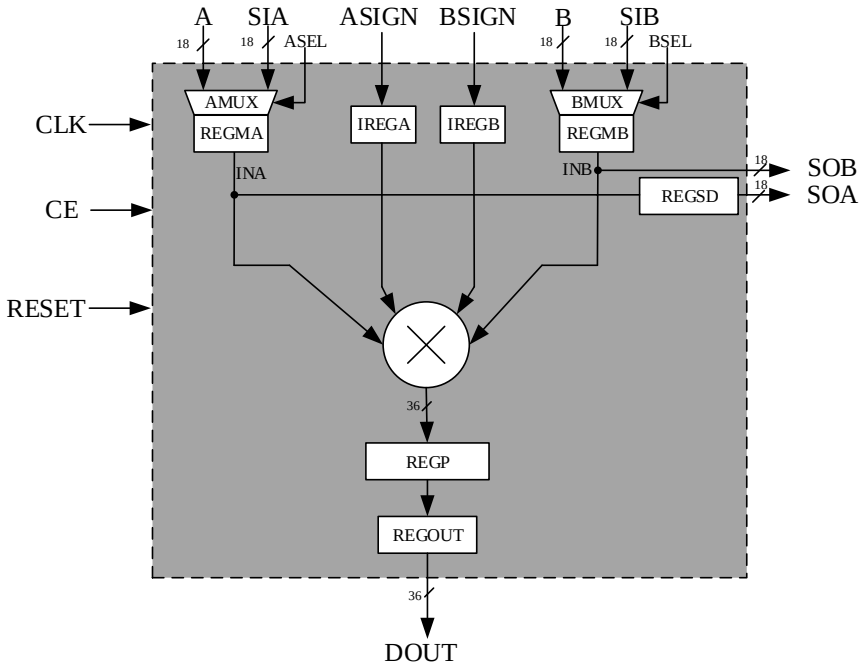
4.2.2 MULT18X18

プリミティブの紹介

MULT18X18(18x18 Multiplier)は 18 ビットの乗算を実現する 18x18 の乗算器です。

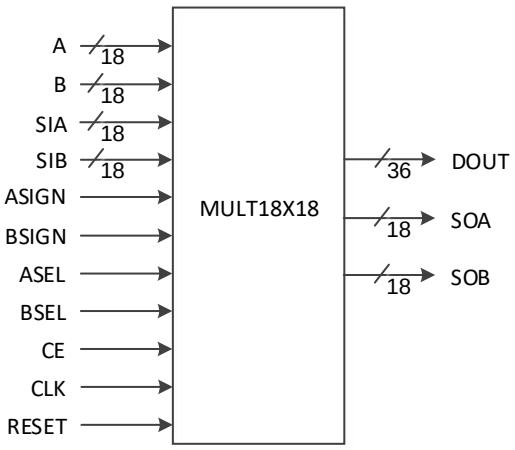
構造

図 4-5 MULT18X18 の構造



ポート図

図 4-6 MULT18X18 のポート図



ポートの説明

表 4-5 MULT18X18 のポートの説明

ポート	I/O	説明
A[17:0]	入力	18-bit データ入力信号 A
B[17:0]	入力	18-bit データ入力信号 B
SIA[17:0]	入力	18-bit シフトデータ入力信号 A
SIB[17:0]	入力	18-bit シフトデータ入力信号 B
ASIGN	入力	A 符号ビット入力信号
BSIGN	入力	B 符号ビット入力信号
ASEL	入力	ソース選択(SIA または A)
BSEL	入力	ソース選択(SIB または B)
CLK	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
RESET	入力	リセット入力、アクティブ High
DOUT[35:0]	出力	データ出力
SOA[17:0]	出力	シフトデータ出力信号 A
SOB[17:0]	出力	シフトデータ出力信号 B

パラメータの説明

表 4-6 MULT18X18 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG	1'b0,1'b1	1'b0	入力 A(SIA または A)レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BREG	1'b0,1'b1	1'b0	入力 B(SIB または B)レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
OUT_REG	1'b0,1'b1	1'b0	出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
PIPE_REG	1'b0,1'b1	1'b0	Pipeline レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
SOA_REG	1'b0,1'b1	1'b0	SOA レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
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

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：

MULT18X18 uut(

```
        .DOUT(dout[35:0]),
        .SOA(soa[17:0]),
        .SOB(sob[17:0]),
        .A(a[17:0]),
        .B(b[17:0]),
        .SIA(sia[17:0]),
        .SIB(sib[17:0]),
        .ASIGN(asign),
        .BSIGN(bsign),
        .ASEL(asel),
        .BSEL(bsel),
        .CE(ce),
        .CLK(clk),
        .RESET(reset)
  );
defparam uut.AREG=1'b1;
defparam uut.BREG=1'b1;
defparam uut.OUT_REG=1'b1;
defparam uut.PIPE_REG=1'b0;
defparam uut.ASIGN_REG=1'b0;
defparam uut.BSIGN_REG=1'b0;
defparam uut.SOA_REG=1'b0;
defparam uut.MULT_RESET_MODE="ASYNC";
```

VHDL でのインスタンス化：

```
COMPONENT MULT18X18
        GENERIC (AREG:bit:='0';
                        BREG:bit:='0';
                        OUT_REG:bit:='0';
                        PIPE_REG:bit:='0';
                        ASIGN_REG:bit:='0';
                        BSIGN_REG:bit:='0';
```

```
                    SOA_REG:bit:='0';
                    MULT_RESET_MODE:string:="SYNC"
        );
      PORT(
          A:IN std_logic_vector(17 downto 0);
            B:IN std_logic_vector(17 downto 0);
            SIA:IN std_logic_vector(17 downto 0);
            SIB:IN std_logic_vector(17 downto 0);
            ASIGN:IN std_logic;
            BSIGN:IN std_logic;
            ASEL:IN std_logic;
            BSEL:IN std_logic;
            CE:IN std_logic;
            CLK:IN std_logic;
            RESET:IN std_logic;
               SOA:OUT std_logic_vector(17 downto 0);
            SOB:OUT std_logic_vector(17 downto 0);
            DOUT:OUT std_logic_vector(35 downto 0)
        );
END COMPONENT;
uut:MULT18X18
    GENERIC MAP (AREG=>'1',
                    BREG=>'1',
                    OUT_REG=>'1',
                    PIPE_REG=>'0',
                    ASIGN_REG=>'0',
                    BSIGN_REG=>'0',
                    SOA_REG=>'0',
                    MULT_RESET_MODE=>"ASYNC"
      )
    PORT MAP (
```

```
        A=>a,
        B=>b,
        SIA=>sia,
        SIB=>sib,
        ASIGN=>asign,
        BSIGN=>bsign,
        ASEL=>asel,
        BSEL=>bsel,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        SOA=>soa,
        SOB=>sob,
        DOUT=>dout
    );
```

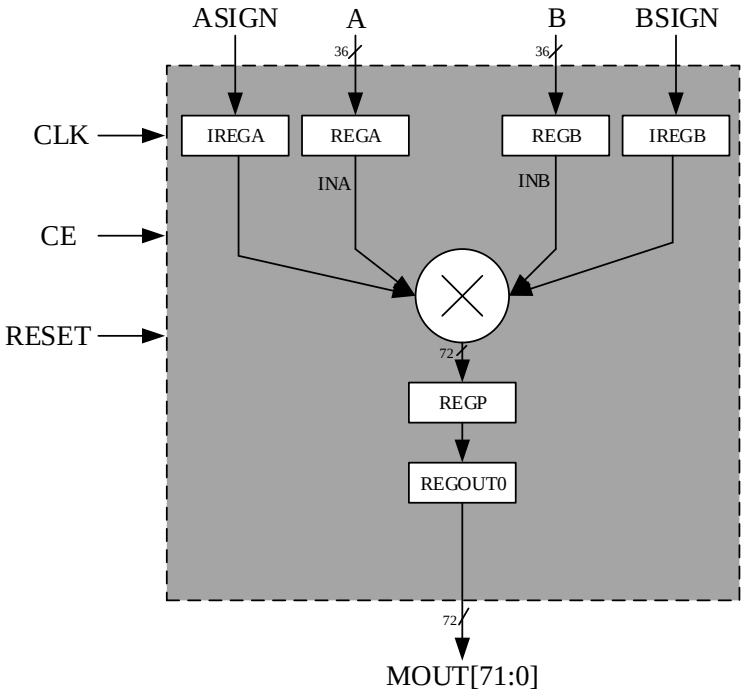
4.2.3 MULT36X36

プリミティブの紹介

MULT36X36(36x36 Multiplier)は 36 ビットの乗算を実現する 36x36 の乗算器です。

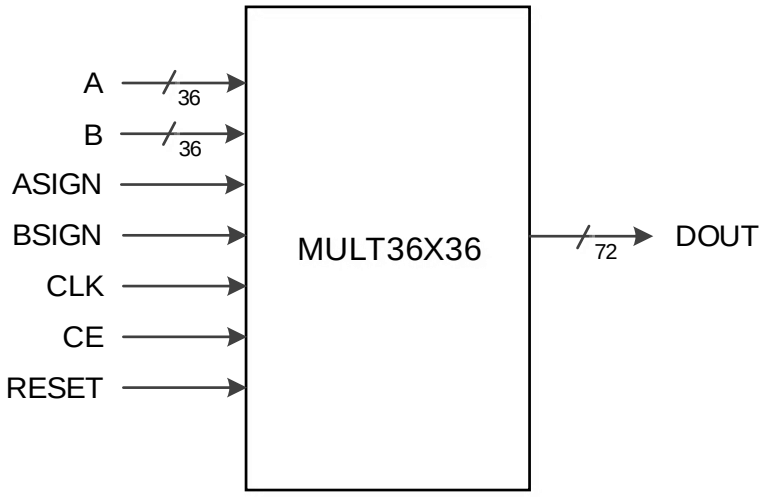
構造

図 4-7 MULT36X36 の構造



ポート図

図 4-8 MULT36X36 のポート図



ポートの説明

表 4-7 MULT36X36 のポートの説明

ポート	I/O	説明
A[35:0]	入力	36-bit データ入力信号 A
B[35:0]	入力	36-bit データ入力信号 B
ASIGN	入力	A 符号ビット入力信号
BSIGN	入力	B 符号ビット入力信号
CLK	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
RESET	入力	リセット入力、アクティブ High
DOUT[71:0]	出力	データ出力

パラメータの説明

表 4-8 MULT36X36 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG	1'b0,1'b1	1'b0	入力 A レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BREG	1'b0,1'b1	1'b0	入力 B レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
OUT0_REG	1'b0,1'b1	1'b0	初段出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
OUT1_REG	1'b0,1'b1	1'b0	2 段目出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
PIPE_REG	1'b0,1'b1	1'b0	Pipeline レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 入力レジスタ

パラメータ	範囲	デフォルト	説明
			1’b0：バイパスモード 1’b1：レジスタモード
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

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：

```
MULT36X36 uut(
        .DOUT(mout[71:0]),
        .A(mdia[35:0]),
        .B(mdib[35:0]),
        .ASIGN(asign),
        .BSIGN(bsign),
        .CE(ce),
        .CLK(clk),
        .RESET(reset)
  );
defparam uut.AREG=1'b1;
defparam uut.BREG=1'b1;
defparam uut.OUT0_REG=1'b0;
defparam uut.OUT1_REG=1'b0;
defparam uut.PIPE_REG=1'b0;
defparam uut.ASIGN_REG=1'b1;
defparam uut.BSIGN_REG=1'b1;
defparam uut.MULT_RESET_MODE="ASYNC";
```

VHDL でのインスタンス化：

```
COMPONENT MULT36X36
```

```
        GENERIC (AREG:bit:='0';
                 BREG:bit:='0';
                 OUT0_REG:bit:='0';
                 OUT1_REG:bit:='0';
                 PIPE_REG:bit:='0';
                 ASIGN_REG:bit:='0';
                 BSIGN_REG:bit:='0';
                 MULT_RESET_MODE:string:="SYNC"
          );
        PORT(
             A:IN std_logic_vector(35 downto 0);
              B:IN std_logic_vector(35 downto 0);
              ASIGN:IN std_logic;
              BSIGN:IN std_logic;
              CE:IN std_logic;
              CLK:IN std_logic;
              RESET:IN std_logic;
                 DOUT:OUT std_logic_vector(71 downto 0)
          );
END COMPONENT;
uut:MULT36X36
      GENERIC MAP (AREG=>'1',
                   BREG=>'1',
                   OUT0_REG=>'0',
                   OUT1_REG=>'0',
                   PIPE_REG=>'0',
                   ASIGN_REG=>'1',
                   BSIGN_REG=>'1',
                   MULT_RESET_MODE=>"ASYNC"
          )
      PORT MAP (
```

```
        A=>mdia,
        B=>mdib,
        ASIGN=>asign,
        BSIGN=>bsign,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        DOUT=>mout
    );
```

4.3 MULTALU

MULTALU モードでは、乗算器の出力は 54-bit ALU 演算が実行されます。MULTALU36X18 と MULTALU18X18 があります。

4.3.1 MULTALU36X18

プリミティブの紹介

MULTALU36X18(36x18 Multiplier with ALU)は ALU 機能付きの 36X18 の乗算器です。

MULTALU36X18 には 3 つの演算モードがあります。

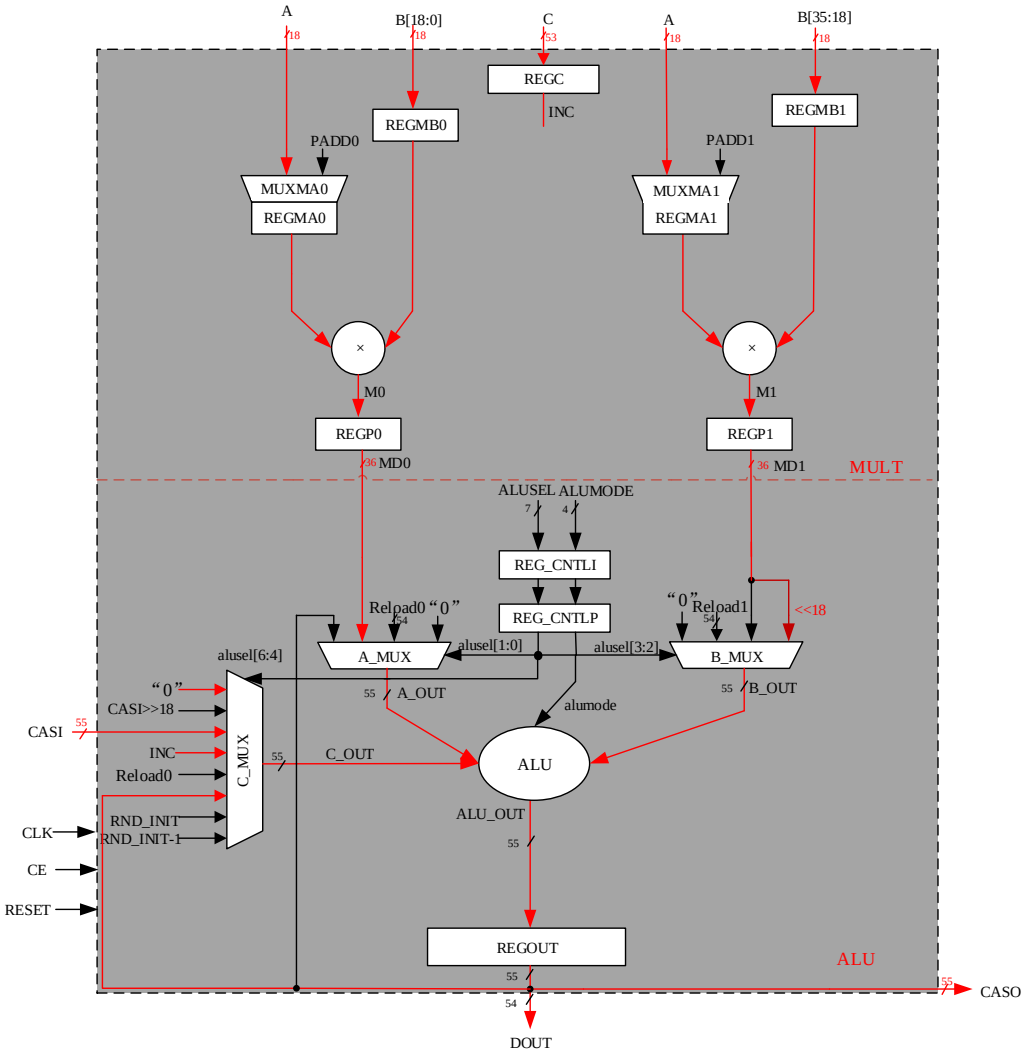
$$DOUT = A*B \pm C$$

$$DOUT = \sum (A*B)$$

$$DOUT = A*B + CASI$$

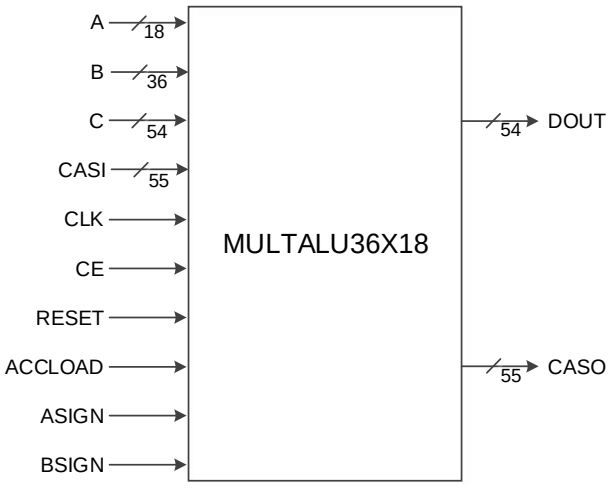
構造

図 4-9 MULTALU36X18 の構造



ポート図

図 4-10 MULTALU36X18 のポート図



ポートの説明

表 4-9 MULTALU36X18 のポート図

ポート	I/O	説明
A[17:0]	入力	18-bit データ入力信号 A
B[35:0]	入力	36-bit データ入力信号 B
C[53:0]	入力	54-bit Reload データ入力信号
CASI[54:0]	入力	55-bit カスケード接続入力信号
ASIGN	入力	A 符号ビット入力信号
BSIGN	入力	B 符号ビット入力信号
CLK	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
RESET	入力	リセット入力、アクティブ High
ACCLOAD	入力	アキュムレータ Reload モード選択信号。値が 0 の場合は 0 をリロードし、値が 1 の場合は累加します
DOUT[53:0]	出力	データ出力
CASO[54:0]	出力	55-bit カスケード接続出力信号

パラメータの説明

表 4-10 MULTALU36X18 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG	1'b0,1'b1	1'b0	入力 A レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BREG	1'b0,1'b1	1'b0	入力 B レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
CREG	1'b0,1'b1	1'b0	入力 C レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
OUT_REG	1'b0,1'b1	1'b0	出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード

パラメータ	範囲	デフォルト	説明
PIPE_REG	1'b0,1'b1	1'b0	Pipeline レジスタ . 1’b0：バイパスモード 1’b1：レジスタモード
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ACCLOAD_REG0	1'b0,1'b1	1'b0	ACCLOAD の初段レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ACCLOAD_REG1	1'b0,1'b1	1'b0	ACCLOAD の二段目レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
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
MULTALU36X18_MODE	0,1,2	0	MULTALU36X18 動作モードおよび入力選択 0:36x18 +/- C; 1:ACC/0 + 36x18; 2:36x18 + CASI
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT 加減算選択 1'b0: add 1'b1: sub

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：

```
MULTALU36X18 multalu36x18_inst(
  .CASO(caso[54:0]),
  .DOUT(dout[53:0]),
  .ASIGN(asign),
  .BSIGN(bsign),
  .CE(ce),
  .CLK(clk),
  .RESET(reset),
  .CASI(casi[54:0]),
  .ACCLOAD(accload),
  .A(a[17:0]),
  .B(b[35:0]),
  .C(c[53:0])
);
    defparam multalu36x18_inst.AREG = 1'b1;
    defparam multalu36x18_inst.BREG = 1'b1;
    defparam multalu36x18_inst.CREG = 1'b0;
    defparam multalu36x18_inst.OUT_REG = 1'b1;
    defparam multalu36x18_inst.PIPE_REG = 1'b0;
    defparam multalu36x18_inst.ASIGN_REG = 1'b0;
    defparam multalu36x18_inst.BSIGN_REG = 1'b0;
    defparam multalu36x18_inst.ACCLOAD_REG0 = 1'b0;
    defparam multalu36x18_inst.ACCLOAD_REG1 = 1'b0;
    defparam multalu36x18_inst.SOA_REG = 1'b0;
    defparam multalu36x18_inst.MULT_RESET_MODE = "SYNC";
    defparam multalu36x18_inst.MULTALU36X18_MODE = 0;
    defparam multalu36x18_inst.C_ADD_SUB = 1'b0;
```

VHDL でのインスタンス化：

```
    COMPONENT MULTALU36X18
        GENERIC (AREG:bit:='0';
                 BREG:bit:='0';
                 CREG:bit:='0';
                 OUT_REG:bit:='0';
```

```
                    PIPE_REG:bit:='0';
                    ASIGN_REG:bit:='0';
                    BSIGN_REG:bit:='0';
                    ACCLOAD_REG0:bit:='0';
                    ACCLOAD_REG1:bit:='0';
                    SOA_REG:bit:='0';
                    MULTALU36X18_MODE:integer:=0;
                    C_ADD_SUB:bit:='0';
                    MULT_RESET_MODE:string:="SYNC"
    );
    PORT(
        A:IN std_logic_vector(17 downto 0);
        B:IN std_logic_vector(35 downto 0);
        C:IN std_logic_vector(53 downto 0);
        ASIGN:IN std_logic;
        BSIGN:IN std_logic;
        CE:IN std_logic;
        CLK:IN std_logic;
        RESET:IN std_logic;
        ACCLOAD:IN std_logic;
        CASI:IN std_logic_vector(54 downto 0);
        CASO:OUT std_logic_vector(54 downto 0);
            DOUT:OUT std_logic_vector(53 downto 0)
    );
END COMPONENT;
uut:MULTALU36X18
    GENERIC MAP (AREG=>'1',
                    BREG=>'1',
                    CREG=>'0',
                    OUT_REG=>'1',
                    PIPE_REG=>'0',
```

```
                    ASIGN_REG=>'0',
                    BSIGN_REG=>'0',
                    ACCLOAD_REG0=>'0',
                    ACCLOAD_REG1=>'0',
                    SOA_REG=>'0',
                    MULTALU36X18_MODE=>0,
                    C_ADD_SUB=>'0',
                    MULT_RESET_MODE=>"SYNC"
    )
    PORT MAP (
        A=>a,
        B=>b,
        C=>c,
        ASIGN=>asign,
        BSIGN=>bsign,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        CASO=>caso,
        DOUT=>dout
    );
```

4.3.2 MULTALU18X18

プリミティブの紹介

MULTALU18X18(18x18 Multiplier with ALU)は ALU 機能付きの 18X18 の乗算器です。

MULTALU18X18 には 3 つの演算モードがあります。

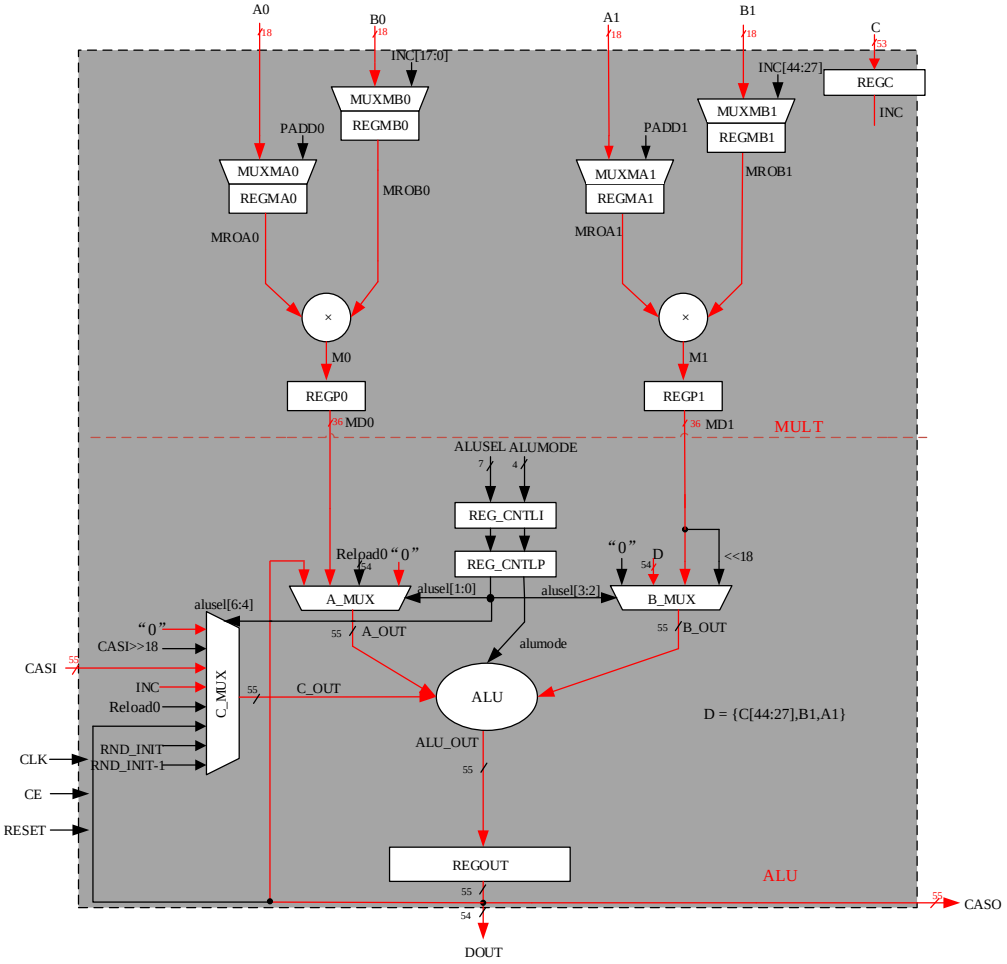
$$DOUT = \sum (A * B) \pm C$$

$$DOUT = \sum (A * B) + CASI$$

$$DOUT = A * B \pm D + CASI$$

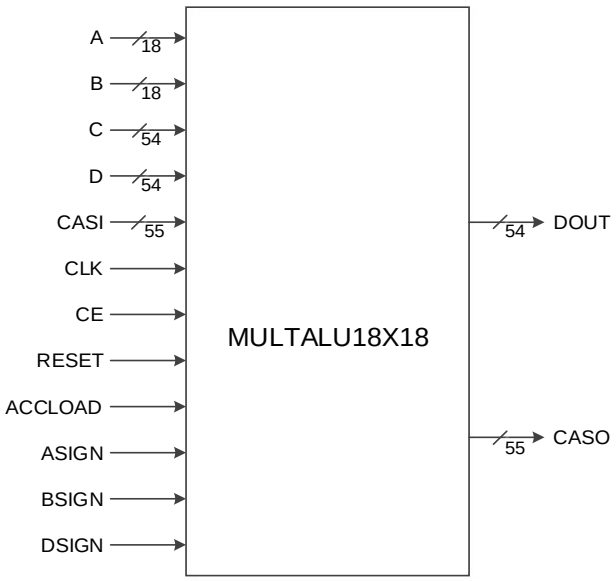
構造

図 4-11 MULTALU18X18 の構造



ポート図

図 4-12 MULTALU18X18 のポート図



ポートの説明

表 4-11 MULTALU18X18 のポートの説明

ポート	I/O	説明
A[17:0]	入力	18-bit データ入力信号 A
B[17:0]	入力	18-bit データ入力信号 B
C[53:0]	入力	54-bit データ入力信号 C
D[53:0]	入力	54-bit データ入力信号 D
CASI[54:0]	入力	55-bit カスケード接続入力信号
ASIGN	入力	A 符号ビット入力信号
BSIGN	入力	B 符号ビット入力信号
DSIGN	入力	D 符号ビット入力信号
CLK	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
RESET	入力	リセット入力、アクティブ High
ACCLOAD	入力	アキュムレータ Reload モード選択信号。値が 0 の場合は 0 をリロードし、値が 1 の場合は累加します
DOUT[53:0]	出力	データ出力
CASO[54:0]	出力	55-bit カスケード接続出力信号

パラメータの説明

表 4-12 MULTALU18X18 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG	1'b0,1'b1	1'b0	入力 A レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BREG	1'b0,1'b1	1'b0	入力 B レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
CREG	1'b0,1'b1	1'b0	入力 C レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
DREG	1'b0,1'b1	1'b0	入力 D レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
DSIGN_REG	1'b0,1'b1	1'b0	DSIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ACCLOAD_REG0	1'b0,1'b1	1'b0	ACCLOAD の初段レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ACCLOAD_REG1	1'b0,1'b1	1'b0	ACCLOAD の二段目レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
MULT_RESET_MODE	“SYNC”,”ASYNC”	“SYNC”	リセットモードの構成 SYNC：同期リセット

パラメータ	範囲	デフォルト	説明
			ASYNC：非同期リセット
PIPE_REG	1'b0,1'b1	1'b0	Pipeline レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
OUT_REG	1'b0,1'b1	1'b0	出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT 加減算モード選択 1’b0：加算 1’b1：減算
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT 加減算モード選択 1’b0：加算 1’b1：減算
MULTALU18X18_MODE	0,1,2	0	MULTALU36X18 動作モードおよび入力選択 0:ACC/0 +/- 18x18 +/- C; 1:ACC/0 +/- 18x18 + CASI; 2: 18x18 +/- D + CASI;

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：

```
MULTALU18X18 multalu18x18_inst(
  .CASO(caso[54:0]),
  .DOUT(dout[53:0]),
  .ASIGN(asign),
  .BSIGN(bsign),
  .DSIGN(dsign),
```

```
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .CASI(casi[54:0]),
    .ACCLOAD(accload),
    .A(a[17:0]),
    .B(b[17:0]),
    .C(c[53:0])
    .D(d[53:0])
);
    defparam multalu18x18_inst.AREG = 1'b1;
    defparam multalu18x18_inst.BREG = 1'b1;
    defparam multalu18x18_inst.CREG = 1'b0;
    defparam multalu18x18_inst.DREG = 1'b0;
    defparam multalu18x18_inst.OUT_REG = 1'b1;
    defparam multalu18x18_inst.PIPE_REG = 1'b0;
    defparam multalu18x18_inst.ASIGN_REG = 1'b0;
    defparam multalu18x18_inst.BSIGN_REG = 1'b0;
    defparam multalu18x18_inst.DSIGN_REG = 1'b0;
    defparam multalu18x18_inst.ACCLOAD_REG0 = 1'b0;
    defparam multalu18x18_inst.ACCLOAD_REG1 = 1'b0;
    defparam multalu18x18_inst.MULT_RESET_MODE = "SYNC";
    defparam multalu18x18_inst.MULTALU18X18_MODE = 0;
    defparam multalu18x18_inst.B_ADD_SUB = 1'b0;
    defparam multalu18x18_inst.C_ADD_SUB = 1'b0;
```

VHDL でのインスタンス化：

```
    COMPONENT MULTALU18X18
        GENERIC (AREG:bit:='0';
                 BREG:bit:='0';
                 CREG:bit:='0';
                 DREG:bit:='0';
                 OUT_REG:bit:='0';
                 PIPE_REG:bit:='0';
                 ASIGN_REG:bit:='0';
                 BSIGN_REG:bit:='0';
```

```
                    DSIGN_REG:bit:='0';
                    ACCLOAD_REG0:bit:='0';
                    ACCLOAD_REG1:bit:='0';
                    B_ADD_SUB:bit:='0';
                    C_ADD_SUB:bit:='0';
                    MULTALU18X18_MODE:integer:=0;
                    MULT_RESET_MODE:string:="SYNC"
        );
      PORT(
          A:IN std_logic_vector(17 downto 0);
            B:IN std_logic_vector(17 downto 0);
            C:IN std_logic_vector(53 downto 0);
            D:IN std_logic_vector(53 downto 0);
            ASIGN:IN std_logic;
            BSIGN:IN std_logic;
            DSIGN:IN std_logic;
            CE:IN std_logic;
            CLK:IN std_logic;
            RESET:IN std_logic;
            ACCLOAD:IN std_logic;
            CASI:IN std_logic_vector(54 downto 0);
            CASO:OUT std_logic_vector(54 downto 0);
               DOUT:OUT std_logic_vector(53 downto 0)
        );
END COMPONENT;
uut:MULTALU18X18
    GENERIC MAP (AREG=>'1',
                    BREG=>'1',
                    CREG=>'0',
                    DREG=>'0',
                    OUT_REG=>'1',
```

```
                    PIPE_REG=>'0',
                    ASIGN_REG=>'0',
                    BSIGN_REG=>'0',
                    DSIGN_REG=>'0',
                    ACCLOAD_REG0=>'0',
                    ACCLOAD_REG1=>'0',
                    B_ADD_SUB=>'0',
                    C_ADD_SUB=>'0',
                    MULTALU18X18_MODE=>0,
                      MULT_RESET_MODE=>"SYNC"
  )
PORT MAP (
        A=>a,
        B=>b,
        C=>c,
        D=>d,
        ASIGN=>asign,
        BSIGN=>bsign,
        DSIGN=>dsign,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        CASO=>caso,
        DOUT=>dout
  );
```

4.4 MULTADDALU

MULTADDALU モードでは、2 つの 18 x 18 乗算器の出力は 54-bit ALU 演算が実行されます。対応するプリミティブは MULTADDALU18X18 です。

MULTALU18X18 には 3 つの演算モードがあります。

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

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

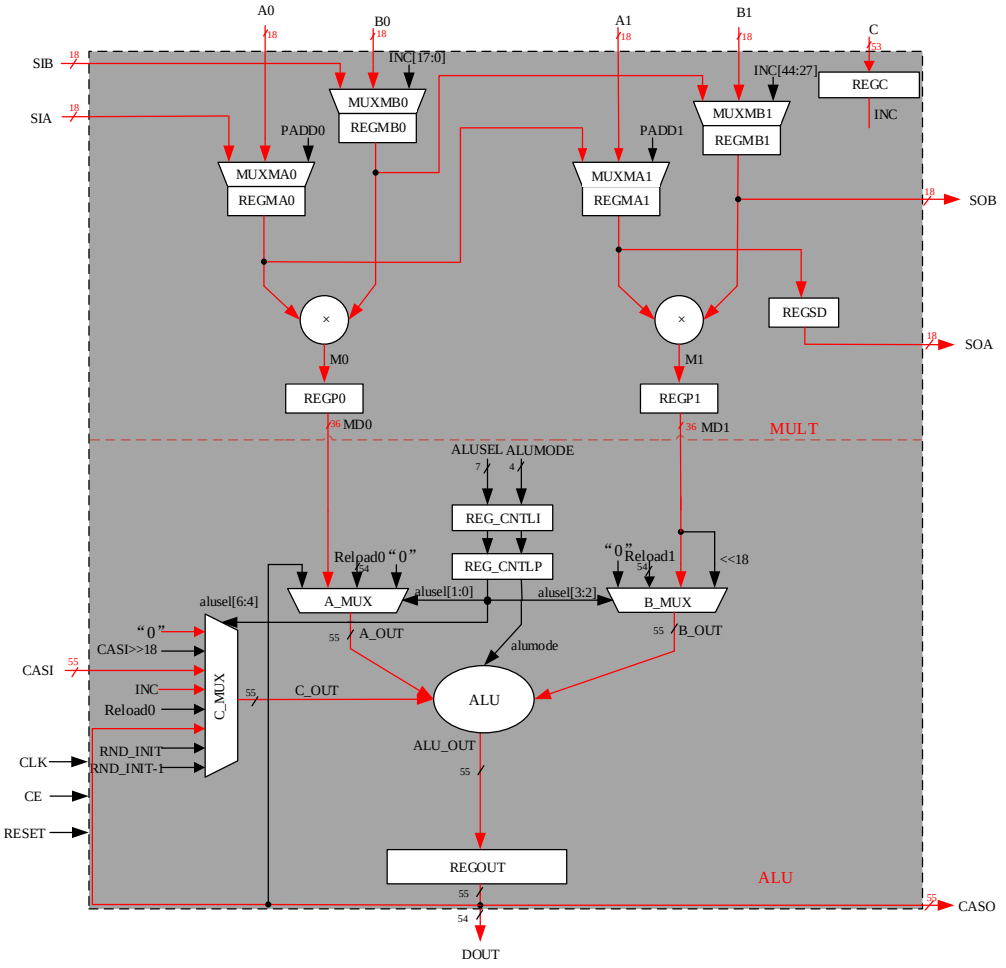
$$DOUT = A0 * B0 \pm A1 * B1 + CASI$$

プリミティブの紹介

MULTADDALU18X18（The Sum of Two 18x18 Multipliers with ALU）は 18 ビット乗算の加算後の累積または reload 演算を実現する ALU 機能付きの 18 ビット乗算加算器です。

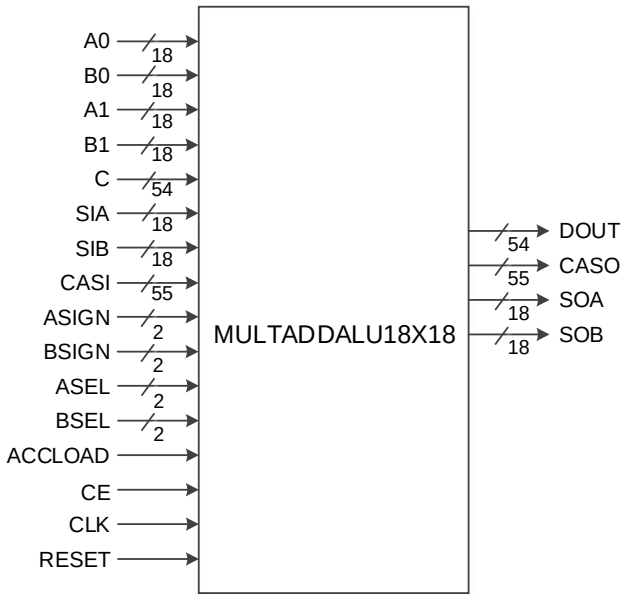
構造

図 4-13 MULTADDALU18X18 の構造



ポート図

図 4-14 MULTADDALU18X18 のポート図



ポートの説明

表 4-13 MULTADDALU18X18 のポートの説明

ポート	I/O	説明
A0[17:0]	入力	18-bit データ入力信号 A0
B0[17:0]	入力	18-bit データ入力信号 B0
A1[17:0]	入力	18-bit データ入力信号 A1
B1[17:0]	入力	18-bit データ入力信号 B1
C[53:0]	入力	54-bit Reload データ入力信号 C
SIA[17:0]	入力	18-bit シフトデータ入力信号 A
SIB[17:0]	入力	18-bit シフトデータ入力信号 B
CASI[54:0]	入力	55-bit カスケード接続入力信号
ASIGN[1:0]	入力	A1,A0 符号ビット入力信号
BSIGN[1:0]	入力	B1,B0 符号ビット入力信号
ASEL[1:0]	入力	入力 A0,A1 ソース選択信号
BSEL[1:0]	入力	入力 B1,B0 ソース選択信号
CLK	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
RESET	入力	リセット入力、アクティブ High
ACCLOAD	入力	アキュムレータ Reload モード選択信

ポート	I/O	説明
		号。値が 0 の場合は 0 をリロードし、値が 1 の場合は累加します
DOUT[53:0]	出力	データ出力
CASO[54:0]	出力	55-bit カスケード接続出力信号
SOA[17:0]	出力	シフトデータ出力信号 A
SOB[17:0]	出力	シフトデータ出力信号 B

パラメータの説明

表 4-14 MULTADDALU18X18 のパラメータの説明

パラメータ	範囲	デフォルト	説明
A0REG	1'b0,1'b1	1'b0	入力 A0(A0 または SIA)レジスタ. 1’b0：バイパスモード 1’b1：レジスタモード
A1REG	1'b0,1'b1	1'b0	入力 A1(A1 またはレジスタ出力 A0)レジスタ. 1’b0：バイパスモード 1’b1：レジスタモード
B0REG	1'b0,1'b1	1'b0	入力 B0(B0 または SIB)レジスタ. 1’b0：バイパスモード 1’b1：レジスタモード
B1REG	1'b0,1'b1	1'b0	入力 B1(B1 またはレジスタ出力 B0)レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
CREG	1'b0,1'b1	1'b0	入力 C レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
PIPE0_REG	1'b0,1'b1	1'b0	Multiplier0 Pipeline レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
PIPE1_REG	1'b0,1'b1	1'b0	Multiplier1 Pipeline レジスタ 1’b0：バイパスモード 1’b1：レジスタモード

パラメータ	範囲	デフォルト	説明
OUT_REG	1'b0,1'b1	1'b0	出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ASIGN0_REG	1'b0,1'b1	1'b0	ASIGN[0]入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ASIGN1_REG	1'b0,1'b1	1'b0	ASIGN[1]入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ACCLOAD_REG0	1'b0,1'b1	1'b0	ACCLOAD の初段レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ACCLOAD_REG1	1'b0,1'b1	1'b0	ACCLOAD の二段目レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BSIGN0_REG	1'b0,1'b1	1'b0	BSIGN[0]入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BSIGN1_REG	1'b0,1'b1	1'b0	BSIGN[1]入力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
SOA_REG	1'b0,1'b1	1'b0	SOA レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT 加減算選択 1’b0：加算 1’b1：減算
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT 加減算選択 1’b0：加算 1’b1：減算
MULTADDALU18X18_MODE	0,1,2	0	MULTADDALU18X18 動作モードおよび入力選択 0:18x18 +/- 18x18 +/- C; 1: ACC/0 + 18x18 +/- 18x18;

パラメータ	範囲	デフォルト	説明
			2:18x18 +/- 18x18 + CASI
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

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：

```
MULTADDALU18X18 uut(
    .DOUT(dout[53:0]),
    .CASO(caso[54:0]),
    .SOA(soa[17:0]),
    .SOB(sob[17:0]),
    .A0(a0[17:0]),
    .B0(b0[17:0]),
    .A1(a1[17:0]),
    .B1(b1[17:0]),
    .C(c[53:0]),
    .SIA(sia[17:0]),
    .SIB(sib[17:0]),
    .CASI(casi[54:0]),
    .ACCLOAD(accload),
    .ASEL(asel[1:0]),
    .BSEL(bsel[1:0]),
    .ASIGN(asign[1:0]),
    .BSIGN(bsign[1:0]),
    .CLK(clk),
    .CE(ce),
```

```
        .RESET(reset)
);
defparam uut.A0REG = 1'b0;
defparam uut.A1REG = 1'b0;
defparam uut.B0REG = 1'b0;
defparam uut.B1REG = 1'b0;
defparam uut.CREG = 1'b0;
defparam uut.PIPE0_REG = 1'b0;
defparam uut.PIPE1_REG = 1'b0;
defparam uut.OUT_REG = 1'b0;
defparam uut.ASIGN0_REG = 1'b0;
defparam uut.ASIGN1_REG = 1'b0;
defparam uut.ACCLOAD_REG0 = 1'b0;
defparam uut.ACCLOAD_REG1 = 1'b0;
defparam uut.BSIGN0_REG = 1'b0;
defparam uut.BSIGN1_REG = 1'b0;
defparam uut.SOA_REG = 1'b0;
defparam uut.B_ADD_SUB = 1'b0;
defparam uut.C_ADD_SUB = 1'b0;
defparam uut.MULTADDALU18X18_MODE = 0;
defparam uut.MULT_RESET_MODE = "SYNC";
```

VHDL でのインスタンス化：

```
COMPONENT MULTADDALU18X18
        GENERIC (A0REG:bit:='0';
                        B0REG:bit:='0';
                        A1REG:bit:='0';
                        B1REG:bit:='0';
                        CREG:bit:='0';
                        OUT_REG:bit:='0';
                        PIPE0_REG:bit:='0';
                        PIPE1_REG:bit:='0';
```

```
                ASIGN0_REG:bit:='0';
                BSIGN0_REG:bit:='0';
                ASIGN1_REG:bit:='0';
                BSIGN1_REG:bit:='0';
                ACCLOAD_REG0:bit:='0';
                ACCLOAD_REG1:bit:='0';
                SOA_REG:bit:='0';
                B_ADD_SUB:bit:='0';
                C_ADD_SUB:bit:='0';
                MULTADDALU18X18_MODE:integer:=0;
                MULT_RESET_MODE:string:="SYNC"
  );
PORT(
        A0:IN std_logic_vector(17 downto 0);
          A1:IN std_logic_vector(17 downto 0);
          B0:IN std_logic_vector(17 downto 0);
          B1:IN std_logic_vector(17 downto 0);
          SIA:IN std_logic_vector(17 downto 0);
          SIB:IN std_logic_vector(17 downto 0);
          C:IN std_logic_vector(53 downto 0);
          ASIGN:IN std_logic_vector(1 downto 0);
          BSIGN:IN std_logic_vector(1 downto 0);
          ASEL:IN std_logic_vector(1 downto 0);
          BSEL:IN std_logic_vector(1 downto 0);
          CE:IN std_logic;
          CLK:IN std_logic;
          RESET:IN std_logic;
          ACCLOAD:IN std_logic;
          CASI:IN std_logic_vector(54 downto 0);
          SOA:OUT std_logic_vector(17 downto 0);
          SOB:OUT std_logic_vector(17 downto 0);
```

```
                CASO:OUT std_logic_vector(54 downto 0);
                  DOUT:OUT std_logic_vector(53 downto 0)
    );
END COMPONENT;
uut:MULTADDALU18X18
    GENERIC MAP (A0REG=>'0',
                 B0REG=>'0',
                 A1REG=>'0',
                 B1REG=>'0',
                 CREG=>'0',
                 OUT_REG=>'0',
                 PIPE0_REG=>'0',
                 PIPE1_REG=>'0',
                 ASIGN0_REG=>'0',
                 BSIGN0_REG=>'0',
                 ASIGN1_REG=>'0',
                 BSIGN1_REG=>'0',
                 ACCLOAD_REG0=>'0',
                 ACCLOAD_REG1=>'0',
                 SOA_REG=>'0',
                 B_ADD_SUB=>'0',
                 C_ADD_SUB=>'0',
                 MULTADDALU18X18_MODE=>0,
                   MULT_RESET_MODE=>"SYNC"
     )
    PORT MAP (
        A0=>a0,
        A1=>a1,
        B0=>b0,
        B1=>b1,
        SIA=>sia,
```

```
        SIB=>sib,
        C=>c,
        ASIGN=>asign,
        BSIGN=>bsign,
        ASEL=>asel,
        BSEL=>bsel,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        SOA=>soa,
        SOB=>sob,
        CASO=>caso,
        DOUT=>dout
    );
```

4.5 PADD モード

PADD(Pre-adder)は前置加算、前置減算、及びシフト機能を実現できる前置加算器です。DSP マクロセルには、前置加算、前置減算、およびシフト機能を実装するための 2 つの前置加算器があります。DSP マクロセルの最先端に位置する前置加算器は 2 つの入力ポートがあります。1 つの入力ポートはパラレル 18-bit 入力 A または SIA、もう 1 つの入力ポートはパラレル 18-bit 入力 B または SBI です。タイミング機能を強化するため、すべての入力ポートには対応するレジスタが配置されています。また、前置加算器をバイパスすることにより、入力ポート A と B を直接乗算器に接続することもできます。GOWIN セミコンダクターFPGA 製品の前置加算器は、機能モジュールとして単独で使用することができ、ビット幅によって 9-bit の PADD9 及び 18-bit の PADD18 に分類できます。

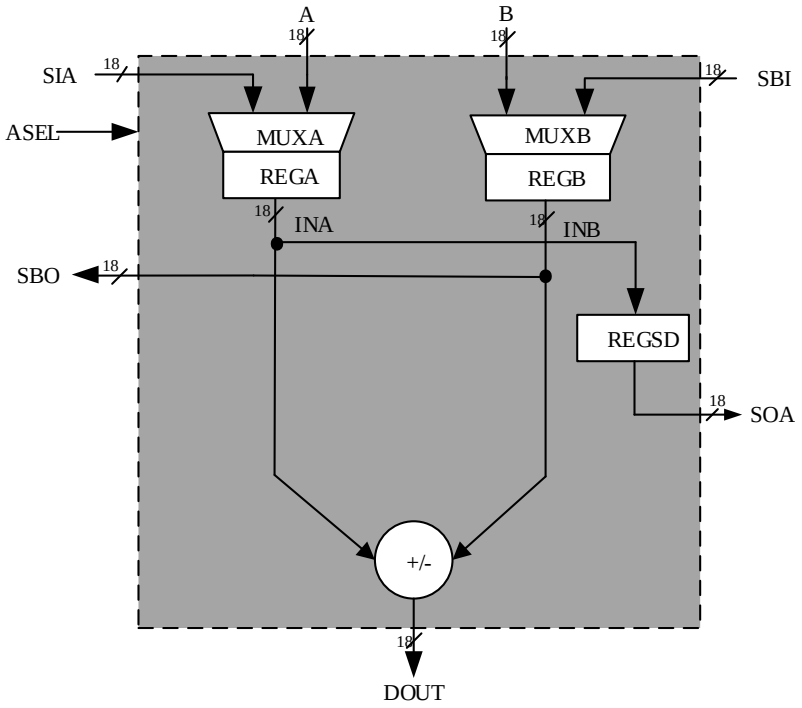
4.5.1 PADD18

プリミティブの紹介

PADD18(18-bit Pre-Adder)は 18 ビットの前置加算、前置減算、またはシフト機能を実現します。

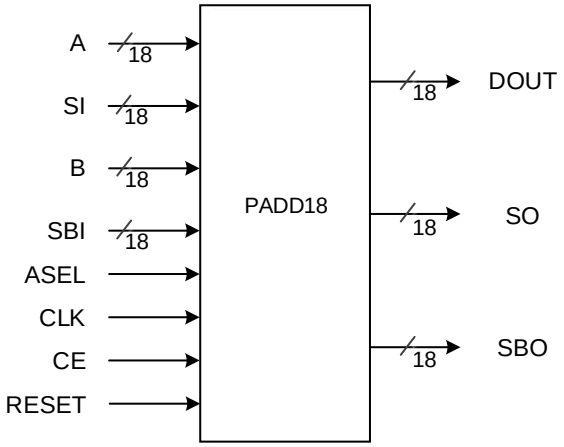
構造

図 4-15 PADD18 の構造



ポート図

図 4-16 PADD18 のポート図



ポートの説明

表 4-15 PADD18 のポートの説明

ポート	I/O	説明
A[17:0]	入力	18-bit データ入力信号 A
B[17:0]	入力	18-bit データ入力信号 B

ポート	I/O	説明
SI[17:0]	入力	シフトデータ入力信号 A
SBI[17:0]	入力	前置加算器のシフト入力信号、逆方向
ASEL	入力	ソース選択入力信号(SI または A)
CLK	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
RESET	入力	リセット入力、アクティブ High
SO[17:0]	出力	シフトデータ出力信号 A
SBO[17:0]	出力	前置加算器のシフト出力信号、逆方向
DOUT[17:0]	出力	データ出力

パラメータの説明

表 4-16 PADD18 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG	1'b0,1'b1	1'b0	入力 A(A または SI)レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
BREG	1'b0,1'b1	1'b0	入力 B(B または SBI)レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ADD_SUB	1'b0,1'b1	1'b0	加減算選択 1’b0：加算 1’b1：減算
PADD_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
BSEL_MODE	1'b1,1'b0	1'b1	入力 B 選択 1'b1: SBI 1'b0: B
S または EG	1'b0,1'b1	1'b0	シフト出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード

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：

```
PADD18 padd18_inst(
        .A(a[17:0]),
        .B(b[17:0]),
        .SO(so[17:0]),
        .SBO(sbo[17:0]),
        .DOUT(dout[17:0]),
        .SI(si[17:0]),
        .SBI(sbi[17:0]),
        .CE(ce),
        .CLK(clk),
        .RESET(reset),
        .ASEL(asel)
);
defparam padd18_inst.AREG = 1'b0;
defparam padd18_inst.BREG = 1'b0;
defparam padd18_inst.ADD_SUB = 1'b0;
defparam padd18_inst.PADD_RESET_MODE = "SYNC";
defparam padd18_inst.SOREG = 1'b0;
defparam padd18_inst.BSEL_MODE = 1'b0;
```

VHDL でのインスタンス化：

```
COMPONENT PADD18
        GENERIC (AREG:bit:='0';
                        BREG:bit:='0';
                        SOREG:bit:='0';
                        ADD_SUB:bit:='0';
                        PADD_RESET_MODE:string:="SYNC" ;
```

```
                    BSEL_MODE:bit:='0'
        );
    PORT(
            A:IN std_logic_vector(17 downto 0);
            B:IN std_logic_vector(17 downto 0);
            ASEL:IN std_logic;
            CE:IN std_logic;
            CLK:IN std_logic;
            RESET:IN std_logic;
            SI:IN std_logic_vector(17 downto 0);
            SBI:IN std_logic_vector(17 downto 0);
           SO:OUT std_logic_vector(17 downto 0);
            SBO:OUT std_logic_vector(17 downto 0);
            DOUT:OUT std_logic_vector(17 downto 0)
    );
END COMPONENT;
uut:PADD18
    GENERIC MAP (AREG=>'0',
                    BREG=>'0',
                    SOREG=>'0',
                    ADD_SUB=>'0',
                    PADD_RESET_MODE=>"SYNC",
                    BSEL_MODE=>'0'
        )
    PORT MAP (
        A=>a,
        B=>b,
        ASEL=>asel,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
```

```
        SI=>si,
        SBI=>sbi,
        SO=>so,
        SBO=>sbo,
        DOUT=>dout
    );
```

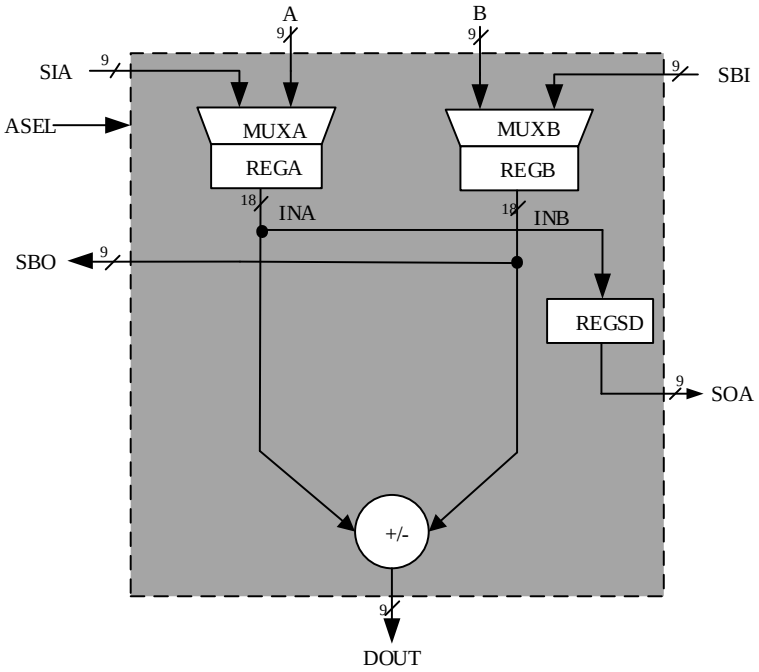
4.5.2 PADD9

プリミティブの紹介

PADD9(9-bit Pre-Adder)は 9 ビットの前置加算、前置減算、またはシフト機能を実現します。

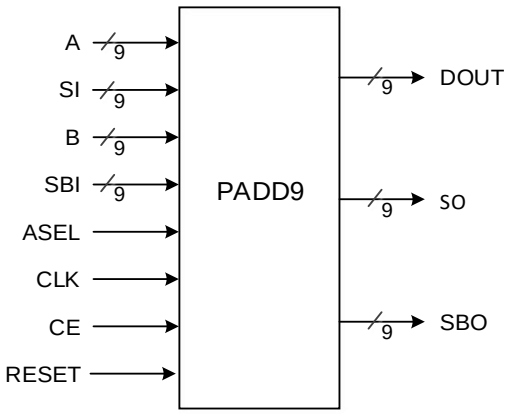
構造

図 4-17 PADD9 の構造



ポート図

図 4-18 PADD9 のポート図



ポートの説明

表 4-17 PADD9 のポートの説明

ポート	I/O	説明
A[8:0]	入力	9-bit データ入力信号 A
B[8:0]	入力	9-bit データ入力信号 B
SI[8:0]	入力	シフトデータ入力信号 A
SBI[8:0]	入力	前置加算器のシフト入力信号、逆方向
ASEL	入力	ソース選択入力信号(SI または A)
CLK	入力	クロック入力信号
CE	入力	クロックイネーブル信号、アクティブ High
RESET	入力	リセット入力、アクティブ High
SO[8:0]	出力	シフトデータ出力信号 A
SBO[8:0]	出力	前置加算器のシフト出力信号、逆方向
DOUT[8:0]	出力	データ出力

パラメータの説明

表 4-18 PADD9 のパラメータの説明

パラメータ	範囲	デフォルト	説明
AREG	1'b0,1'b1	1'b0	入力 A(A または SI)レジスタ 1’b0：バイパスモード

パラメータ	範囲	デフォルト	説明
			1’b1：レジスタモード
BREG	1'b0,1'b1	1'b0	入力 B(B または SBI)レジスタ 1’b0：バイパスモード 1’b1：レジスタモード
ADD_SUB	1'b0,1'b1	1'b0	加減算選択 1’b0：加算 1’b1：減算
PADD_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
BSEL_MODE	1'b1,1'b0	1'b1	入力 B 選択 1'b1: SBI 1'b0: B
SOREG	1'b0,1'b1	1'b0	シフト出力レジスタ 1’b0：バイパスモード 1’b1：レジスタモード

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：

```
PADD9 padd9_inst(
        .A(a[8:0]),
        .B(b[8:0]),
        .SO(so[8:0]),
        .SBO(sbo[8:0]),
        .DOUT(dout[8:0]),
        .SI(si[8:0]),
        .SBI(sbi[8:0]),
        .CE(ce),
        .CLK(clk),
        .RESET(reset),
        .ASEL(asel)
);
defparam padd9_inst.AREG = 1'b0;
defparam padd9_inst.BREG = 1'b0;
defparam padd9_inst.ADD_SUB = 1'b0;
defparam padd9_inst.PADD_RESET_MODE = "SYNC";
defparam padd9_inst.SOREG = 1'b0;
defparam padd9_inst.BSEL_MODE = 1'b0;
```

VHDL でのインスタンス化：

```
COMPONENT PADD9
        GENERIC (AREG:bit:='0';
                        BREG:bit:='0';
                        SOREG:bit:='0';
                        ADD_SUB:bit:='0';
                        PADD_RESET_MODE:string:="SYNC" ;
                        BSEL_MODE:bit:='0'
```

```
        );
    PORT(
            A:IN std_logic_vector(8 downto 0);
            B:IN std_logic_vector(8 downto 0);
            ASEL:IN std_logic;
            CE:IN std_logic;
            CLK:IN std_logic;
            RESET:IN std_logic;
            SI:IN std_logic_vector(8 downto 0);
            SBI:IN std_logic_vector(8 downto 0);
           SO:OUT std_logic_vector(8 downto 0);
            SBO:OUT std_logic_vector(8 downto 0);
            DOUT:OUT std_logic_vector(8 downto 0)
    );
END COMPONENT;
uut:PADD9
    GENERIC MAP (AREG=>'0',
                    BREG=>'0',
                    SOREG=>'0',
                    ADD_SUB=>'0',
                    PADD_RESET_MODE=>"SYNC",
                    BSEL_MODE=>'0'
        )
    PORT MAP (
            A=>a,
            B=>b,
            ASEL=>asel,
            CE=>ce,
            CLK=>clk,
            RESET=>reset,
            SI=>si,
```

```
        SBI=>sbi,
        SO=>so,
        SBO=>sbo,
        DOUT=>dout
);
```

5 IP の呼び出し

IP Core Generator の DSP ブロックでは 5 種類の Gowin プリミティブ (ALU54、MULT、MULTADDALU、MULTALU、PADD)がサポートされています。

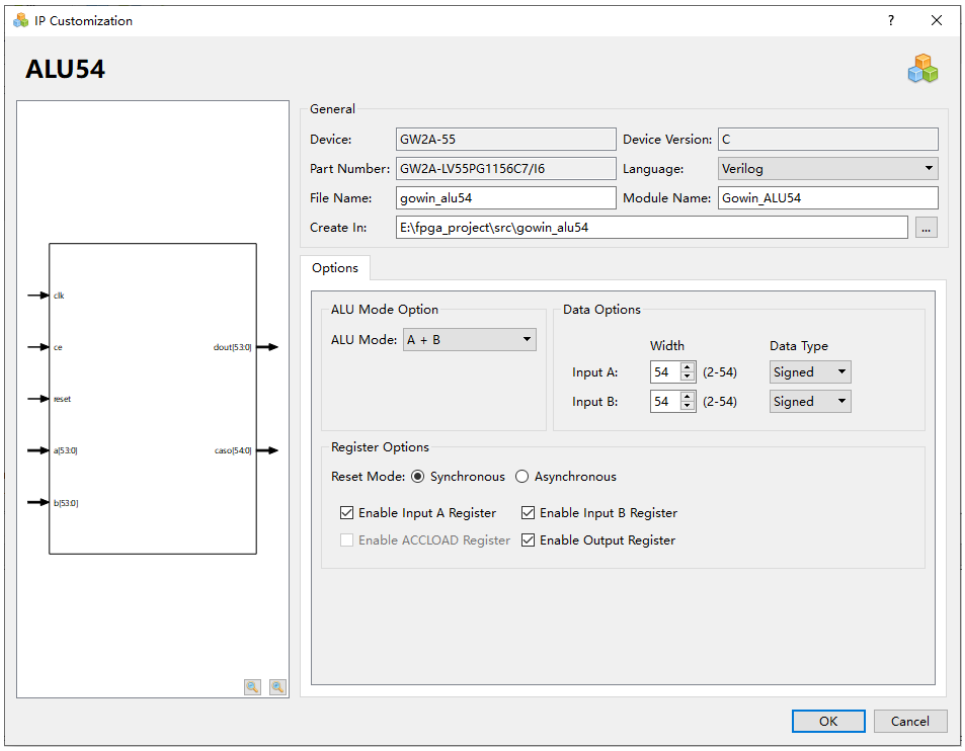
5.1 ALU54

ALU54 は 54 ビットの算術論理演算を実現します。IP Core Generator のインターフェースで ALU54 をクリックすると、右側に ALU54 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“ALU54”をダブルクリックすると、ALU54 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 General 構成タブ、Options 構成タブ、およびポート図があります。

図 5-1 ALU54 IP の構成ウィンドウ



1. General 構成タブは、IP ファイルの構成に使用されます。

- Device：対象デバイス。
- Device Version：デバイスのバージョン。
- 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。同じ場合、エラーメッセージがポップアップします。
- 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. Options 構成タブ：IP のカスタマイズに使用されます(図 5-1)。

- ALU Mode Option：ALU54 の演算モードを構成します。
  - A + B；

- A – B；
- Accum + A + B；
- Accum + A – B；
- Accum - A + B；
- Accum - A – B；
- B + CASI；
- Accum + B + CASI；
- Accum - B + CASI；
- A + B + CASI；
- A - B + CASI；

- Data Options：データオプションを構成します。
  - ALU54 入力データ幅を構成します。入力 A/B ポートのデータは 1～54 ビットに構成できます。
  - 出力ポートのデータ幅はユーザー設定を必要としません。入力データ幅に従って自動的に調整されます。
  - “Data Type”オプションは Signed、Unsigned として構成できます。
- Register Options：レジスタの動作モードを構成します。
  - “Reset Mode”オプションは ALU54 のリセットモードを構成し、同期モード“Synchronous”と非同期モード“Asynchronous”をサポートします。
  - “Enable Input A Register”：チェックすると、Input A registe r がイネーブルされます。
  - “Enable Input B Register”：チェックすると、Input B registe r がイネーブルされます。
  - “Enable ACCLOAD Register”：チェックすると、ACCLOAD register がイネーブルされます。
  - “Enable Output Register”：チェックすると、Output register がイネーブルされます。

3. ポート図：現在の IP Core の構成結果を表示し、入力・出力ポートのビット幅は 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、構成ファイルの“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_alu54.v”は完全な verilog モジュールです。
- gowin_alu54_tmp.v は IP のテンプレートファイルです。
- “gowin_alu54.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。

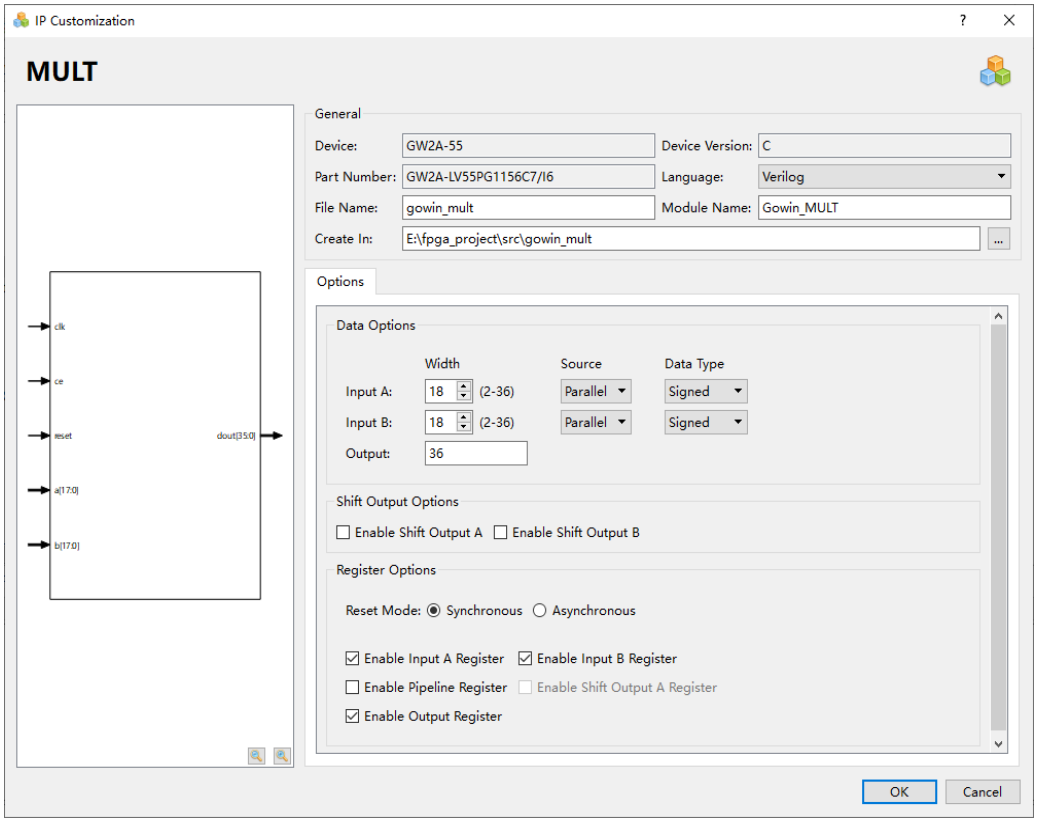
5.2 MULT

MULT は乗算機能を実現します。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 5-2)。このウィンドウには General 構成タブ、Options 構成タブ、およびポート図があります。

図 5-2 MULT IP の構成ウィンドウ



- General 構成タブは、IP ファイルの構成に使用されます。MULT

の General 構成タブの使用は ALU54 モジュールと同様です。5.1 ALU54 を参照してください。

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

- Data Options：データオプションを構成します。
  - 入力ポート(Input A Width/ Input B Width)の最大データ幅は 36 ビットです。
  - 出力ポートのデータ幅(Output Width)はユーザー設定を必要としません。入力データ幅に従って自動的に調整されます。

    インスタンス化の際にデータ幅に従って MULT9X9、MULT18X18、または MULT36X36 を生成します。
  - 入力ポート A/B は Parallel、Shift として構成できます。
  - このデータタイプは Unsigned、Signed として構成できます。
- Shift Output Options：入力ポートのデータ幅(Input A Width/Input B Width)が 18 以下の場合、shift out 機能をイネーブルできます。

注記：

入力ポートのデータ幅(Input A Width/ Input B Width)のいずれかが 18 を超える時、Shift Output Options はグレーアウトし、使用できません。

- Register Options：このオプションの機能と使用法は、ALU54 の Register Options オプションと同じです。5.1 ALU54 の Option 構成タブを参照してください。

- ポート図：現在の IP Core の構成結果を表示し、入力・出力ポートの数およびビット幅は 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、構成ファイルの“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。

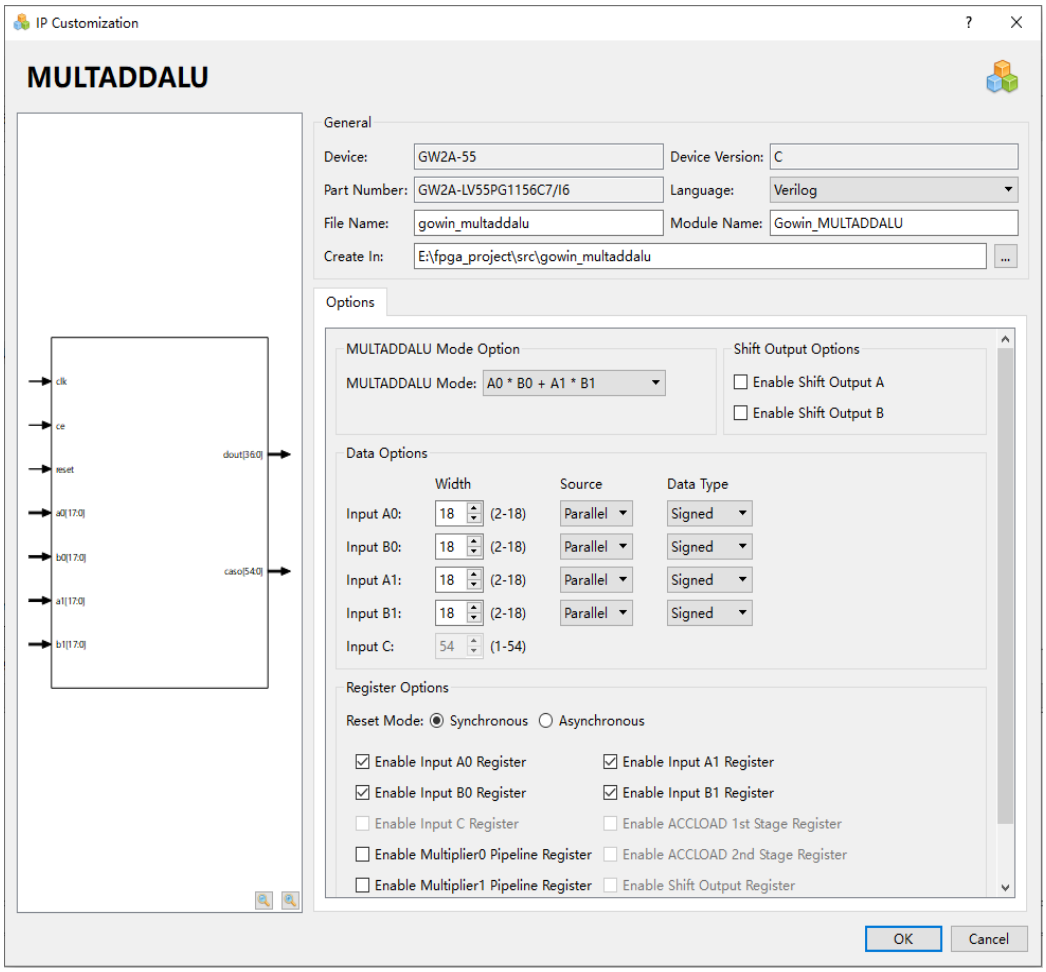
5.3 MULTADDALU

MULTADDALU は、積和機能を実現します。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 General 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



- General 構成タブは、IP ファイルの構成に使用されます。MULTADDALU の General 構成タブの使用は ALU54 モジュールと同様です。5.1 ALU54 を参照してください。
- Options 構成タブは、IP のカスタマイズに使用されます(図 5-3)。

- MULTADDALU Mode Option：MULTADDALU の演算モードを構成します。
  - A0*B0 + A1*B1
  - A0*B0 - A1*B1
  - A0*B0 + A1*B1 + C
  - A0*B0 + A1*B1 - C
  - A0*B0 - A1*B1 + C
  - A0*B0 - A1*B1 - C
  - Accum + A0*B0 + A1*B1
  - Accum + A0*B0 - A1*B1
  - A0*B0 + A1*B1 + CASI
  - A0*B0 - A1*B1 + CASI
- MULTADDALU の Data Options と Register Options 構成タブの使用は MULT モジュールと同様です。5.2 MULT を参照してください。
  - ポート図：現在の IP Core の構成結果を表示し、入力・出力ポートのビット幅は Data Options および Register 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、構成ファイルの“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。

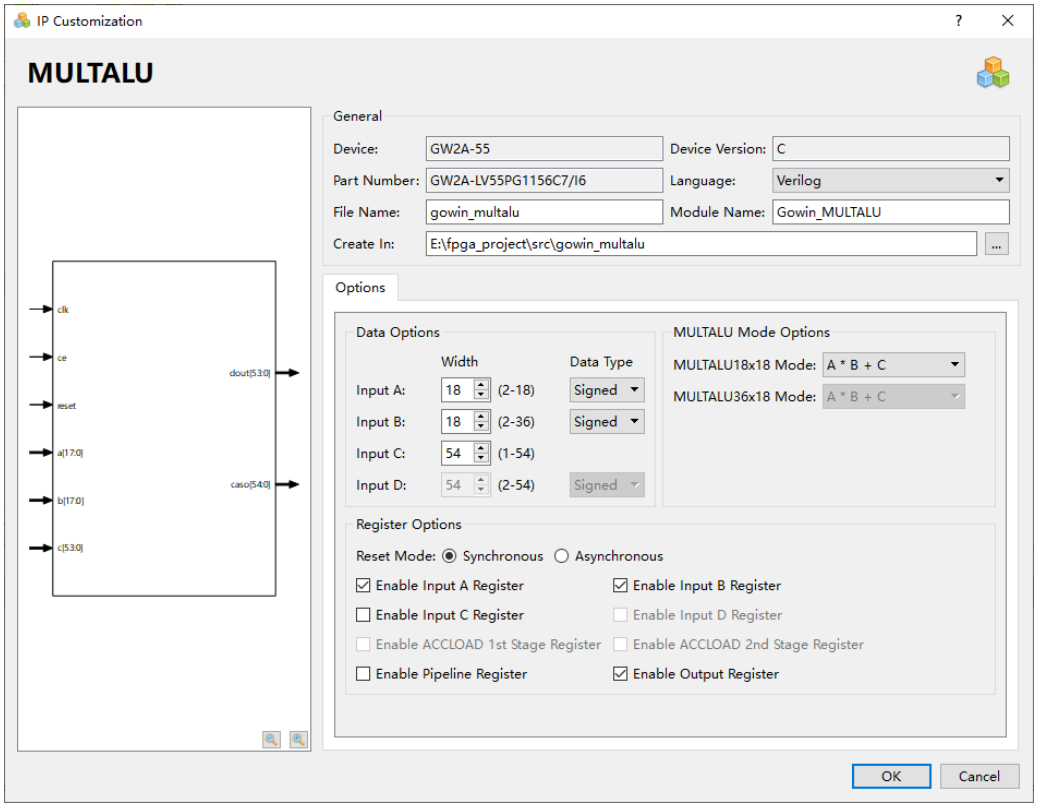
5.4 MULTALU

MULTALU は乗算後の加算または累積を実現します。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 General 構成タブ、Options 構成タブ、およびポート図があります(図 5-4)。

図 5-4 MULTALU IP の構成ウィンドウ



1. General 構成タブは、IP ファイルの構成に使用されます。MULTALU の General 構成タブの使用は ALU54 モジュールと同様です。5.1 ALU54 を参照してください。
2. Options 構成タブは、IP のカスタマイズに使用されます(図 5-4)。

- MULTALU Mode Option：IP Core の MULTALU は入力ポートのビット幅に応じて 2 種類のモジュールを生成できます：MULTALU36X18 または MULTALU18X18。Input A と Input B の width が 18 ビット以下の場合、Options 構成タブの右側にある MULTALU Mode Options の

MULTALU36X18 Mode はグレーアウトします。MULTALU18X18 Mode は次のように構成できます。

- A*B + C
- A*B - C
- Accum + A*B + C
- Accum + A*B - C
- Accum - A*B + C
- Accum - A*B - C
- A*B + CASI
- Accum + A*B + CASI
- Accum - A*B + CASI
- A*B + D + CASI
- A*B - D + CASI

- Input B の width が 18 ビット以上の場合、MULTALU18X18 Mode はグレーアウトします。MULTALU36X18 Mode は次のように構成できます。
  - A*B + C
  - A*B - C
  - Accum + A*B
  - A*B + CASI
- MULTALU の Data Options と Register Options 構成タブの使用は MULT モジュールと同様です。5.2 MULT を参照してください。

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

生成されるファイル

IP の構成が完了したら、構成ファイルの“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_multtalu.v”は完全な verilog モジュールです。
- gowin_multtalu_tmp.v は IP のテンプレートファイルです。
- “gowin_multtalu.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。

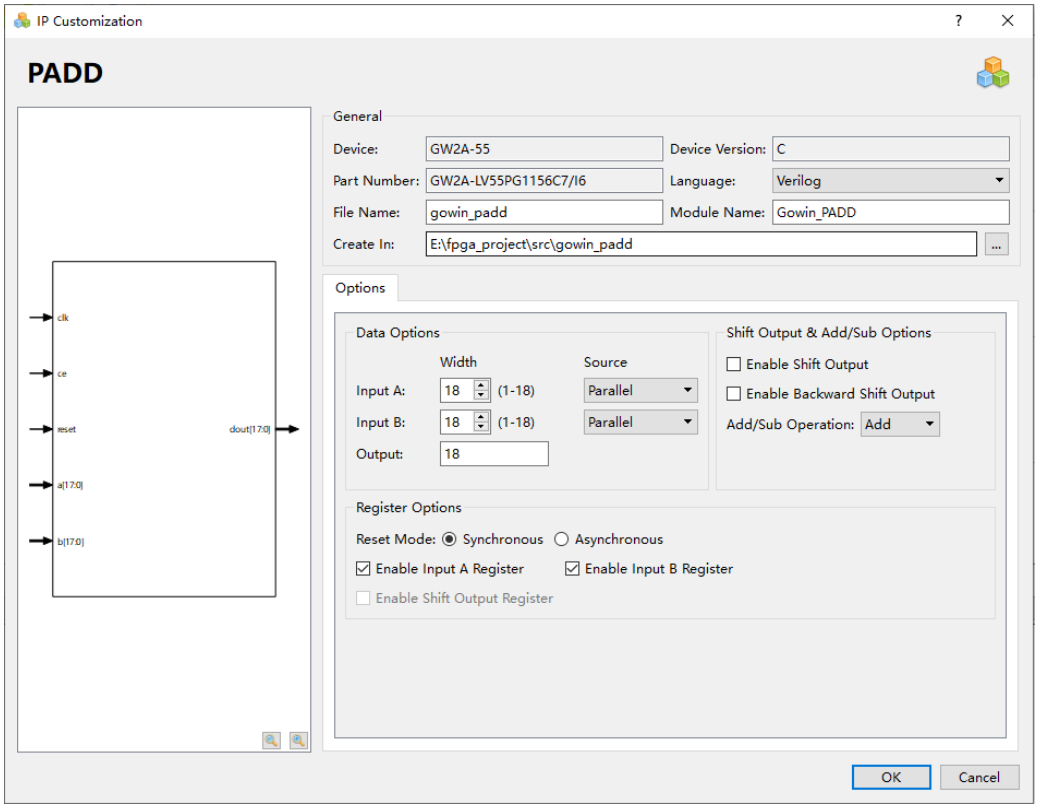
5.5 PADD

PADD は前置加算、前置減算、またはシフト機能を実現します。IP Core Generator のインターフェースで PADD をクリックすると、右側に PADD 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“PADD”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 General 構成タブ、Options 構成タブ、およびポート図があります(図 5-5)。

図 5-5 PADD IP の構成ウィンドウ



- General 構成タブは、IP ファイルの構成に使用されます。PADD の General 構成タブの使用は ALU54 モジュールと同様です。5.1 ALU54 を参照してください。
- Options 構成タブは IP のカスタマイズに使用されます(図 5-5)。
- Data Options：データオプションを構成します。
  - 入力ポート(Input A Width/ Input B Width)の最大データ幅は 18 ビットです。

- 出力ポートのデータ幅(Output Width)はユーザー設定を必要としません。入力データ幅に従って自動的に調整されます。インスタンス化の際にデータ幅に従って PADD9 または PADD18 を生成します。
- 入力ポート A のデータソースは、“Input A Source”オプションを介して Parallel および Shift として構成できます。
- 入力ポート B のデータソースは、“Input B Source”オプションを介して Parallel および Backward Shift として構成できます。

- Shift Output & Add/Sub Options：Shift Output と Backward Shift Output のイネーブル、および加減算の設定が可能です。
  - “Enable Shift Output”をチェックして Shift Output をイネーブルします。
  - “Enable Backward Shift Output”をチェックして Backward Shift Output をイネーブルします。
  - “Add/Sub Operation”オプションで Add または Sub を選択することにより加算または減算を選択します。
- Register Options：レジスタの動作モードを構成します。
  - “Reset Mode”オプションは PADD のリセットモードの構成に使用され、同期モード“Synchronous”と非同期モード“Asynchronous”がサポートされます。
  - “Enable Input A Register”：チェックすると、Input A registe r がイネーブルされます。
  - “Enable Input B Register”：チェックすると、Input B registe r がイネーブルされます。
  - “Enable Output Register”：チェックすると、Output register がイネーブルされます。

- ポート図：現在の IP Core の構成結果を表示し、入力・出力ポートの数およびビット幅は Options 構成に従ってリアルタイムで更新されます(図 5-5)。

生成されるファイル

IP の構成が完了したら、構成ファイルの“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_padd.v”は完全な verilog モジュールです。
- “gowin_padd_tmp.v”は IP のテンプレートファイルです。
- “gowin_padd.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