Gowin Digital Signal Processing (DSP)

User Guide

UG287-1.3.1E, 10/12/2021

Revision History

Date	Version	Description
05/16/2016	1.05E	Initial version published.
07/04/2016	1.06E	Block diagram of PADD18 modified.
07/11/2016	1.07E	The graphics standardized.
08/16/2016	1.08E	The number of multipliers for the GW2A-18 device modified.
11/08/2016	1.09E	The multiplier block diagram modified.
10/09/2017	1.10E	Modified according to the latest primitives.
08/18/2020	1.2E	● The chapter structure modified. ● Chapter 5 IP Configuration optimized.
06/21/2021	1.3E	● The figures in chapter 5 updated. ● Help information removed on IP configuration GUI.
10/12/2021	1.3.1E	The descriptions of RESET, CE, etc. updated.

Contents

List of Figures

List of Tables

1 About This Guide

1.1 Purpose

This manual provides descriptions of DSP structure, signal definition, and configuration, etc., to help you learn Gowin DSP operating flow and enhance design efficiency.

1.2 Related Documents

The latest user guides are available on the GOWINSEMI Website. You can find the related documents at www.gowinsemi.com:

1. DS100, GW1N series of FPGA Products Data Sheet
2. DS117, GW1NR series of FPGA Products Data Sheet
3. DS821, GW1NS series of FPGA Products Data Sheet
4. DS841, GW1NZ series of FPGA Products Data Sheet
5. DS861, GW1NSR series of FPGA Products Data Sheet
6. DS871, GW1NSE series of SecureFPGA Products Data Sheet
7. DS881, GW1NSER series of SecureFPGA Products Data Sheet
8. DS891, GW1NRF series of Bluetooth FPGA Products Data Sheet
9. DS102, GW2A series of FPGA Products Data Sheet
10. DS226, GW2AR series of FPGA Products Data Sheet
11. DS961, GW2ANR series of FPGA Products Data Sheet
12. DS971, GW2AN series of FPGA Products Data Sheet

1.3 Terminology and Abbreviations

The terminology and abbreviations used in this manual are as shown in Table 1-1.

Table 1-1 Terminology and Abbreviations

Terminology and Abbreviations	Meaning
DSP	Digital Signal Processing
FIR	Finite Impulse Response
FFT	Fast Fourier Transformation
CFU	Configurable Function Unit
MULT	Multiplier
PADD	Pre-adder
ALU54	54-bit Arithmetic Logic Unit

1.4 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly by the following ways.

Website: www.gowinsemi.com

E-mail: support@gowinsemi.com

2 Overview

Gowin FPGA products have abundant DSP resources to meet customers' needs for high performance digital signal processing, such as FIR and FFT design. DSP blocks deliver the advantages of stable timing performance, high resource utilization, and low-power. The functions and features of the DSP blocks are as follows:

The functions and features of the DSP blocks are as follows:

- 9-bit, 18-bit, 36-bit multiplier
- 54-bit ALU
- Multiple multipliers can be cascaded to increase data width.
- Barrel shifter
- Adaptive filtering through feedback signal
- Supports registers pipeline and bypass

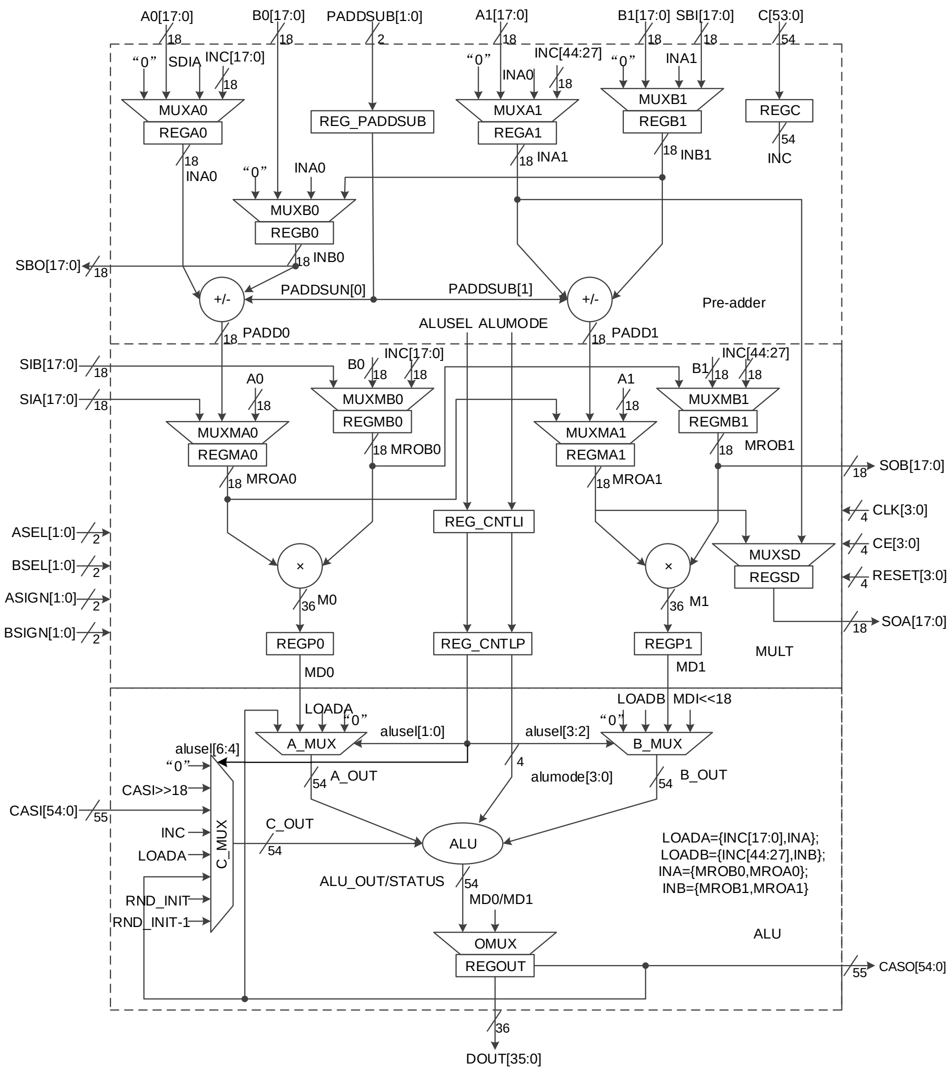
Note!

GW1N-1, GW1N-1S, GW1NR-1, GW1NS-2, GW1NS-2C, GW1NSE-2C, GW1NSR-2, GW1NSR-2C, GW1NZ-1, GW1NZ-1C, GW1N-2, GW1N-1P5, GW1N-2B, GW1N-1P5B, GW1NR-2, GW1NR-2B do not support DSP.

3 DSP Structure

Gowin FPGA products DSP modules are distributed in FPGA arrays in the form of rows. The DSP block includes two macros, and each of which contains two pre-adders, two 18-bit multipliers, and one three-input ALU54. And the macro unit diagram is shown in Figure 3-1.

Figure 3-1 Macro Unit Architecture



DSP port description is as shown in Table 3-1. The internal register description is as shown in Table 3-2. In addition, input signals CLK, CE, and RESET are used to control the registers.

Table 3-1 DSP Port Description

Port	I/O	Description
A0[17:0]	I	18-bit data input A0
B0[17:0]	I	18-bit data input B0
A1[17:0]	I	18-bit data input A1
B1[17:0]	I	18-bit data input B1
C[53:0]	I	54-bit data input C
SIA[17:0]	I	Shift data input A, used for CASCADE connection. The input signal, SIA, is directly connected to the output signal, SOA, of the previously adjacent DSP.
SIB[17:0]	I	Shift data input B, used for CASCADE connection. The input signal, SIB, is directly connected to the output signal, SOB, of the previously adjacent DSP.
SBI[17:0]	I	Pre－adder logic shift input, backward direction.
CASI[54:0]	I	CASO from previous DSP block, ALU cascade input, used for cascade connection.
ASEL[1:0]	I	Source select for Pre-adder or multiplier.
BSEL[1:0]	I	Source select for multiplier input B
ASIGN[1:0]	I	Sign bit for input A
BSIGN[1:0]	I	Sign bit for input B
PADDSUB[1:0]	I	Operating control signal of Pre-adder, used for Pre-adder logic add/subtract selection.
CLK[3:0]	I	Clock input
CE[3:0]	I	Clock enable signal
RESET[3:0]	I	Reset signal, support synchronous/asynchronous mode
SOA[17:0]	O	Shift data output A
SOB[17:0]	O	Shift data output B
SBO[17:0]	O	Pre－adder logic shift output, backward direction.
DOUT[35:0]	O	DSP output data
CASO[54:0]	O	ALU output to next DSP block for cascade connection, the highest bit is sign-extended.

Table 3-2 Description of Internal Registers in DSP Module

Register	Description
REGA0	A0 input register
REGA1	A1 input register
REGB0	B0 input register
REGB1	B1 input register
REGC	C input register
REGMA0	Left multiplier A0 input register
REGMA1	Right multiplier A1 input register
REGMB0	Left multiplier B0 input register
REGMB1	Right multiplier B1 input register
REGP0	Pipeline output register for left multiplier
REGP1	Pipeline output register for right multiplier
REGOUT	Register for DOUT output
REG_CNTLI	The first level register for control signal
REG_CNTLP	The second level register for control signal
REGSD	Register for SOA shift output

4 DSP Primitive

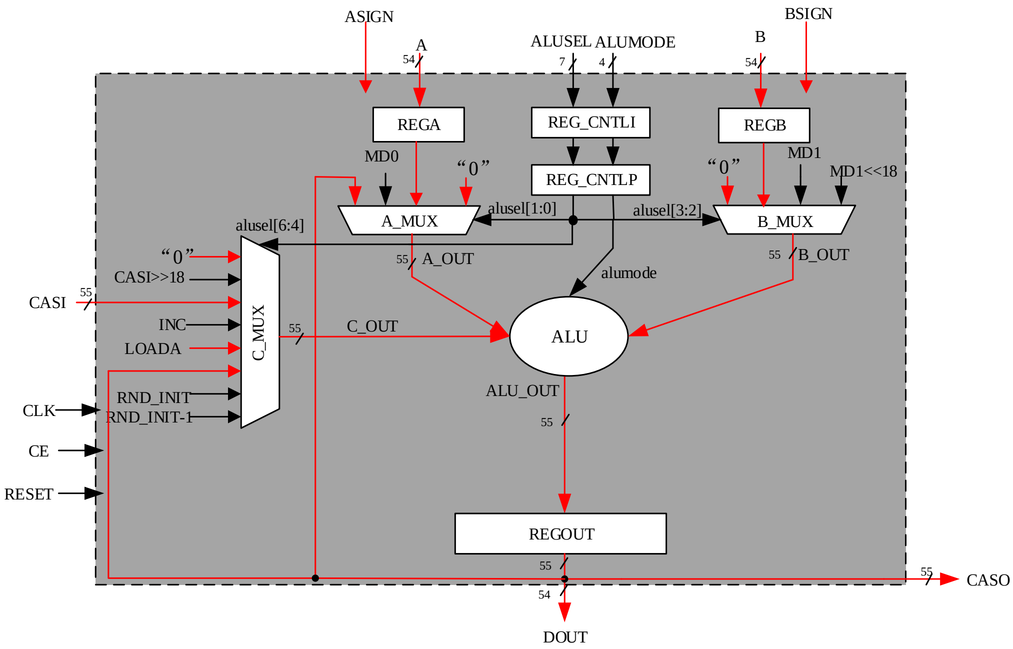
4.1 ALU54

Primitive Introduction

54-bit Arithmetic Logic Unit (ALU54D) is a 54-bit arithmetic logic unit.

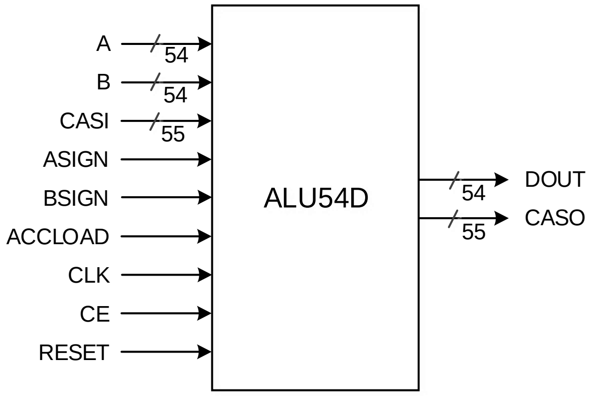
Logic Architecture Diagram

Figure 4-1 ALU54D Logic Architecture Diagram



Port Diagram

Figure 4-2 ALU54D Port Diagram



Port Description

Table 4-1 ALU54D Port Description

Port	I/O	Description
A[53:0]	Input	54-bit data input signal A
B[53:0]	Input	54-bit data input signal B
CASI[54:0]	Input	55-bit for cascade input signal
ASIGN	Input	A sign bit input signal
BSIGN	Input	B sign bit input signal
ACCLOAD	Input	Accumulator reload mode selection signal
CLK	Input	Clock input signal
CE	Input	Clock enable signal, active-high
RESET	Input	Reset input signal, active-high
DOUT[53:0]	Output	ALU54D data output signal
CASO[54:0]	Output	55-bit for cascade output signal

Parameter Description

Table 4-2 ALU54D Parameter Description

Parameter	Range	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A register 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B register 1'b0: bypass mode 1'b1: registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN Input Register 1'b0: bypass mode 1'b1: registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN Input Register 1'b0: bypass mode 1'b1: registered mode
ACCLOAD_REG	1'b0,1'b1	1'b0	ACCLOAD Register 1'b0: bypass mode 1'b1: registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register 1'b0: bypass mode 1'b1: registered mode
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT plus/minus mode selection 1'b0: plus 1'b1: minus
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT plus/minus mode selection 1'b0: plus 1'b1: minus
ALUMODE	0,1, 2	0	ALU54 operation mode and input selection 0:ACC/0 +/- B +/- A; 1:ACC/0 +/- B + CASI; 2:A +/- B + CASI;
ALU_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	Reset mode configuratiom SYNC: synchronized reset

Parameter	Range	Default Value	Description
			ASYNC: asynchronous reset

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more details, you can refer to 5 IP Configuration.

Verilog Instantiation:

```
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 Instantiation:

```
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 is the multiplier unit of the DSP, where the multiplier input signal is defined as A and B, and the product output signal is defined as DOUT, which can implement multiplication: $DOUT = A * B$.

Each DSP macro unit has two multipliers that perform the multiplication. To meet different multiplication bit widths, the MULT mode can be configured as 9x9, 18x18, 36x36 multipliers depending on the data bit width, corresponding to the primitives MULT9X9, MULT18X18, and MULT36X36 respectively. The 36 x 36 multiplier requires one DSP module (i.e., two macro units) to configure.

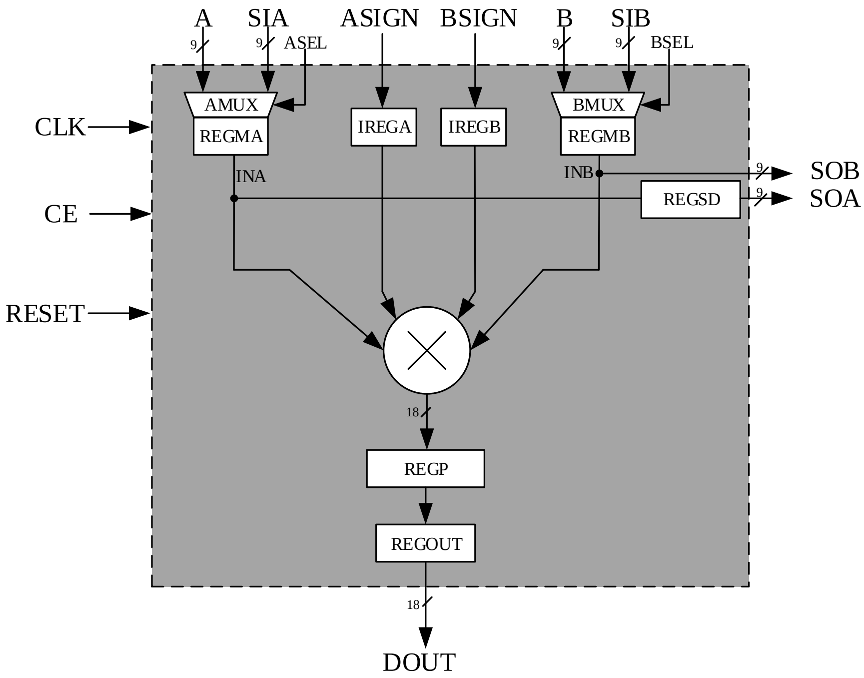
4.2.1 MULT9X9

Primitive Introduction

MULT9X9 supports 9-bit multiplication.

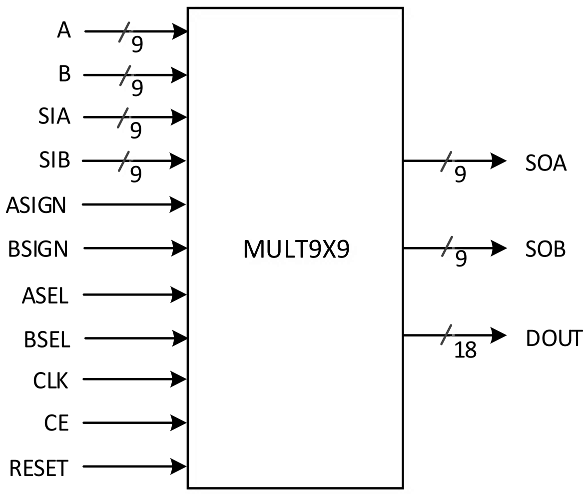
Logic Diagram

Figure 4-3 MULT9X9 Logic Diagram



Port Diagram

Figure 4-4 MULT9X9 Logic Diagram



Port Description

Table 4-3 MULT9X9 Port Description

Port	I/O	Description
A[8:0]	Input	9-bit data input signal A
B[8:0]	Input	9-bit data input signal B
SIA[8:0]	Input	9-bit shift data input signal A
SIB[8:0]	Input	9-bit shift data input signal B
ASIGN	Input	A sign bit input signal
BSIGN	Input	B sign bit input signal
ASEL	Input	Source selection signal, SIA or A.
BSEL	Input	Source selection signal, SIB or B.
CLK	Input	Clock input signal
CE	Input	Clock enable signal, active-high
RESET	Input	Reset input signal, active-high
DOUT[17:0]	Output	Data output signal
SOA[8:0]	Output	Shift data output signal A
SOB[8:0]	Output	Shift data output signal B

Parameter Description

Table 4-4 MULT9X9 Parameter Description

Parameter	Range	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A (SIA or A) register 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B (SIB or B) register 1'b0: bypass mode 1'b1: registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register 1'b0: bypass mode 1'b1: registered mode
PIPE_REG	1'b0,1'b1	1'b0	Pipeline Register 1'b0: bypass mode 1'b1: registered mode

Parameter	Range	Default Value	Description
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN Input Register 1'b0: bypass mode 1'b1: registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN Input Register 1'b0: bypass mode 1'b1: registered mode
SOA_REG	1'b0,1'b1	1'b0	SOA register 1'b0: bypass mode 1'b1: registered mode
MULT_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	Reset mode configuratiom SYNC: synchronized reset ASYNC: asynchronous reset

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Configuration.

Verilog Instantiation:

```
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 Instantiation:

```
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
```

);

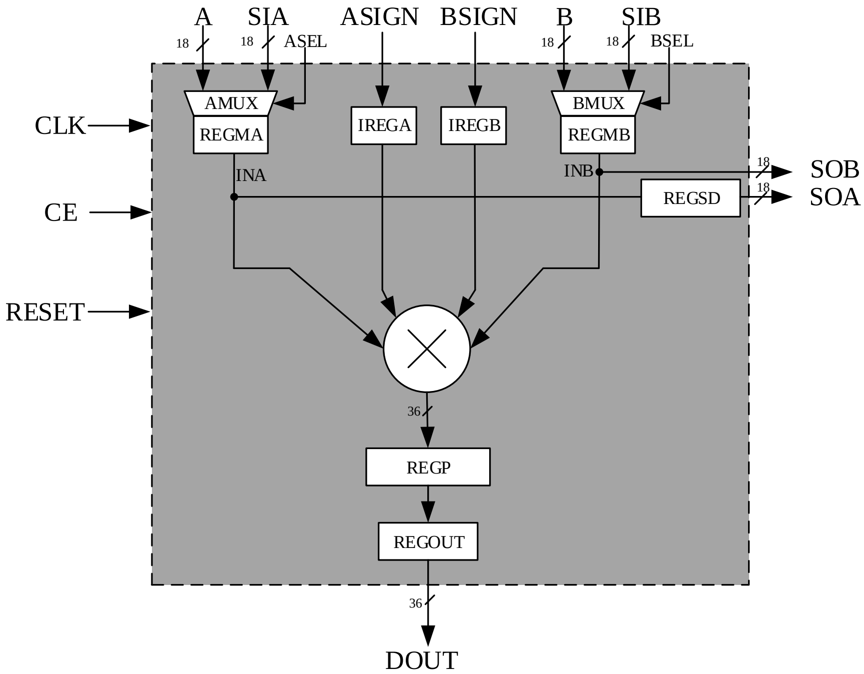
4.2.2 MULT18X18

Primitive Introduction

MULT18X18 supports 18-bit multiplication.

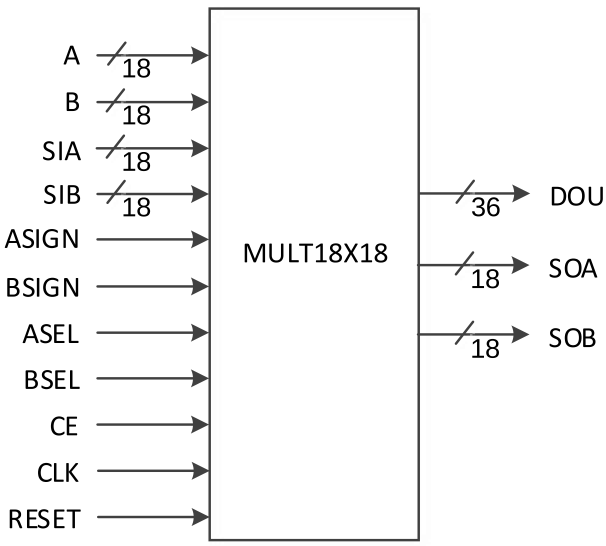
Logic Architecture Diagram

Figure 4-5 MULT18X18 Logic Architecture Diagram



Port Diagram

Figure 4-6 MULT18X18 Logic Architecture Diagram



Port Description

Table 4-5 MULT18X18 Port Description

Ports	I/O	Description
A[17:0]	Input	18-bit data input signal A
B[17:0]	Input	18-bit data input signal B
SIA[17:0]	Input	18-bit shift data input signal A
SIB[17:0]	Input	18-bit shift data input signal B
ASIGN	Input	A sign bit input signal
BSIGN	Input	B sign bit input signal
ASEL	Input	Source selection signal, SIA or A
BSEL	Input	Source selection signal, SIB or B
CLK	Input	Clock input signal
CE	Input	Clock enable signal, active-high
RESET	Input	Reset input signal, active-high
DOUT[35:0]	Output	Data output signal
SOA[17:0]	Output	Shift data output signal A
SOB[17:0]	Output	Shift data output signal B

Parameter Description

Table 4-6 MULT18X18 Parameter Description

Parameter	Range	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A (SIA or A) register 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B (SIB or B) register 1'b0: bypass mode 1'b1: registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register 1'b0: bypass mode 1'b1: registered mode
PIPE_REG	1'b0,1'b1	1'b0	Pipeline Register 1'b0: bypass mode 1'b1: registered mode

Parameter	Range	Default Value	Description
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN Input Register 1'b0: bypass mode 1'b1: registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN Input Register 1'b0: bypass mode 1'b1: registered mode
SOA_REG	1'b0,1'b1	1'b0	SOA register 1'b0: bypass mode 1'b1: registered mode
MULT_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	Reset mode configuration SYNC: synchronized reset ASYNC: asynchronous reset

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more details, you can refer to 5 IP Configuration.

Verilog Instantiation:

```
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 Instantiation:

```
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
```

);

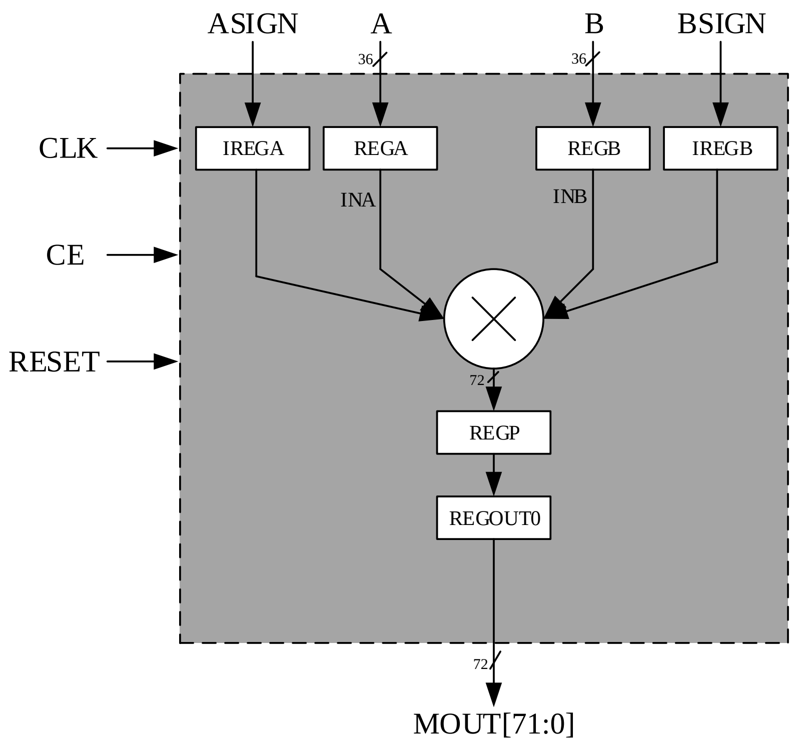
4.2.3 MULT36X36

Primitive Introduction

MULT36X36 supports 36-bit multiplication.

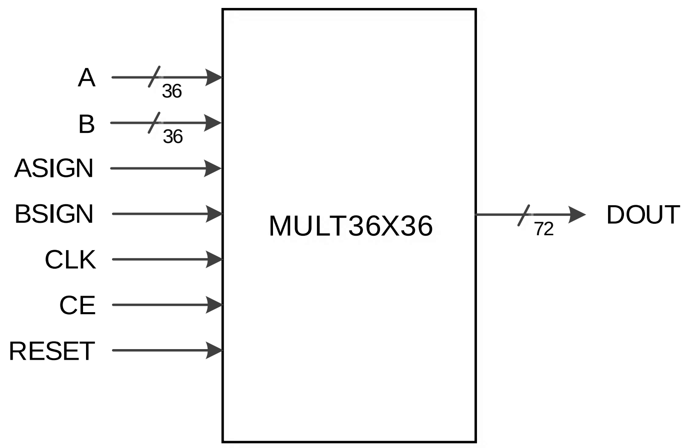
Logic Diagram

Figure 4-7 MULT36X36 Logic Diagram



Port Diagram

Figure 4-8 MULT36X36 Logic Diagram



Port Description

Table 4-7 MULT36X36 Port Description

Ports	I/O	Description
A[35:0]	Input	36-bit data input signal A
B[35:0]	Input	36-bit data input signal B
ASIGN	Input	A sign bit input signal
BSIGN	Input	B sign bit input signal
CLK	Input	Clock input signal
CE	Input	Clock enable signal, active-high
RESET	Input	Reset input signal, active-high
DOUT[71:0]	Output	Data output signal

Parameter Description

Table 4-8 MULT36X36 Parameter Description

Parameter	Range	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A register 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B register 1'b0: bypass mode 1'b1: registered mode
OUT0_REG	1'b0,1'b1	1'b0	Output0 register 1'b0: bypass mode 1'b1: registered mode
OUT1_REG	1'b0,1'b1	1'b0	Output1 register 1'b0: bypass mode 1'b1: registered mode
PIPE_REG	1'b0,1'b1	1'b0	Pipeline Register 1'b0: bypass mode 1'b1: registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN Input Register 1'b0: bypass mode 1'b1: registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN Input Register

Parameter	Range	Default Value	Description
			1'b0: bypass mode 1'b1: registered mode
MULT_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	Reset mode configuration SYNC: synchronized reset ASYNC: asynchronous reset

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Configuration.

Verilog Instantiation:

```
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 Instantiation:

```
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

The MULTALU mode implements a multiplier output by 54-bit ALU operation, including MULTALU36X18 and MULTALU18X18.

4.3.1 MULTALU36X18

Primitive Introduction

36x18 Multiplier with ALU (MULTALU36X18) is a 36x18 multiplier with ALU function.

MULTALU36X18 supports three arithmetic modes:

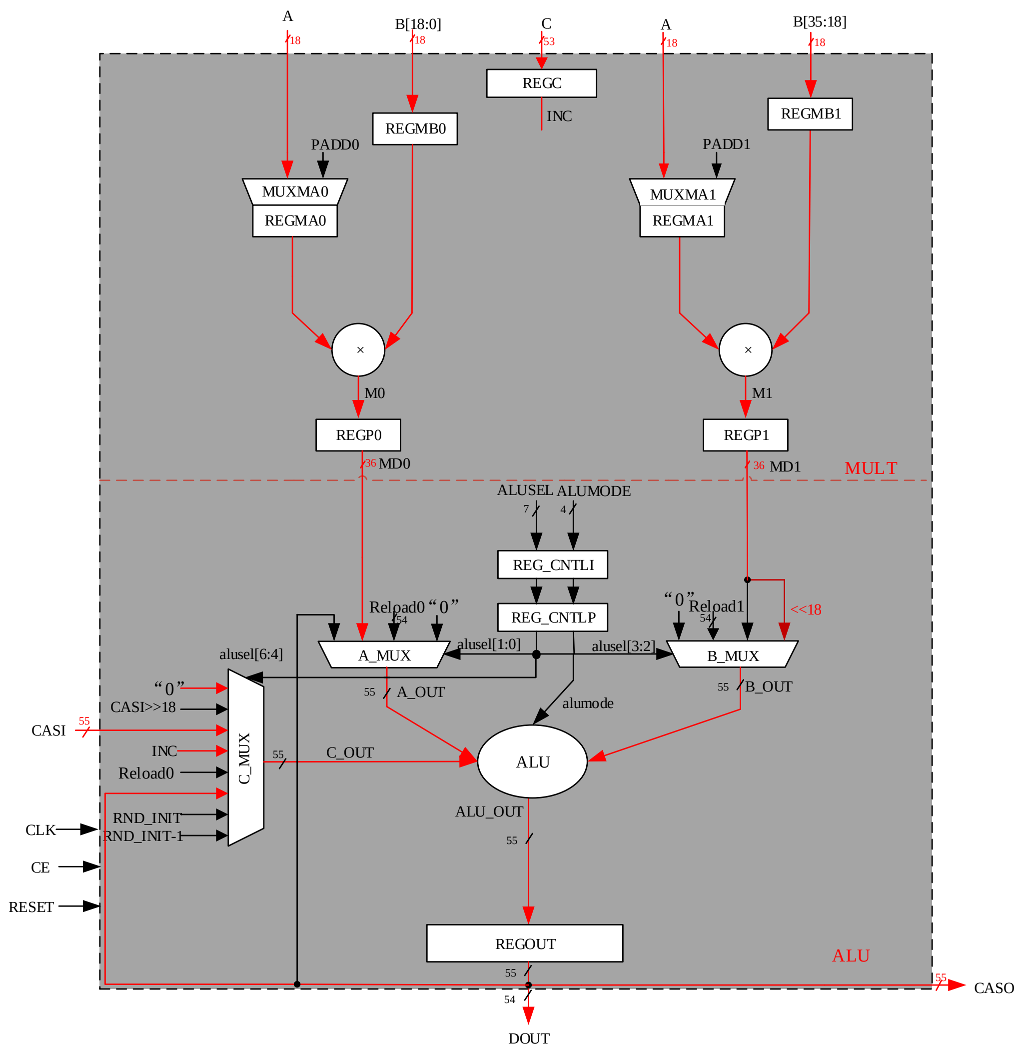
$$DOUT = A * B \pm C$$

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

$$DOUT = A * B + CASI$$

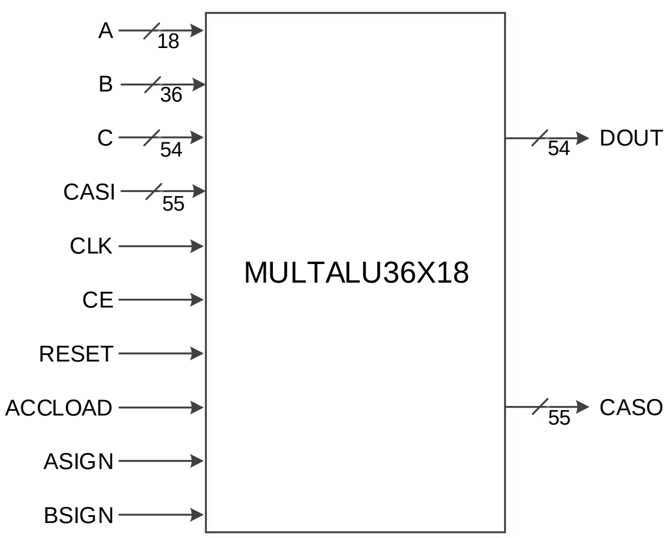
Logic Diagram

Figure 4-9 MULTALU36X18 Logic Diagram



Port Diagram

Figure 4-10 MULTALU36X18 Logic Diagram



Port Description

Table 4-9 MULTALU36X18 Port Description

Port	I/O	Description
A[17:0]	Input	18-bit data input signal A
B[35:0]	Input	36-bit data input signal B
C[53:0]	Input	54-bit reload data input signal
CASI[54:0]	Input	55-bit for cascade input signal
ASIGN	Input	A sign bit input signal
BSIGN	Input	B sign bit input signal
CLK	Input	Clock input signal
CE	Input	Clock enable signal, active-high
RESET	Input	Reset input signal, active-high
ACCLOAD	Input	Accumulator reload mode selection signal When the value is 0, reload 0; When the value is 1, accumulate it.
DOUT[53:0]	Output	Data output signal
CASO[54:0]	Output	55-bit for cascade output signal

Parameter Description

Table 4-10 MULTALU36X18 Parameter Description

Parameter	Range	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A register 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B register 1'b0: bypass mode 1'b1: registered mode
CREG	1'b0,1'b1	1'b0	Input C register 1'b0: bypass mode 1'b1: registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register 1'b0: bypass mode 1'b1: registered mode
PIPE_REG	1'b0,1'b1	1'b0	Pipeline Register

Parameter	Range	Default Value	Description
			1'b0: bypass mode 1'b1: registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN Input Register 1'b0: bypass mode 1'b1: registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN Input Register 1'b0: bypass mode 1'b1: registered mode
ACCLOAD_REG0	1'b0,1'b1	1'b0	ACCLOAD Output0 register 1'b0: bypass mode 1'b1: registered mode
ACCLOAD_REG1	1'b0,1'b1	1'b0	ACCLOAD Output1 register 1'b0: bypass mode 1'b1: registered mode
MULT_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	Reset mode configuratiom SYNC: synchronized reset ASYNC: asynchronous reset
MULTALU36X18_MODE	0,1, 2	0	MULTALU36X18 operation mode and input selection 0:36x18 +/- C; 1:ACC/0 + 36x18; 2:36x18 + CASI
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT plus/minus selection 1'b0: add 1'b1: sub

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Configuration.

Verilog Instantiation:

```
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 Instantiation:

```
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

Primitive Introduction

18x18 Multiplier with ALU (MULTALU18X18) is a 36x18 multiplier with ALU function.

MULTALU18X18 supports three arithmetic modes:

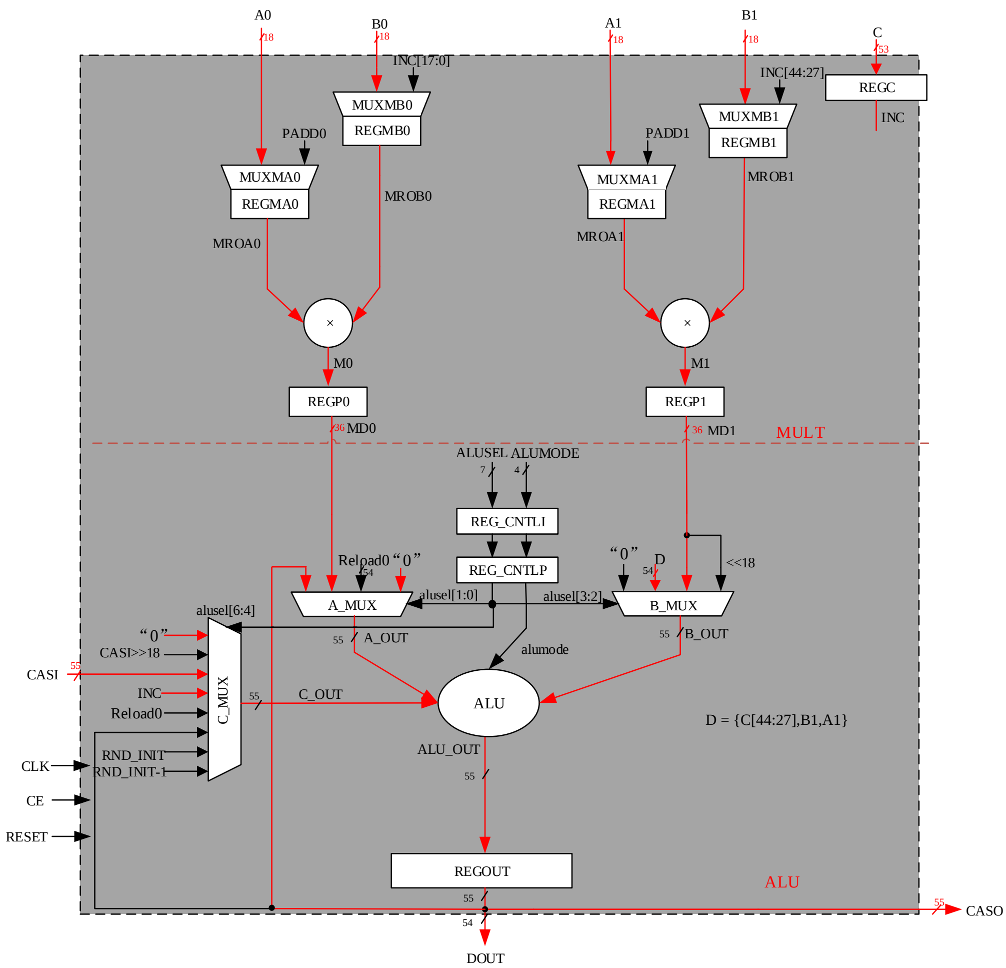
$$DOUT = \sum(A * B) \pm C$$

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

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

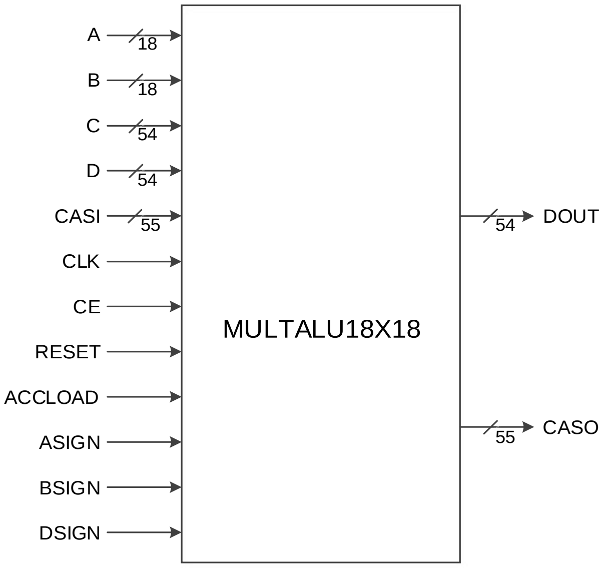
Logic Architecture Diagram

Figure 4-11 MULTALU18X18 Logic Diagram



Port Diagram

Figure 4-12 MULTALU18X18 Logic Diagram



Port Description

Table 4-11 MULTALU18X18 Logic Architecture Diagram

Ports	I/O	Description
A[17:0]	Input	18-bit data input signal A
B[17:0]	Input	18-bit data input signal B
C[53:0]	Input	54-bit data input signal C
D[53:0]	Input	54-bit data input signal D
CASI[54:0]	Input	55-bit for cascade input signal
ASIGN	Input	A sign bit input signal
BSIGN	Input	B sign bit input signal
DSIGN	Input	D sign bit input signal
CLK	Input	Clock input signal
CE	Input	Clock enable signal, active-high
RESET	Input	Reset input signal, active-high
ACCLOAD	Input	Accumulator reload mode selection signal When the value is 0, reload 0; When the value is 1, accumulate it.
DOUT[53:0]	Output	Data output signal
CASO[54:0]	Output	55-bit for cascade output signal

Parameter Description

Table 4-12 MULTALU18X18 Parameter Description

Parameter	Range	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A register 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B register 1'b0: bypass mode 1'b1: registered mode
CREG	1'b0,1'b1	1'b0	Input C register 1'b0: bypass mode 1'b1: registered mode
DREG	1'b0,1'b1	1'b0	Input D register 1'b0: bypass mode 1'b1: registered mode
DSIGN_REG	1'b0,1'b1	1'b0	DSIGN Input Register 1'b0: bypass mode 1'b1: registered mode
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN Input Register 1'b0: bypass mode 1'b1: registered mode
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN Input Register 1'b0: bypass mode 1'b1: registered mode
ACCLOAD_REG0	1'b0,1'b1	1'b0	ACCLOAD Output0 register 1'b0: bypass mode 1'b1: registered mode
ACCLOAD_REG1	1'b0,1'b1	1'b0	ACCLOAD Output1 register 1'b0: bypass mode 1'b1: registered mode
MULT_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	Reset mode configuratiom SYNC: synchronized reset ASYNC: asynchronous

Parameter	Range	Default Value	Description
			reset
PIPE_REG	1'b0,1'b1	1'b0	Pipeline Register 1'b0: bypass mode 1'b1: registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register 1'b0: bypass mode 1'b1: registered mode
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT plus/minus mode selection 1'b0: plus 1'b1: minus
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT plus/minus mode selection 1'b0: plus 1'b1: minus
MULTALU18X18_MODE	0,1, 2	0	MULTALU36X18 operation mode and input selection 0:ACC/0 +/- 18x18 +/- C; 1:ACC/0 +/- 18x18 + CASI; 2: 18x18 +/- D + CASI;

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Configuration.

Verilog Instantiation:

```
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 Instantiation:

```
     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

The MULTADDALU mode implements two 18 x 18 multipliers output by 54-bit ALU operation, corresponding to the primitive MULTADDALU18X18.

The MULTADDALU18X18 has three modes of operation：

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

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

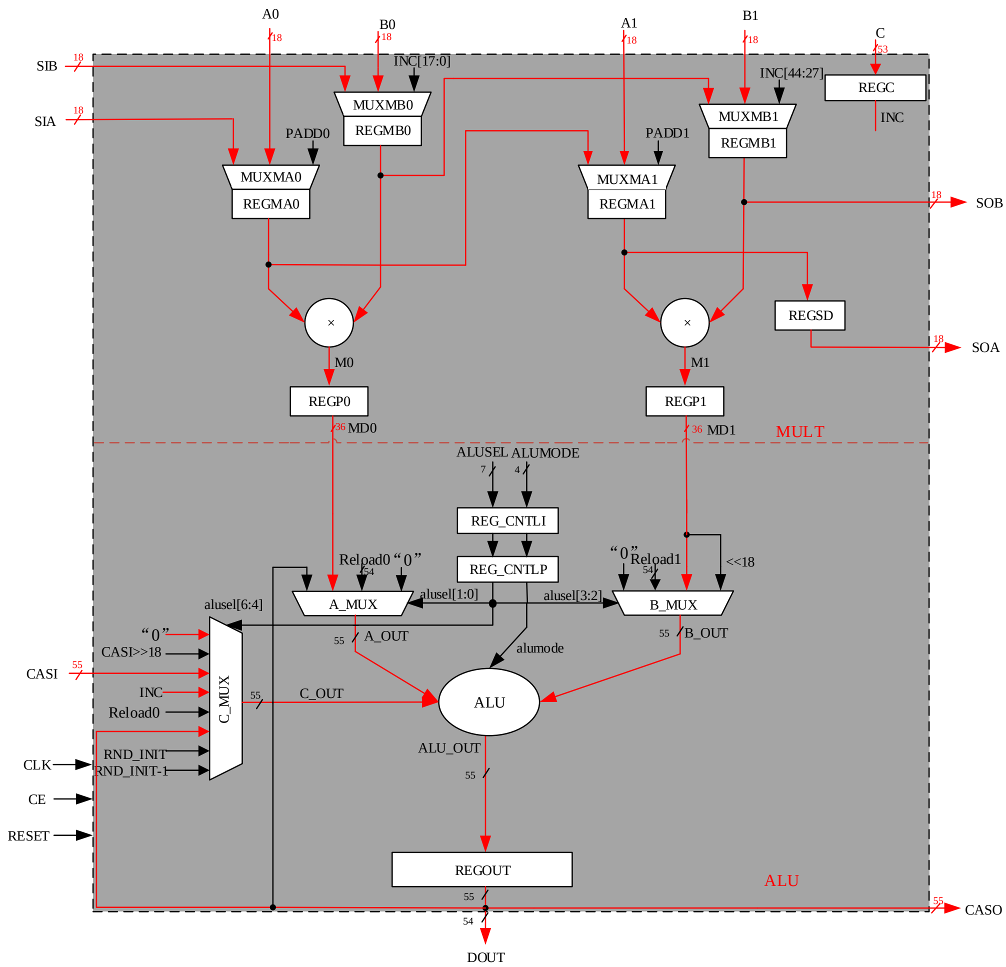
$$DOUT = A0 * B0 \pm A1 * B1 + CASI$$

Primitive Introduction

The Sum of Two 18x18 Multipliers with ALU (MULTADDALU18X18) is a 18x18 MAC with the function of ALU, which can be used to accumulate the sum of multiplication or reload.

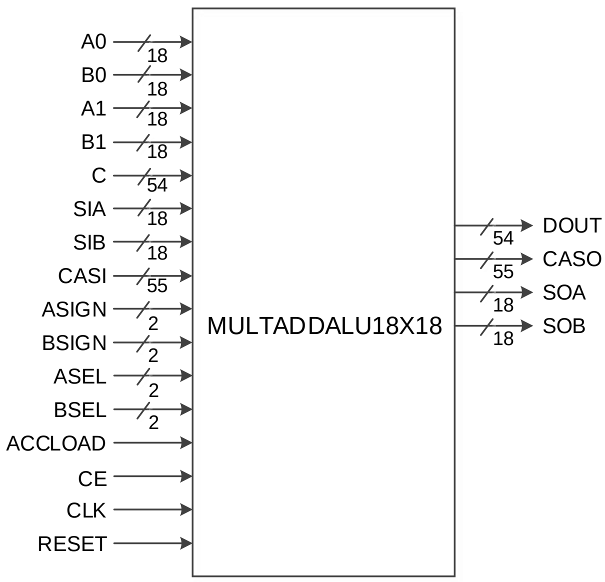
Logic Diagram

Figure 4-13 MULTADDALU18X18 Logic Diagram



Port Diagram

Figure 4-14 MULTADDALU18X18 Logic Diagram



Port Description

Table 4-13 MULTADDALU18X18 Port Description

Port	I/O	Description
A0[17:0]	Input	18-bit data input signal A0
B0[17:0]	Input	18-bit data input signal B0
A1[17:0]	Input	18-bit data input signal A1
B1[17:0]	Input	18-bit data input signal B1
C[53:0]	Input	54-bit reload data input signal C
SIA[17:0]	Input	18-bit shift data input signal A
SIB[17:0]	Input	18-bit shift data input signal B
CASI[54:0]	Input	55-bit for cascade input signal
ASIGN[1:0]	Input	A1, A0 sign bit input signal
BSIGN[1:0]	Input	B1, B0 sign bit input signal
ASEL[1:0]	Input	Input A1,A0 source selection signal
BSEL[1:0]	Input	Input A1,A0 source selection signal
CLK	Input	Clock input signal
CE	Input	Clock enable signal, active-high
RESET	Input	Reset input signal, active-high
ACCLOAD	Input	Accumulator reload mode selection signal When the value is 0, reload 0; When the value is 1, accumulate it.
DOUT[53:0]	Output	Data output signal
CASO[54:0]	Output	55-bit for cascade output signal
SOA[17:0]	Output	Shift data output signal A
SOB[17:0]	Output	Shift data output signal B

Parameter Description

Table 4-14 MULTADDALU18X18 Parameter Description

Parameter	Range	Default Value	Description
A0REG	1'b0,1'b1	1'b0	Input A0 (A0 or SIA) register 1'b0: bypass mode 1'b1: registered mode
A1REG	1'b0,1'b1	1'b0	Input A1 (A1 or Register Output A0) register 1'b0: bypass mode 1'b1: registered mode
B0REG	1'b0,1'b1	1'b0	Input B0 (B0 or SIB) register 1'b0: bypass mode 1'b1: registered mode
B1REG	1'b0,1'b1	1'b0	Input B1 (A1 or Register Output A0) register 1'b0: bypass mode 1'b1: registered mode
CREG	1'b0,1'b1	1'b0	Input C register 1'b0: bypass mode 1'b1: registered mode
PIPE0_REG	1'b0,1'b1	1'b0	Multiplier0 Pipeline Register 1'b0: bypass mode 1'b1: registered mode
PIPE1_REG	1'b0,1'b1	1'b0	Multiplier1 Pipeline Register 1'b0: bypass mode 1'b1: registered mode
OUT_REG	1'b0,1'b1	1'b0	Output register 1'b0: bypass mode 1'b1: registered mode
ASIGN0_REG	1'b0,1'b1	1'b0	ASIGN[0] Input Register 1'b0: bypass mode 1'b1: registered mode
ASIGN1_REG	1'b0,1'b1	1'b0	ASIGN[1] Input Register 1'b0: bypass mode 1'b1: registered mode
ACCLOAD_REG0	1'b0,1'b1	1'b0	ACCLOAD Output0 register

Parameter	Range	Default Value	Description
			1'b0: bypass mode 1'b1: registered mode
ACCLOAD_REG1	1'b0,1'b1	1'b0	ACCLOAD Output1 register 1'b0: bypass mode 1'b1: registered mode
BSIGN0_REG	1'b0,1'b1	1'b0	BSIGN[0] Input Register 1'b0: bypass mode 1'b1: registered mode
BSIGN1_REG	1'b0,1'b1	1'b0	BSIGN[1] Input Register 1'b0: bypass mode 1'b1: registered mode
SOA_REG	1'b0,1'b1	1'b0	SOA register 1'b0: bypass mode 1'b1: registered mode
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT plus/minus selection 1'b0: plus 1'b1: minus
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT plus/minus selection 1'b0: plus 1'b1: minus
MULTADDALU18X18_MODE	0,1, 2	0	MULTALU36X18 operation mode and input selection 0:18x18 +/- 18x18 +/- C; 1: ACC/0 + 18x18 +/- 18x18; 2:18x18 +/- 18x18 + CASI
MULT_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	Reset mode configuratiom SYNC: synchronized reset ASYNC: asynchronous reset

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Configuration.

Verilog Instantiation:

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 Instantiation:

```
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 Mode

PADD (pre-adder) performs the functions of pre-add, pre-subtract, and shifting. Each DSP macro unit includes two pre-adders to implement pre-add, pre-subtract, and shifting. PADD is located at the very front of the DSP macro unit and have two inputs, one parallel 18-bit input A or SIA and the other parallel 18-bit input B or SBI. To enhance the timing function, corresponding registers have been added to each input. Alternatively, it is possible to bypass the pre-adder so that input A and B act directly on the multiplier module. Gowin PADD can be used as a function block independently. PADD contains 9-bit PADD9 and 18-bit PADD18.

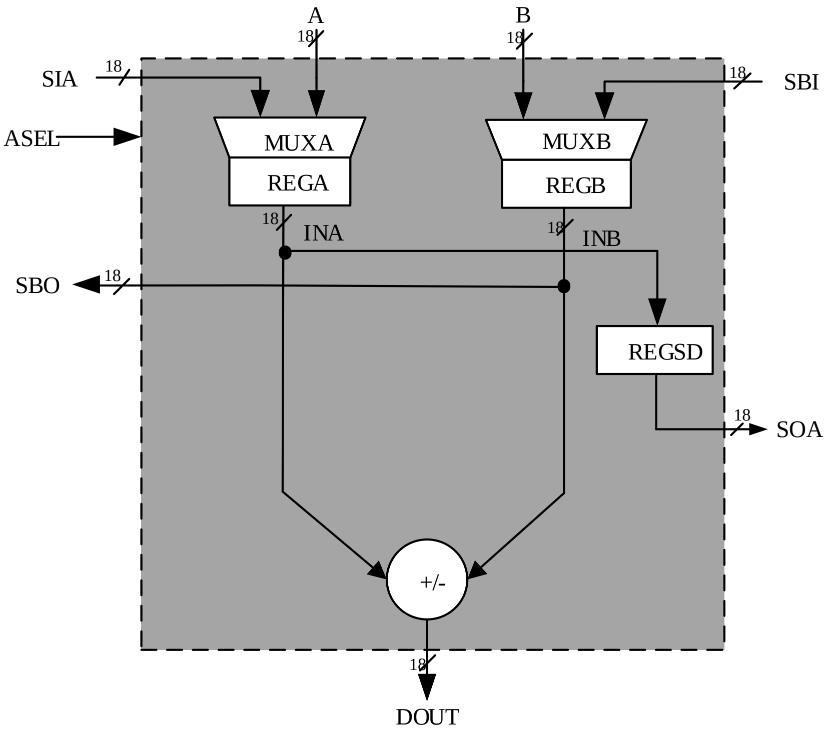
4.5.1 PADD18

Primitive Introduction

The 18-bit pre-adder (PADD18) is a 18-bit pre-adder that performs the function of pre-add, pre-subtract, or shifting.

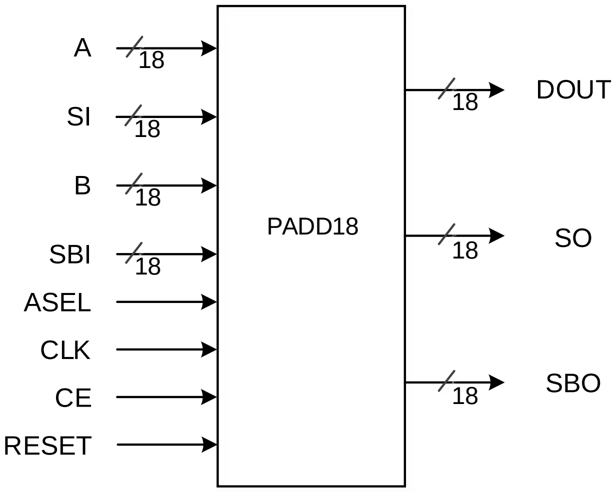
Logic Diagram

Figure 4-15 PADD18 Logic Diagram



Port Diagram

Figure 4-16 PADD18 Logic Diagram



Port Description

Table 4-15 PADD18 Port Diagram

Port	I/O	Description
A[17:0]	Input	18-bit data input signal A
B[17:0]	Input	18-bit data input signal B
SI[17:0]	Input	Shift data input signal A
SBI[17:0]	Input	Pre-adder shift input signal, reverse.
ASEL	Input	Source selection Input signal, SI or A
CLK	Input	Clock input signal
CE	Input	Clock enable signal, active-high
RESET	Input	Reset input signal, active-high
SO[17:0]	Output	Shift data output signal A
SBO[17:0]	Output	Pre-adder shift output signal, reverse.
DOUT[17:0]	Output	Data output signal

Parameter Description

Table 4-16 PADD18 Parameter Description

Parameter	Range	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A (A or SI) register 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B (B or SBI) register 1'b0: bypass mode 1'b1: registered mode
ADD_SUB	1'b0,1'b1	1'b0	plus/minus selection 1'b0: plus 1'b1: minus
PADD_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	Reset mode configuratiom SYNC: synchronized reset ASYNC: asynchronous reset
BSEL_MODE	1'b1,1'b0	1'b1	B input selection

Parameter	Range	Default Value	Description
			1'b1: SBI 1'b0: B
S or EG	1'b0,1'b1	1'b0	Shift output register 1'b0: bypass mode 1'b1: registered mode

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Configuration.

Verilog Instantiation:

```
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 Instantiation:

```
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
    );
```

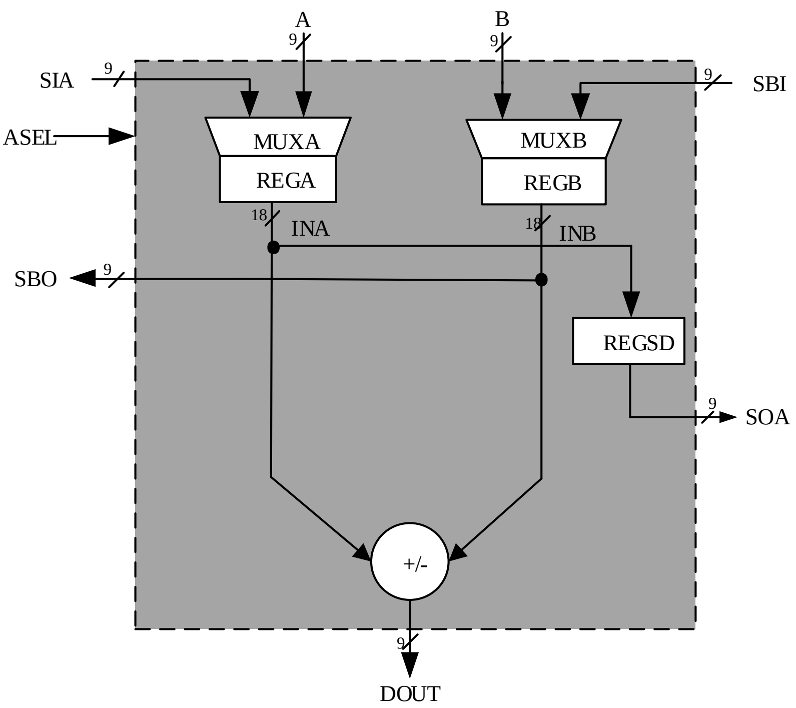
4.5.2 PADD9

Primitive Introduction

9-bit pre-adder (PADD9) is a 9-bit pre-adder that performs the function of pre-add, pre-subtract or shifting.

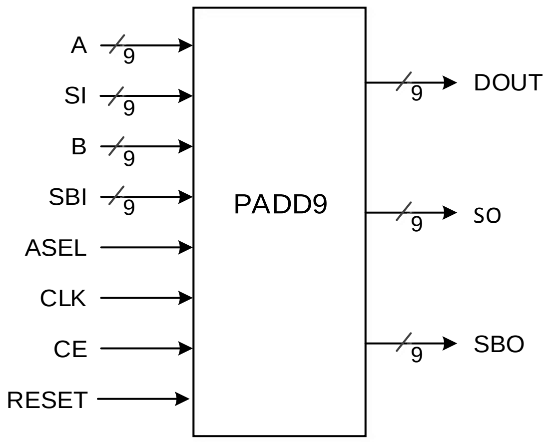
Logic Diagram

Figure 4-17 PADD9 Logic Diagram



Port Diagram

Figure 4-18 PADD9 Logic Diagram



Port Description

Table 4-17 PADD9 Port Diagram

Port	I/O	Description
A[8:0]	Input	9-bit data input signal A
B[8:0]	Input	9-bit data input signal B
SI[8:0]	Input	Shift data input signal A
SBI[8:0]	Input	Pre-adder shift input signal, reverse.
ASEL	Input	Source selection Input signal, SI or A
CLK	Input	Clock input signal
CE	Input	Clock enable signal, active-high
RESET	Input	Reset input signal, active-high
SO[8:0]	Output	Shift data output signal A
SBO[8:0]	Output	Pre-adder shift output signal, reverse.
DOUT[8:0]	Output	Data Output Signal

Parameter Description

Table 4-18 PADD9 Parameter Description

Parameter	Range	Default Value	Description
AREG	1'b0,1'b1	1'b0	Input A (A or SI) register 1'b0: bypass mode 1'b1: registered mode
BREG	1'b0,1'b1	1'b0	Input B (B or SBI) register 1'b0: bypass mode 1'b1: registered mode
ADD_SUB	1'b0,1'b1	1'b0	plus/minus selection 1'b0: plus 1'b1: minus
PADD_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	Reset mode configuration SYNC: synchronized reset ASYNC: asynchronous reset
BSEL_MODE	1'b1,1'b0	1'b1	Input B selection 1'b1: SBI 1'b0: B
SOREG	1'b0,1'b1	1'b0	Shift output register 1'b0: bypass mode 1'b1: registered mode

Primitive Instantiation

The primitive can be instantiated directly, or generated by the IP Core Generator tool. For more information, you can refer to 5 IP Configuration.

Verilog Instantiation:

```
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 Instantiation:

```
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 Configuration

There are five types of primitives of DSP module in IP Core Generator: ALU54, MULT, MULTADDALU, MULTALU and PADD.

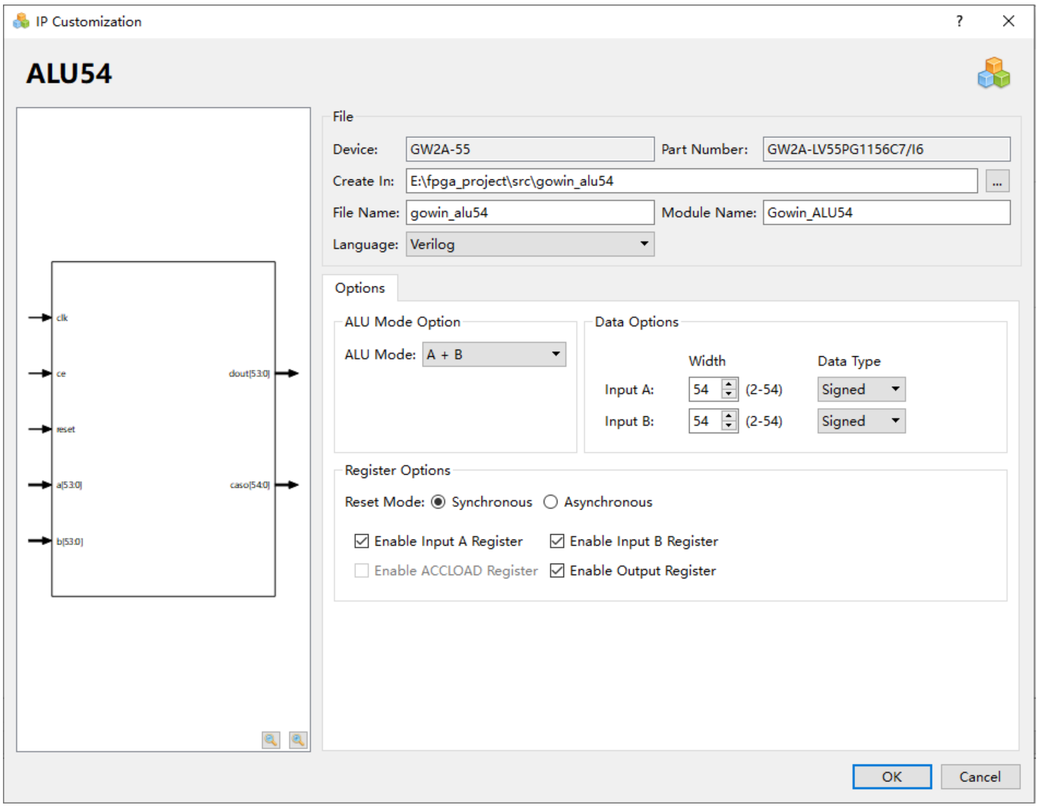
5.1 ALU54

ALU54 can be used to implement 54-bit arithmetic and logic operation. Click "ALU54" on the IP Core Generator, and a brief introduction to "ALU54" will be displayed.

IP Configuration

On the IP Core Generator interface, double-click "ALU54" to open the "IP Customization" window, as shown in Figure 5-1. This includes "File", "Options", and ports diagram.

Figure 5-1 IP Customization of ALU54



1. File Configuration

The File Configuration is used to configure information about the resulting IP design file.

- Device: Displays information about the configured Device.
- Part Number: Display the configured Part Number.
- Language: Hardware description language used to generate the IP design files. Click the drop-down list to select the language, including Verilog and VHDL.
- Module Name: The module name of the generated IP design files. Enter the module name in the text box. Module name cannot be the same as the primitive name. If it is the same, an error will be reported.
- File Name: The name of the generated IP design files. Enter the file name in the text box.
- Create In: The path on which the generated IP files will be stored. Enter the target path in the box or select the target path by clicking the option.

2. Options

The Options is used to configure ALU54 by users, as shown in .

- ALU Mode Option: Allows you to select the operation modes. The MULTADDALU can be configured in the following operation modes:
  - 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: Allows you to configure data.
  - Configure ALU54 input data width. The data width of input port A/B can be configured as 1-54 bits.
  - Output width adjusts automatically according to the input width.
  - Data Type: Can be configured as signed or unsigned.
- Register Options: Allows you to configure registers operation mode.
  - "Reset Mode" option configures the reset mode of the ALU54, which supports the synchronous mode "Synchronous" and the asynchronous mode "Asynchronous".
  - Enable Input A Register: Allows you to enable or disable input A register.
  - Enable Input B Register: Allows you to enable or disable input B register.
  - Enable ACCLOAD Register: Allows you to enable or disable ACCLOAD register.
  - Enable Output Register: Allows you to enable or disable Output register.

3. Ports Diagram

The ports diagram is based on the current IP Core configuration. The input/output bit-width updates in real time based on the "Options" configuration, as shown in Figure 5-1.

IP Generation Files

After configuration, it will generate three files that are named after the "File Name".

- "gowin_alu54.v" file is a complete Verilog module to generate instance ROM16, and it is generated according to the IP configuration.
- "gowin_alu54_tmp.v" is the instance template file.
- "gowin_alu54.ipc" file is IP configuration file. You can load the file to configure the IP.

Note!

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

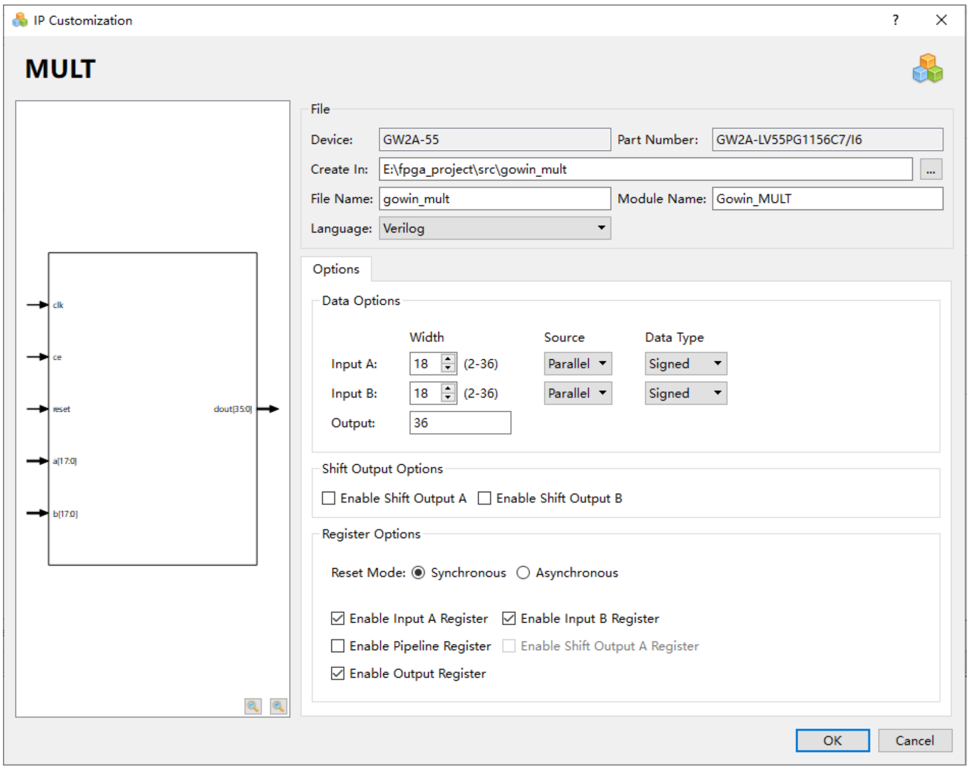
5.2 MULT

MULT can be configured as an multiplier. Click "MULT" on the IP Core Generator, and a brief introduction to the MULT will be displayed.

IP Configuration

Double-click "MULT" to open the "IP Customization" window, as shown in Figure 5-2 . This includes "File", "Options", and ports diagram.

Figure 5-2 IP Customization of MULT



1. File Configuration
   - The File Configuration is used to configure the generated IP design file.
   - The MULT file configuration is similar to that of ALU54. For the details, please refer to 5.1 ALU54 > File Configuration.
2. Options Configuration
   - The Options Configuration is used to configure IP, as shown in .
   - Data Options: Allows you to configure data.
     - The maximum data width of the input ports (Input A Width/ Input B Width) is 36;
     - The Output Width will automatically adjust according to the input bit width.
     - Input port A/B can be set as Parallel, Shift.
     - The data type can be configured as Unsigned or Signed.
   - Shift Output Options: Allows you to select whether to enable shift out. This option can be checked when both Input A and Input B width are less than or equal to 18.

   Note!

   If either Input A width or Input B width is greater than 18, the Shift Output Options will be grayed.

   - Register Options: The function and operatiING of the register options are the same as that of ALU54. Please refer to the Options Configuration in 5.1ALU54 for details.
3. Ports Diagram

   The ports diagram is based on the current IP Core configuration. The input/output number of bit-width updates in real time based on the "Options" configuration, as shown in ;

IP Generation Files

After configuration, it will generate three files that are named after the "File Name".

- "gowin_mult.v" file is a complete Verilog module to generate instance MULT, and it is generated according to the IP configuration;
- "gowin_mult_tmp.v" is the instance template file;
- "gowin_rpll.ipc" file is IP configuration file. You can load the file to configure the IP.

Note!

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

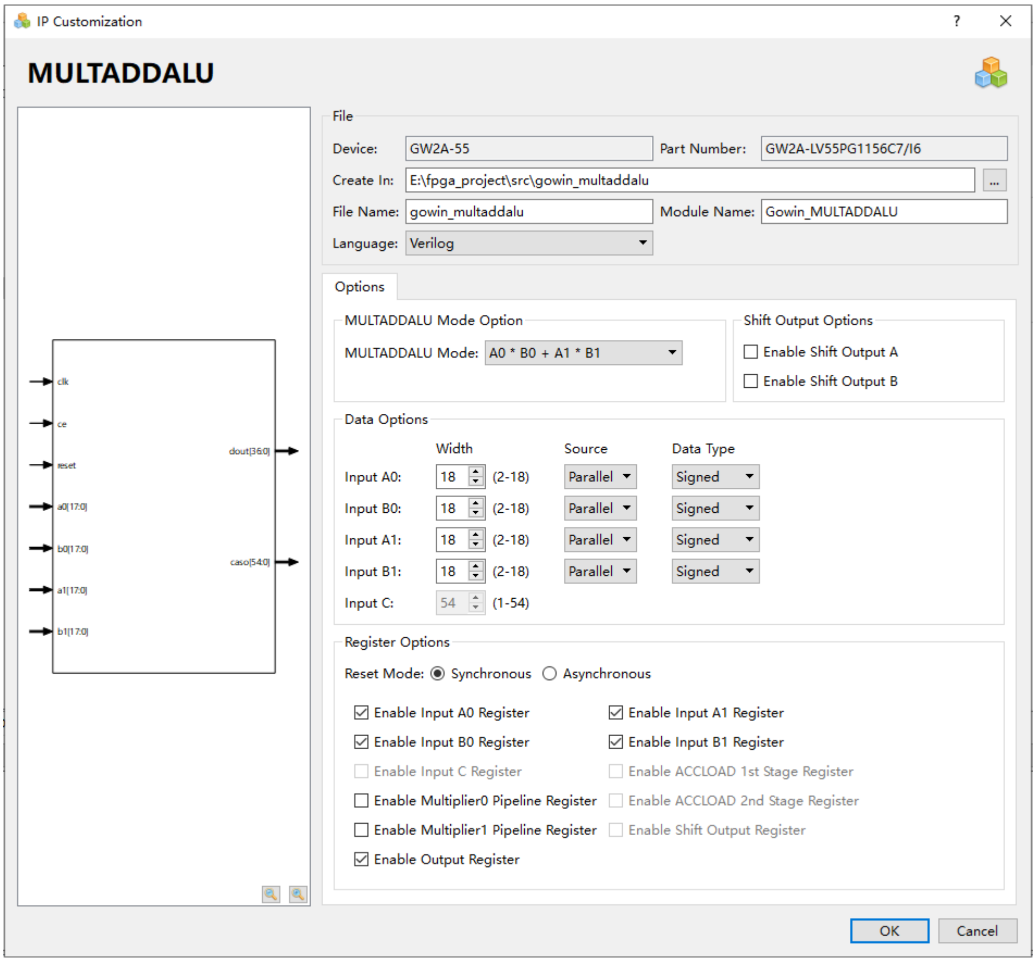
5.3 MULTADDALU

MULTADDALU implements the multiplier quadratic summation or accumulation function. Click "MULT" on the IP Core Generator, and a brief introduction to the MULTADDALU will be displayed.

IP Configuration

Double-click the "MULTADDALU" to open the "IP Customization" window. This includes the "File", "Options", and ports diagram, as shown in Figure 5-3.

Figure 5-3 IP Customization of MULTADDALU



1. File Configuration

- The File Configuration is used to configure the generated IP design file.
- The MULTADDALU file configuration is similar to that of ALU54.

For the details, please refer to 5.1 ALU54 > File Configuration.

2. Options Configuration

The Options Configuration is used to configure IP, as shown in .

- MULTADDALU Mode Option: Allows you to select the operation modes. The MULTADDALU can be configured as following operation modes:
  - 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
- The configuration of MULTADDALU Data Options and Register Options is similar to that of MULT. For the details, please refer to 5.2 MULT.

3. Ports Diagram

The ports diagram is based on the current IP Core configuration. The input/output bit-width updates in real time based on the "Data Options" and "Register Options" configuration, as shown in ;

IP Generation Files

After configuration, it will generate three files that are named after the "File Name".

- "gowin_multaddalu.v" file is a complete Verilog module to generate instance MULTADDALU, and it is generated according to the IP configuration.
- "gowin_multaddalu_tmp.v" is the instance enable.
- "gowin_multaddalu.ipc" file is IP configuration file. You can load the file to configure the IP.

Note!

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

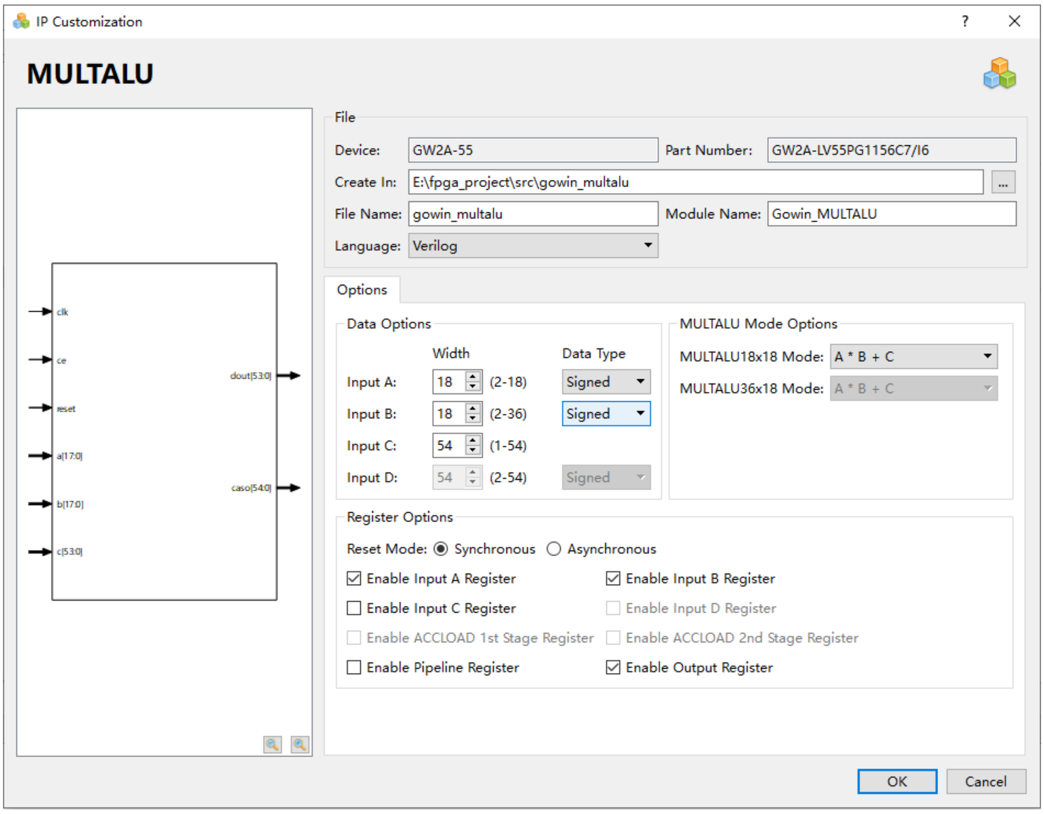
5.4 MULTALU

MULTALU can implement the Multiplier ALU mode. Click "MULTALU" on the IP Core Generator, and a brief introduction to the MULTALU will be displayed.

IP Configuration

Double-click "MULTALU" to open the "IP Customization" window. This includes the "File", "Options", and ports diagram, as shown in Figure 5-4.

Figure 5-4 IP Customization of MULTALU



1. File Configuration

   - The File Configuration is used to configure the generated IP design file.
   - The MULTALU file configuration is similar to that of ALU54. For the detailed configuration instructions, please refer to 5.1 ALU54 > File Configuration.

2. Options Configuration

   The Options Configuration is used to configure IP, as shown in .

   - MULTALU Mode Option

MULTALU can generate two modules according to the input port width: MULTALU36X18 or MULTALU18X18. When the Input A width and Input B width are less than or equal to 18, the MULTALU36X18 mode will be greyed out, and MULTALU18X18 mode can be configured as:

- 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

- When the width of Input B is greater than 18 bits, the MULTALU18X18 Mode is grayed out
- MULTALU36X18 Mode can be configured as follows:
  - A*B + C
  - A*B - C
  - Accum + A*B
  - A*B + CASI
- The configuration of the MULTALU Data Options and Register Options is similar to that of MULT. For the details, please refer to 5.2 MULT.

3. Ports Diagram

The ports diagram is based on the current IP Core configuration. The input/output bit-width updates in real time based on the "Options" configuration, as shown in Figure 5-4.

IP Generation Files

After configuration, it will generate three files that are named after the "File Name".

- "gowin_multtalu.v" file is a complete Verilog module to generate instance MULTALU, and it is generated according to the IP configuration;

- "gowin_multtalu_tmp.v" is the instance template file;
- "gowin_multtalu.ipc" file is IP configuration file. You can load the file to configure the IP.

Note!

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

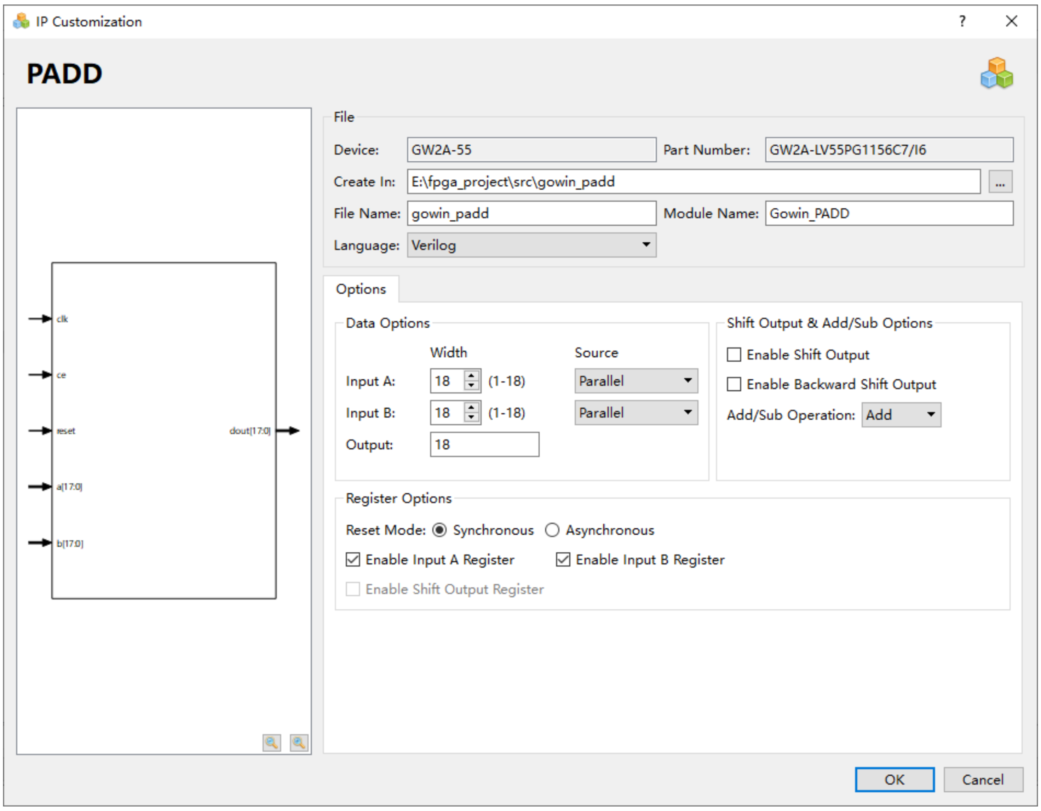
5.5 PADD

PADD can be configured as a pre-add, pre-subtract, or shifting. Click "PADD" on the IP Core Generator, and a brief introduction to the PADD will be displayed.

IP Configuration

Double-click the "PADD" to open the "IP Customization" window. This includes the "File", "Options", and ports diagram, as shown in Figure 5-5.

Figure 5-5 IP Customization of PADD



1. File Configuration
   - The File Configuration is used to configure the generated IP design file.

- The PADD file configuration is similar to that of ALU54. For the detailed configuration instructions, please refer to 5.1 ALU54 > File Configuration.

2. Options Configuration

The Options Configuration is used to configure IP, as shown in Figure 5-5.

- Data Options: Allows you to configure data.
  - The maximum data width of the input ports (Input A Width/ Input B Width) is 18;
  - The output width automatically adjusts according to the input width, and the width determines whether PADD9 or PADD18 are generated during instance.
  - Input A Source: you can select Parallel A or Shift;
  - Input B Source: you can select Parallel or Backward Shift.
- Shift Output and Add/Sub Options: Allows you to enable or disable Shift Output, Backward Shift Output, and add/sub operation.
  - Check "Enable Shift Output" to enable shift output;
  - Check "Enable Backward Shift Output" to enable backward shift output;
  - Configure "Add/Sub Operation" to perform add/sub operation.
- Register Options: Allows you to configure registers operation mode.
  - Reset Mode: Sets whether the reset mode is synchronous or asynchronous;
  - Enable Input A Register: Allows you to enable or disable Input A register;
  - Enable Input B Register: Allows you to enable or disable Input B register;
  - Enable Output Register: Allows you to enable or disable Output register.

3. Ports Diagram

The ports diagram is based on the current IP Core configuration. The input/output number of bit-width updates in real time based on the "Options" configuration, as shown in ;

IP Generation Files

After configuration, it will generate three files that are named after the "File Name".

- "gowin_padd.v" file is a complete Verilog module to generate instance PADD, and it is generated according to the IP configuration;
- "gowin_padd_tmp.v" is the instance template file;
- "gowin_padd.ipc" file is IP configuration file. You can load the file to configure the IP.

Note!

If VHDL is selected as the hardware description language, the first two files will be named with .vhd suffix.

GOWIN

PROGRAMMING FOR THE FUTURE