



Gowin 数字信号处理器 (DSP) 用户指南

UG287-1.2,2020-08-18

版权所有©2020 广东高云半导体科技股份有限公司

未经本公司书面许可，任何单位和个人不得擅自摘抄、复制、翻译本档内容的部分或全部，并不得以任何形式传播。

免责声明

本档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外，高云半导体概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和 / 或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。高云半导体对档中包含的文字、图片及其它内容的准确性和完整性不承担任何法律或非法律责任，高云半导体保留修改档中任何内容的权利，恕不另行通知。高云半导体不承诺对这些档进行适时的更新。

版本信息

日期	版本	说明
2016/05/16	1.05	初始版本。
2016/07/04	1.06	修改 PADD18 的结构框图。
2016/07/11	1.07	标准化插图。
2016/08/16	1.08	修改 GW2A-18 器件的乘法器数目。
2016/11/08	1.09	修改乘法器框图。
2017/10/09	1.10	根据新原语修改相关内容。
2020/08/18	1.2	<ul style="list-style-type: none">● 修改章节结构；● 优化第五章“IP 调用”内容。

目录

目录	iv
图目录	v
表目录	vi
1 关于本手册	1
1.1 手册内容	1
1.2 术语、缩略语	1
1.3 技术支持与反馈	1
2 概述	2
3 DSP 结构	3
4 DSP 原语	7
4.1 ALU54	7
4.2 MULT	10
4.2.1 MULT9X9	13
4.2.2 MULT18X18	15
4.2.3 MULT36X36	25
4.3 MULTALU	27
4.3.1 MULTALU36X18	29
4.3.2 MULTALU18X18	36
4.4 MULTADDALU	41
4.5 PADD 模式	50
4.5.1 PADD18	56
4.5.2 PADD9	58
5 IP 调用	66
5.1 ALU54	66
5.2 MULT	69
5.3 MULTADDALU	72
5.4 MULTALU	74
5.5 PADD	77

图目录

图 3-1 宏单元的组成结构.....	4
图 4-1 ALU54D 逻辑结构示意图.....	7
图 4-2 ALU54D 端口示意图.....	8
图 4-3 MULT9X9 逻辑结构示意图.....	13
图 4-4 MULT9X9 端口示意图.....	14
图 4-5 MULT18X18 逻辑结构示意图.....	19
图 4-6 MULT18X18 端口示意图.....	20
图 4-7 MULT36X36 逻辑结构示意图.....	25
图 4-8 MULT36X36 端口示意图.....	25
图 4-9 MULTALU36X18 逻辑结构示意图.....	30
图 4-10 MULTALU36X18 端口示意图.....	31
图 4-11 MULTALU18X18 逻辑结构示意图.....	37
图 4-12 MULTALU18X18 端口示意图.....	38
图 4-13 MULTADDALU18X18 逻辑结构示意图.....	46
图 4-14 MULTADDALU18X18 端口示意图.....	47
图 4-15 PADD18 逻辑结构示意图.....	56
图 4-16 PADD18 端口示意图.....	56
图 4-17 PADD9 逻辑结构示意图.....	61
图 4-18 PADD9 端口示意图.....	61
图 5-1 ALU54 的 IP Customization 窗口结构.....	67
图 5-2 MULT 的 IP Customization 窗口结构.....	70
图 5-3 MULTADDALU 的 IP Customization 窗口结构.....	72
图 5-4 MULTALU 的 IP Customization 窗口结构.....	75
图 5-5 PADD 的 IP Customization 窗口结构.....	78

表目录

表 1-1 术语、缩略语	1
表 3-1 DSP 模块端口描述及说明	5
表 3-2 DSP 模块内部寄存器描述	6
表 4-1 ALU54D 端口介绍	8
表 4-2 ALU54D 参数介绍	9
表 4-3 MULT9X9 端口介绍	14
表 4-4 MULT9X9 参数介绍	15
表 4-5 MULT18X18 端口介绍	20
表 4-6 MULT18X18 参数介绍	21
表 4-7 MULT36X36 端口介绍	26
表 4-8 MULT36X36 参数介绍	26
表 4-9 MULTALU36X18 端口介绍	31
表 4-10 MULTALU36X18 参数介绍	32
表 4-11 MULTALU18X18 端口介绍	39
表 4-12 MULTALU18X18 参数介绍	40
表 4-13 MULTADDALU18X18 端口介绍	48
表 4-14 MULTADDALU18X18 参数介绍	49
表 4-15 PADD18 端口介绍	57
表 4-16 PADD18 参数介绍	57
表 4-17 PADD9 端口介绍	62
表 4-18 PADD9 参数介绍	62

1 关于本手册

1.1 手册内容

本手册主要描述高云数字信号处理器（DSP）资源的结构、信号定义及用户调用方法等内容，旨在帮助用户快速熟悉高云 DSP 的使用流程，提高设计效率。

1.2 术语、缩略语

表 1-1 中列出了本手册中出现的相关术语、缩略语及相关释义。

表 1-1 术语、缩略语

术语、缩略语	全称	含义
DSP	Digital Signal Processor	数字信号处理器
FIR	Finite Impulse Response	有限脉冲响应滤波器
FFT	Fast Fourier Transformation	快速傅里叶变换
CFU	Configurable Function Unit	可配置功能单元
MULT	Multiplier	乘法器
PADD	Pre-adder	预加器
ALU54	54-bit Arithmetic Logic Unit	54 位算术逻辑单元

1.3 技术支持与反馈

高云半导体提供全方位技术支持，在使用过程中如有任何疑问或建议，可直接与公司联系：

网址：www.gowinsemi.com

E-mail：support@gowinsemi.com

Tel: +86 755 8262 0391

2 概述

高云半导体 FPGA 产品具有丰富的 DSP 资源，可满足用户对高性能数字信号的处理需求，如 FIR 和 FFT 的设计等。DSP 模块具有时序性能稳定、资源利用率高和功耗低等优点。本手册旨在帮助用户快速了解 DSP 的结构和使用方法。

DSP 模块的功能及特性如下：

- 3 种宽度（9-bit，18-bit，36-bit）的乘法器
- 54-bit 的算术/逻辑运算单元
- 多个乘法器可级联以增加数据宽度
- 桶形移位器
- 通过反馈信号做自适应滤波
- 支持寄存器的流水线和旁路功能

注！

GW1N-1、GW1N-1S、GW1NR-1、GW1NS-2、GW1NS-2C、GW1NSE-2C、GW1NSR-2、GW1NSR-2C、GW1NZ-1 器件，不支持 DSP。

3 DSP 结构

高云半导体 FPGA 产品的 DSP 模块以行的形式分布在 FPGA 阵列中，DSP 模块由两个宏单元组成，宏单元包含两个预加器（Pre-adder）、两个 18-bit 的乘法器（MULT18X18）和三输入算术/逻辑运算单元（ALU54），宏单元的结构组成如图 3-1 所示。

DSP 模块端口描述及含义说明如表 3-1 所示。内部寄存器如表 3-2 所示。此外，输入信号 CLK，CE 和 RESET 用于控制寄存器。

表 3-1 DSP 模块端口描述及说明

端口名称	I/O 类型	说明
A0[17:0]	I	18-bit 数据输入 A0
B0[17:0]	I	18-bit 数据输入 B0
A1[17:0]	I	18-bit 数据输入 A1
B1[17:0]	I	18-bit 数据输入 B1
C[53:0]	I	54-bit 数据输入 C
SIA[17:0]	I	移位数据输入 A，用于级联连接。输入信号 SIA 直接连接到先前相邻的 DSP 模块的输出信号 SOA
SIB[17:0]	I	移位数据输入 B，用于级联连接。输入信号 SIB 直接连接到先前相邻的 DSP 模块的输出信号 SOB
SBI[17:0]	I	预加器逻辑移位输入，反向
CASI[54:0]	I	来自前一个 DSP 模块的 CASO，ALU 级联输入，用于级联连接
ASEL[1:0]	I	预加器或乘法器的 A 输入源选择
BSEL[1:0]	I	乘法器的 B 输入源选择
ASIGN[1:0]	I	输入信号 A 符号位
BSIGN[1:0]	I	输入信号 B 符号位
PADDSUB[1:0]	I	预加器的操作控制信号，用于预加器逻辑加减法选择
CLK[3:0]	I	时钟输入
CE[3:0]	I	时钟使能信号
RESET[3:0]	I	复位信号，支持同步/异步模式
SOA[17:0]	O	移位数据输出 A
SOB[17:0]	O	移位数据输出 B
SBO[17:0]	O	预加器逻辑移位输出，反向
DOUT[35:0]	O	DSP 输出数据
CASO[54:0]	O	ALU 输出到下一个 DSP 模块进行级联连接，最高位符号扩展

表 3-2 DSP 模块内部寄存器描述

寄存器	说明及相关属性
REGA0	A0输入寄存器
REGA1	A1输入寄存器
REGB0	B0输入寄存器
REGB1	B1输入寄存器
REGC	C输入寄存器
REGMA0	左乘数A0输入寄存器
REGMA1	右乘数A1输入寄存器
REGMB0	左乘数B0输入寄存器
REGMB1	右乘数B1输入寄存器
REGP0	左乘法器流水线输出寄存器
REGP1	右乘法器流水线输出寄存器
REGOUT	DOUT输出寄存器
REG_CNTL1	控制信号第一级寄存器
REG_CNTL2	控制信号第二级寄存器
REGSD	SOA 的移位输出寄存器

4 DSP 原语

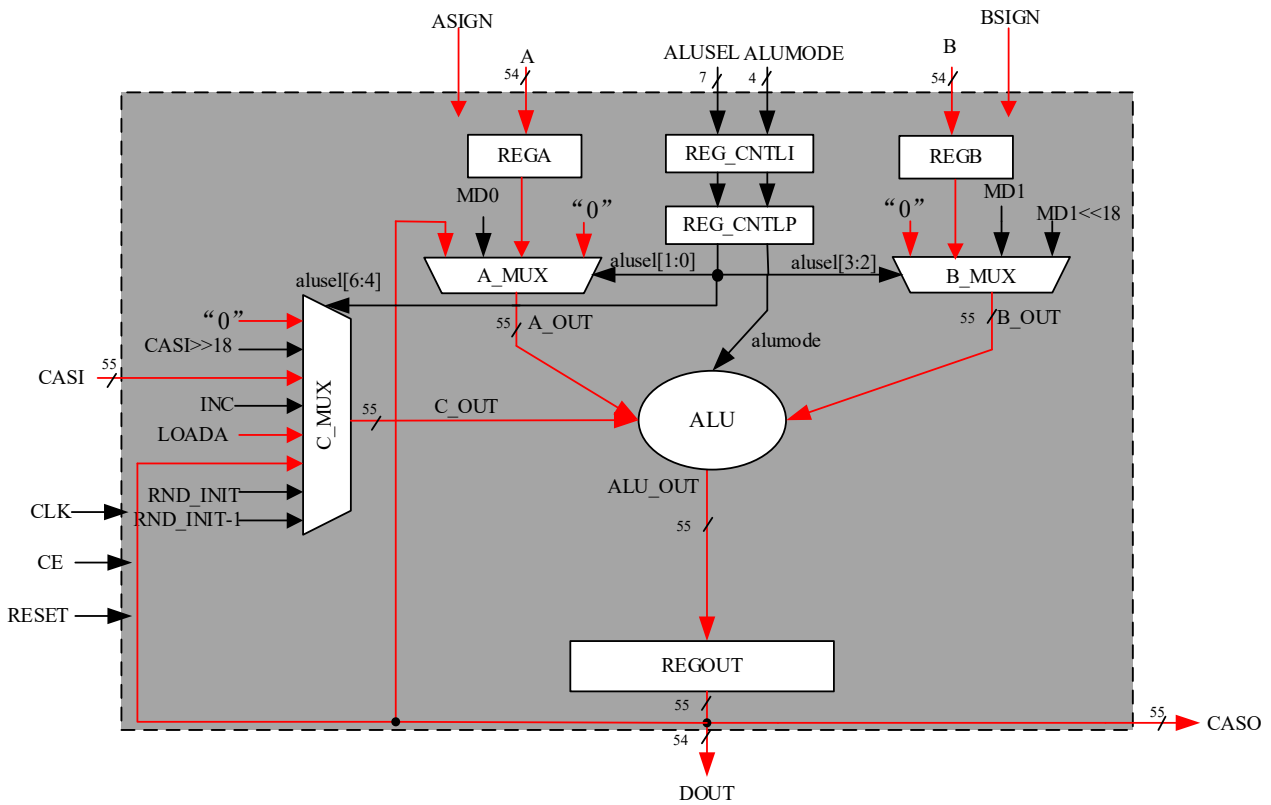
4.1 ALU54

原语介绍

ALU54D (54-bit Arithmetic Logic Unit) 是 54 位算术逻辑单元, 实现 54 位的算术逻辑运算。

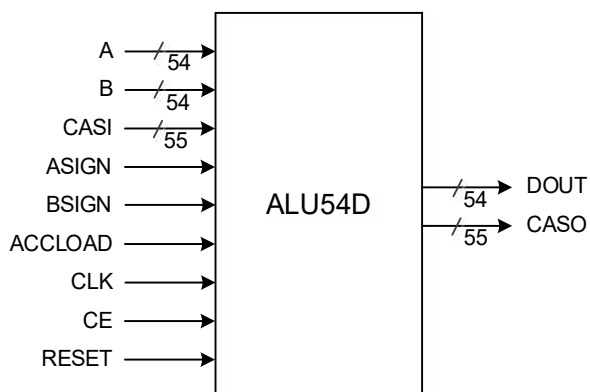
逻辑结构示意图

图 4-1 ALU54D 逻辑结构示意图



端口示意图

图 4-2 ALU54D 端口示意图



端口介绍

表 4-1 ALU54D 端口介绍

端口	I/O	描述
A[53:0]	Input	54-bit 数据输入信号 A
B[53:0]	Input	54-bit 数据输入信号 B
CASI[54:0]	Input	55-bit 级联输入信号
ASIGN	Input	A 符号位输入信号
BSIGN	Input	B 符号位输入信号
ACCLOAD	Input	累加器 Reload 模式选择信号
CLK	Input	时钟输入信号
CE	Input	时钟使能信号
RESET	Input	复位输入信号
DOUT[53:0]	Output	ALU54D 数据输出信号
CASO[54:0]	Output	55-bit 级联输出信号

参数介绍

表 4-2 ALU54D 参数介绍

参数	范围	默认	描述
AREG	1'b0,1'b1	1'b0	输入 A 寄存器 1'b0:旁路模式 1'b1:寄存器模式
BREG	1'b0,1'b1	1'b0	输入 B 寄存器 1'b0:旁路模式 1'b1:寄存器模式
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
ACCLOAD_REG	1'b0,1'b1	1'b0	ACCLOAD 寄存器 1'b0:旁路模式 1'b1:寄存器模式
OUT_REG	1'b0,1'b1	1'b0	输出寄存器 1'b0:旁路模式 1'b1:寄存器模式
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT 加/减模式选择 1'b0:加 1'b1:减
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT 加/减模式选择 1'b0:加 1'b1:减
ALUMODE	0,1,2	0	ALU54 操作模式及输入选择 0:ACC/0 +/- B +/- A; 1:ACC/0 +/- B + CASI; 2:A +/- B + CASI;
ALU_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	复位模式配置 SYNC: 同步复位 ASYNC: 异步复位

原语例化

可以直接实例化原语，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

Verilog 例化:

```

ALU54D alu54_inst (
    .A(a[53:0]),
    .B(b[53:0]),
    .CASI(casi[54:0]),
    .ASIGN(assign),
    .BSIGN(bsign),
    .ACCLOAD(acclload),
    .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.ACCLLOAD_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 例化:

```

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=>assign,
        BSIGN=>bsign,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        CASO=>caso,
        DOUT=>dout
    );
```

4.2 MULT

MULT (Multiplier) 是 DSP 的乘法器单元，乘法器的乘数输入信号定义为 A 和 B，乘积输出信号定义为 DOUT，可实现乘法运算： $DOUT = A * B$ 。

DSP 宏单元包含两个乘法器来进行乘法运算。为了满足不同的乘法位宽的需求，MULT 模式根据数据位宽可配置成 9x9, 18x18, 36x36 等乘法器，分别对应原语 MULT9X9, MULT18X18, MULT36X36。其中 36 x 36 乘法器需要一个 DSP 模块（即两个宏单元）进行配置。

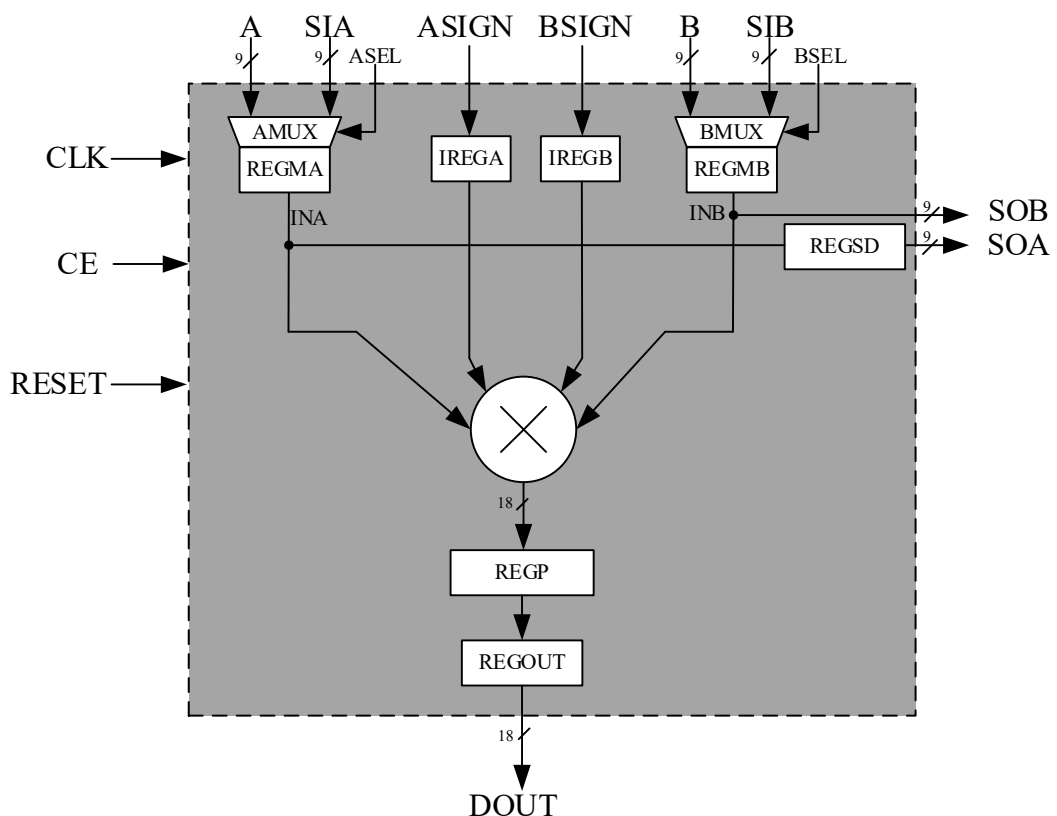
4.2.1 MULT9X9

原语介绍

MULT9X9 (9x9 Multiplier) 是 9x9 乘法器，实现了 9 位乘法运算。

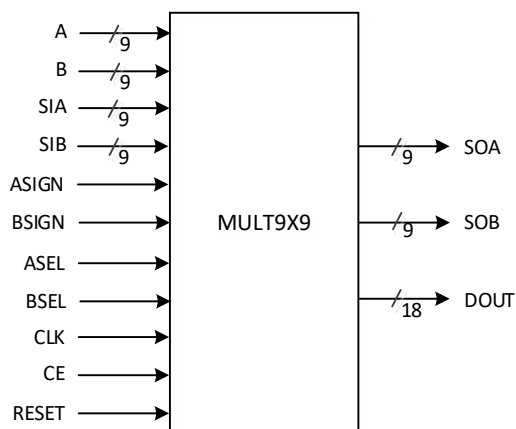
逻辑结构示意图

图 4-3 MULT9X9 逻辑结构示意图



端口示意图

图 4-4 MULT9X9 端口示意图



端口介绍

表 4-3 MULT9X9 端口介绍

端口	I/O	描述
A[8:0]	Input	9-bit 数据输入信号 A
B[8:0]	Input	9-bit 数据输入信号 B
SIA[8:0]	Input	9-bit 移位数据输入信号 A
SIB[8:0]	Input	9-bit 移位数据输入信号 B
ASIGN	Input	A 符号位输入信号
BSIGN	Input	B 符号位输入信号
ASEL	Input	源选择信号, SIA 或者 A
BSEL	Input	源选择信号, SIB 或者 B
CLK	Input	时钟输入信号
CE	Input	时钟使能信号
RESET	Input	复位输入信号
DOUT[17:0]	Output	数据输出信号
SOA[8:0]	Output	移位数据输出信号 A
SOB[8:0]	Output	移位数据输出信号 B

参数介绍

表 4-4 MULT9X9 参数介绍

参数	范围	默认	描述
AREG	1'b0,1'b1	1'b0	输入 A (SIA 或者 A) 寄存器 1'b0:旁路模式 1'b1:寄存器模式
BREG	1'b0,1'b1	1'b0	输入 B(SIB 或者 B)寄存器 1'b0:旁路模式 1'b1:寄存器模式
OUT_REG	1'b0,1'b1	1'b0	输出寄存器 1'b0:旁路模式 1'b1:寄存器模式
PIPE_REG	1'b0,1'b1	1'b0	Pipeline 寄存器 1'b0:旁路模式 1'b1:寄存器模式
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
SOA_REG	1'b0,1'b1	1'b0	SOA 寄存器 1'b0:旁路模式 1'b1:寄存器模式
MULT_RESET_MODE	"SYNC", "ASYNC"	"SYNC"	复位模式配置 SYNC: 同步复位 ASYNC: 异步复位

原语例化

可以直接实例化原语，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

Verilog 例化:

```
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(assign),
        .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 例化:

```

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=>assign,  
BSIGN=>bsign,  
ASEL=>asel,  
BSEL=>bsel,  
CE=>ce,  
CLK=>clk,  
RESET=>reset,  
SOA=>soa,  
SOB=>sob,  
DOUT=>dout  
);
```

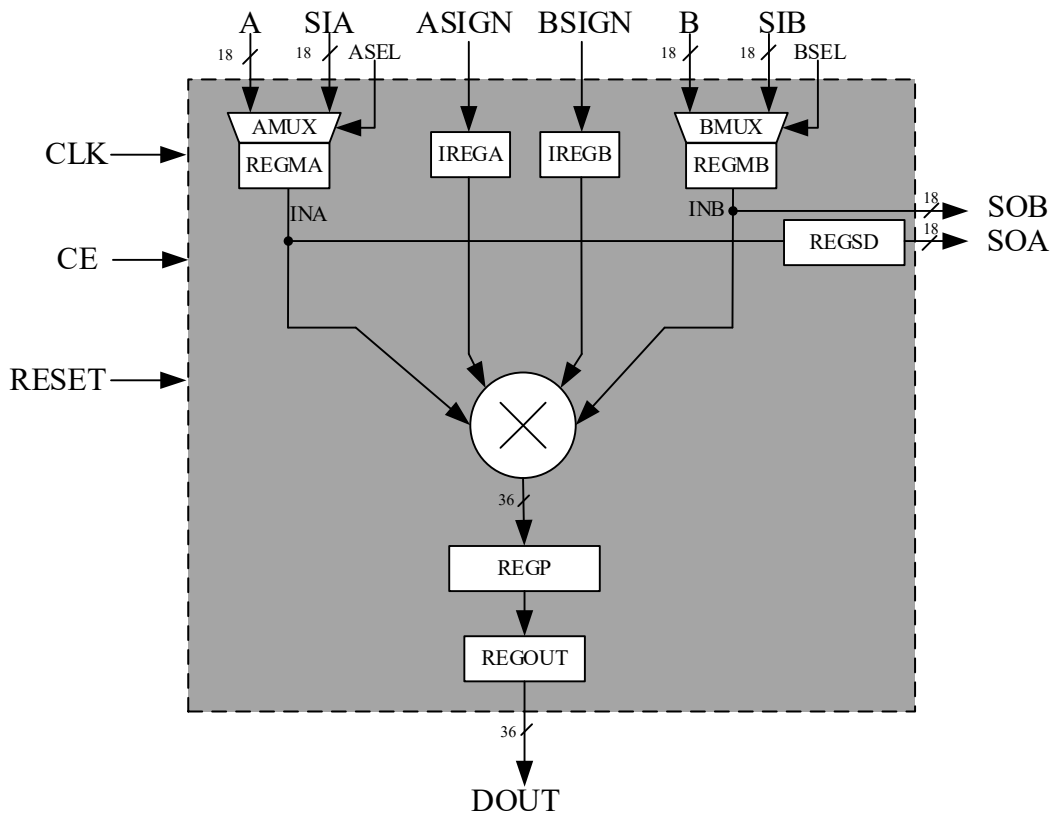

4.2.2 MULT18X18

原语介绍

MULT18X18 (18x18 Multiplier) 是 18x18 乘法器, 实现了 18 位乘法运算。

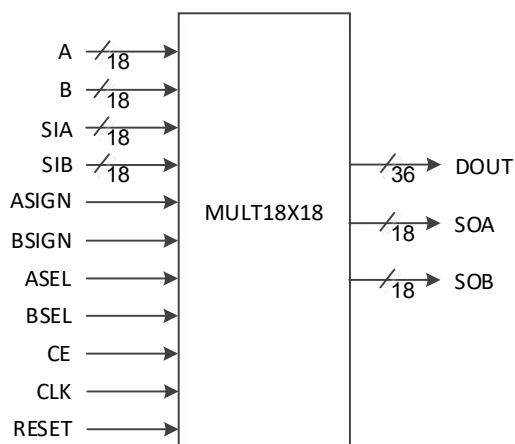
逻辑结构示意图

图 4-5 MULT18X18 逻辑结构示意图



端口示意图

图 4-6 MULT18X18 端口示意图



端口介绍

表 4-5 MULT18X18 端口介绍

端口	I/O	描述
A[17:0]	Input	18-bit 数据输入信号 A
B[17:0]	Input	18-bit 数据输入信号 B
SIA[17:0]	Input	18-bit 移位数据输入信号 A
SIB[17:0]	Input	18-bit 移位数据输入信号 B
ASIGN	Input	A 符号位输入信号
BSIGN	Input	B 符号位输入信号
ASEL	Input	源选择信号, SIA 或者 A
BSEL	Input	源选择信号, SIB 或者 B
CLK	Input	时钟输入信号
CE	Input	时钟使能信号
RESET	Input	复位输入信号
DOUT[35:0]	Output	数据输出信号
SOA[17:0]	Output	移位数据输出信号 A
SOB[17:0]	Output	移位数据输出信号 B

参数介绍

表 4-6 MULT18X18 参数介绍

参数	范围	默认	描述
AREG	1'b0,1'b1	1'b0	输入 A(SIA 或者 A)寄存器 1'b0:旁路模式 1'b1:寄存器模式
BREG	1'b0,1'b1	1'b0	输入 B(SIB 或者 B)寄存器 1'b0:旁路模式 1'b1:寄存器模式
OUT_REG	1'b0,1'b1	1'b0	输出寄存器 1'b0:旁路模式 1'b1:寄存器模式
PIPE_REG	1'b0,1'b1	1'b0	Pipeline 寄存器 1'b0:旁路模式 1'b1:寄存器模式
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
SOA_REG	1'b0,1'b1	1'b0	SOA 寄存器 1'b0:旁路模式 1'b1:寄存器模式
MULT_RESET_MODE	"SYNC","ASYNC"	"SYNC"	复位模式配置 SYNC: 同步复位 ASYNC: 异步复位

原语例化

可以直接实例化原语，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

Verilog 例化:

```
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(assign),  
    .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 例化:

```

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=>assign,
        BSIGN=>bsign,
        ASEL=>asel,
        BSEL=>bsel,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        SOA=>soa,
        SOB=>sob,
        DOUT=>dout
    );
```

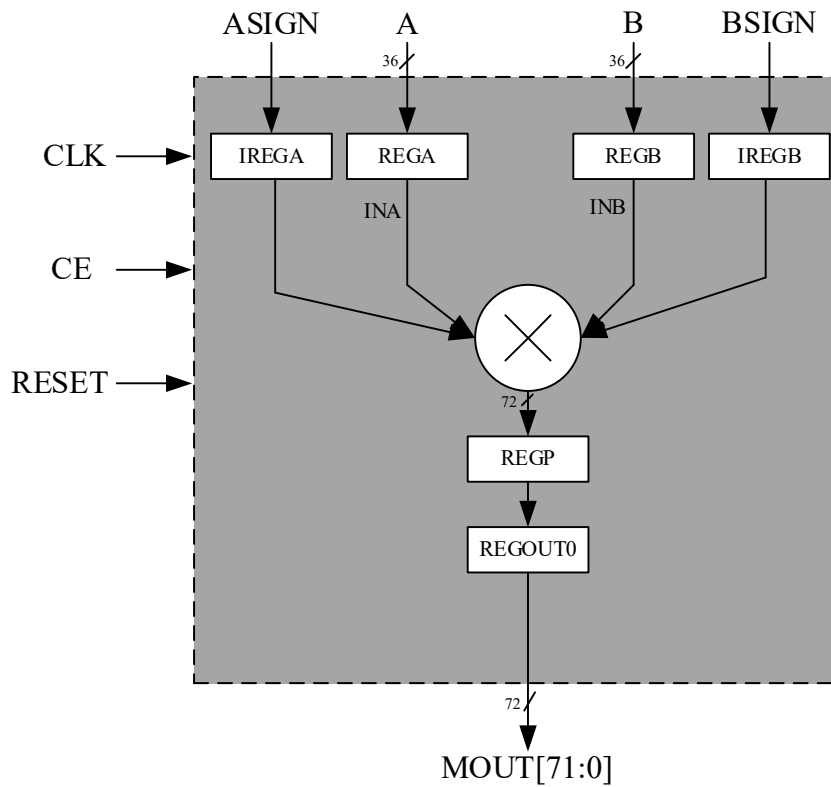
4.2.3 MULT36X36

原语介绍

MULT36X36 (36x36 Multiplier) 是 36x36 乘法器, 实现了 36 位的乘法运算。

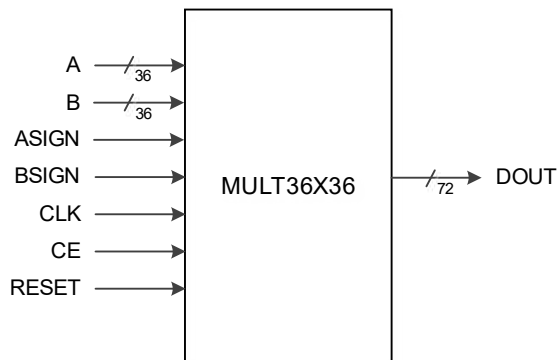
逻辑结构示意图

图 4-7 MULT36X36 逻辑结构示意图



端口示意图

图 4-8 MULT36X36 端口示意图



端口介绍

表 4-7 MULT36X36 端口介绍

端口	I/O	描述
A[35:0]	Input	36-bit 数据输入信号 A
B[35:0]	Input	36-bit 数据输入信号 B
ASIGN	Input	A 符号位输入信号
BSIGN	Input	B 符号位输入信号
CLK	Input	时钟输入信号
CE	Input	时钟使能信号
RESET	Input	复位输入信号
DOUT[71:0]	Output	数据输出信号

参数介绍

表 4-8 MULT36X36 参数介绍

参数	范围	默认	描述
AREG	1'b0,1'b1	1'b0	输入 A 寄存器. 1'b0:旁路模式 1'b1:寄存器模式
BREG	1'b0,1'b1	1'b0	输入 B 寄存器. 1'b0:旁路模式 1'b1:寄存器模式
OUT0_REG	1'b0,1'b1	1'b0	第一级输出寄存器 1'b0:旁路模式 1'b1:寄存器模式
OUT1_REG	1'b0,1'b1	1'b0	第二级输出寄存器 1'b0:旁路模式 1'b1:寄存器模式
PIPE_REG	1'b0,1'b1	1'b0	Pipeline 寄存器 1'b0:旁路模式 1'b1:寄存器模式
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 输入寄存器 1'b0:旁路模式

参数	范围	默认	描述
			1'b1:寄存器模式
MULT_RESET_MODE	"SYNC","ASYNC"	"SYNC"	复位模式配置 SYNC: 同步复位 ASYNC: 异步复位

原语例化

可以直接实例化原语，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

Verilog 例化:

```

MULT36X36 uut(
    .DOUT(mout[71:0]),
    .A(mdia[35:0]),
    .B(mdib[35:0]),
    .ASIGN(assign),
    .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 例化:

```
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=>assign,
                                    BSIGN=>bsign,
                                    CE=>ce,
                                    CLK=>clk,
                                    RESET=>reset,
                                    DOUT=>mout
                                );

```

4.3 MULTALU

MULTALU 模式实现一个乘法器输出经过 54-bit ALU 运算，包括 MULTALU36X18 和 MULTALU18X18。

4.3.1 MULTALU36X18

原语介绍

MULTALU36X18 (36x18 Multiplier with ALU) 是带 ALU 功能的 36X18 乘法器。

MULTALU36X18 有三种运算模式：

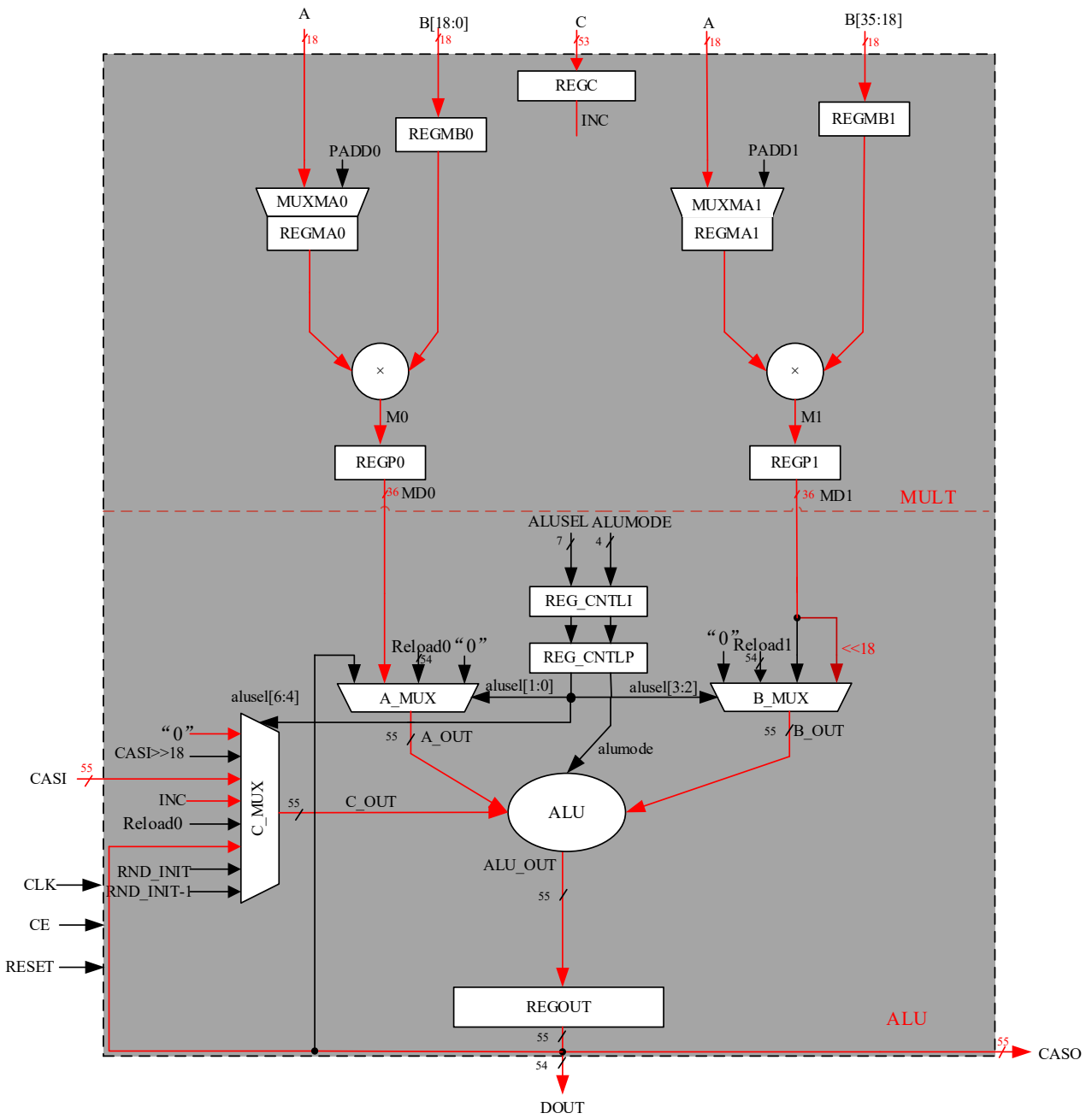
$$DOUT = A * B \pm C$$

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

$$DOUT = A * B + CASI$$

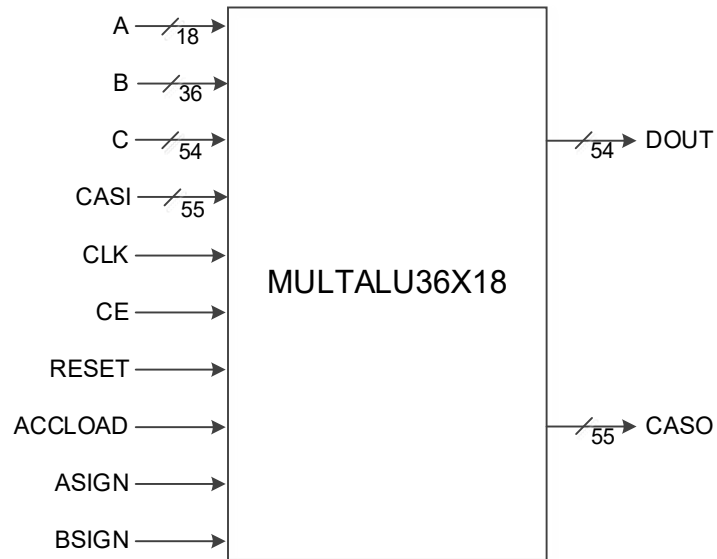
逻辑结构示意图

图 4-9 MULTALU36X18 逻辑结构示意图



端口示意图

图 4-10 MULTALU36X18 端口示意图



端口介绍

表 4-9 MULTALU36X18 端口介绍

端口	I/O	描述
A[17:0]	Input	18-bit 数据输入信号 A
B[35:0]	Input	36-bit 数据输入信号 B
C[53:0]	Input	54-bit Reload 数据输入信号
CASI[54:0]	Input	55-bit 级联输入信号
ASIGN	Input	A 符号位输入信号
BSIGN	Input	B 符号位输入信号
CLK	Input	时钟输入信号
CE	Input	时钟使能信号
RESET	Input	复位输入信号
ACCLOAD	Input	累加器 Reload 模式选择信号
DOUT[53:0]	Output	数据输出信号
CASO[54:0]	Output	55-bit 级联输出信号

参数介绍

表 4-10 MULTALU36X18 参数介绍

参数	范围	默认	描述
AREG	1'b0,1'b1	1'b0	输入 A 寄存器 1'b0:旁路模式 1'b1:寄存器模式
BREG	1'b0,1'b1	1'b0	输入 B 寄存器 1'b0:旁路模式 1'b1:寄存器模式
CREG	1'b0,1'b1	1'b0	输入 C 寄存器 1'b0:旁路模式 1'b1:寄存器模式
OUT_REG	1'b0,1'b1	1'b0	输出寄存器 1'b0:旁路模式 1'b1:寄存器模式
PIPE_REG	1'b0,1'b1	1'b0	Pipeline 寄存器 1'b0:旁路模式 1'b1:寄存器模式
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
ACCLOAD_REG0	1'b0,1'b1	1'b0	ACCLOAD 第一级寄存器 1'b0:旁路模式 1'b1:寄存器模式
ACCLOAD_REG1	1'b0,1'b1	1'b0	ACCLOAD 第二级寄存器 1'b0:旁路模式 1'b1:寄存器模式
MULT_RESET_MODE	"SYNC","ASYNC"	"SYNC"	复位模式配置 SYNC: 同步复位 ASYNC: 异步复位
MULTALU36X18_MODE	0,1,2	0	MULTALU36X18 操作模式 及输入选择 0:36x18 +/- C; 1:ACC/0 + 36x18; 2:36x18 + CASI
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT 加/减选择 1'b0: add 1'b1: sub

原语例化

可以直接实例化原语，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

Verilog 例化:

```
MULTALU36X18 multalu36x18_inst(
    .CASO(caso[54:0]),
    .DOUT(dout[53:0]),
    .ASIGN(assign),
    .BSIGN(bsign),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .CASI(casi[54:0]),
    .ACCLOAD(acclload),
    .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 例化:

```
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=>assign,
        BSIGN=>bsign,
        CE=>ce,
        CLK=>clk,
        RESET=>reset,
        ACCLOAD=>accload,
        CASI=>casi,
        CASO=>caso,
        DOUT=>dout
    );
```

4.3.2 MULTALU18X18

原语介绍

MULTALU18X18 (18x18 Multiplier with ALU) 是带 ALU 功能的 18x18 乘法器。

MULTALU18X18 有三种运算模式：

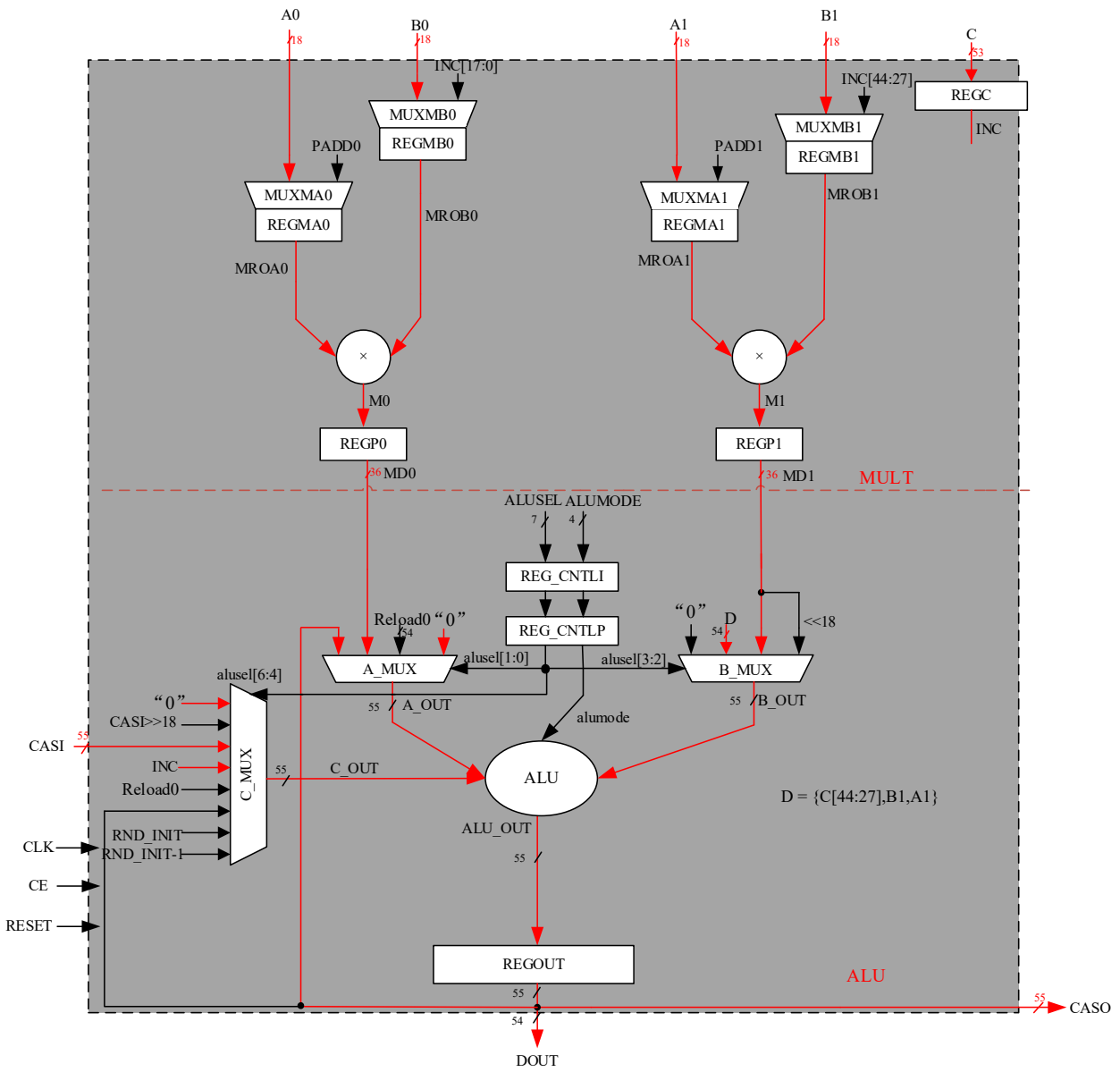
$$DOUT = \sum(A * B) \pm C$$

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

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

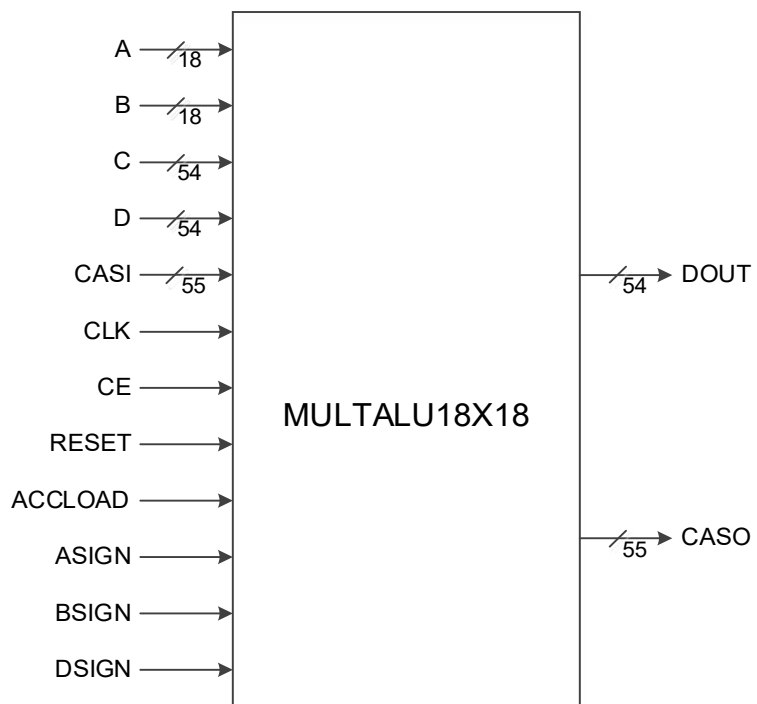
逻辑结构示意图

图 4-11 MULTALU18X18 逻辑结构示意图



端口示意图

图 4-12 MULTALU18X18 端口示意图



端口介绍

表 4-11 MULTALU18X18 端口介绍

端口	I/O	描述
A[17:0]	Input	18-bit 数据输入信号 A
B[17:0]	Input	18-bit 数据输入信号 B
C[53:0]	Input	54-bit 数据输入信号 C
D[53:0]	Input	54-bit 数据输入信号 D
CASI[54:0]	Input	55-bit 级联输入信号
ASIGN	Input	A 符号位输入信号
BSIGN	Input	B 符号位输入信号
DSIGN	Input	D 符号位输入信号
CLK	Input	时钟输入信号
CE	Input	时钟使能信号
RESET	Input	复位输入信号
ACCLOAD	Input	累加器 Reload 模式选择信号
DOUT[53:0]	Output	数据输出信号
CASO[54:0]	Output	55-bit 级联输出信号

参数介绍

表 4-12 MULTALU18X18 参数介绍

参数	范围	默认	描述
AREG	1'b0,1'b1	1'b0	输入 A 寄存器 1'b0:旁路模式 1'b1:寄存器模式
BREG	1'b0,1'b1	1'b0	输入 B 寄存器 1'b0:旁路模式 1'b1:寄存器模式
CREG	1'b0,1'b1	1'b0	输入 C 寄存器 1'b0:旁路模式 1'b1:寄存器模式
DREG	1'b0,1'b1	1'b0	输入 D 寄存器 1'b0:旁路模式 1'b1:寄存器模式
DSIGN_REG	1'b0,1'b1	1'b0	DSIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
ASIGN_REG	1'b0,1'b1	1'b0	ASIGN 输入寄存器 1'b0:旁路模式 1'b1:寄存器模式
BSIGN_REG	1'b0,1'b1	1'b0	BSIGN 输入寄存器. 1'b0:旁路模式 1'b1:寄存器模式
ACCLOAD_REG0	1'b0,1'b1	1'b0	ACCLOAD 第一级寄存器 1'b0:旁路模式 1'b1:寄存器模式
ACCLOAD_REG1	1'b0,1'b1	1'b0	ACCLOAD 第二级寄存器 1'b0:旁路模式 1'b1:寄存器模式
MULT_RESET_MODE	"SYNC","ASYNC"	"SYNC"	复位模式配置 SYNC: 同步复位 ASYNC: 异步复位
PIPE_REG	1'b0,1'b1	1'b0	Pipeline 寄存器 1'b0:旁路模式 1'b1:寄存器模式
OUT_REG	1'b0,1'b1	1'b0	输出寄存器 1'b0:旁路模式 1'b1:寄存器模式
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT 加/减模式选择

参数	范围	默认	描述
			1'b0:加 1'b1:减
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT 加/减模式选择 1'b0:加 1'b1:减
MULTALU18X18_MODE	0,1,2	0	MULTALU36X18 操作模式及输入选择 0:ACC/0 +/- 18x18 +/- C; 1:ACC/0 +/- 18x18 + CASI; 2: 18x18 +/- D + CASI;

原语例化

可以直接实例化原语，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

Verilog 例化:

```
MULTALU18X18 multalu18x18_inst(
    .CASO(caso[54:0]),
    .DOUT(dout[53:0]),
    .ASIGN(assign),
    .BSIGN(bsign),
    .DSIGN(dsign),
    .CE(ce),
    .CLK(clk),
    .RESET(reset),
    .CASI(casi[54:0]),
    .ACCLoad(acclload),
    .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 例化:

```

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=>assign,  
    BSIGN=>bsign,  
    DSIGN=>dsign,  
    CE=>ce,  
    CLK=>clk,  
    RESET=>reset,  
    ACCLOAD=>accload,  
    CASI=>casi,  
    CASO=>caso,  
    DOUT=>dout  
);
```

4.4 MULTADDALU

MULTADDALU 模式实现两个 18 x 18 乘法器输出经过 54-bit ALU 运算，对应原语为 MULTADDALU18X18。

MULTADDALU18X18 有三种运算模式：

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

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

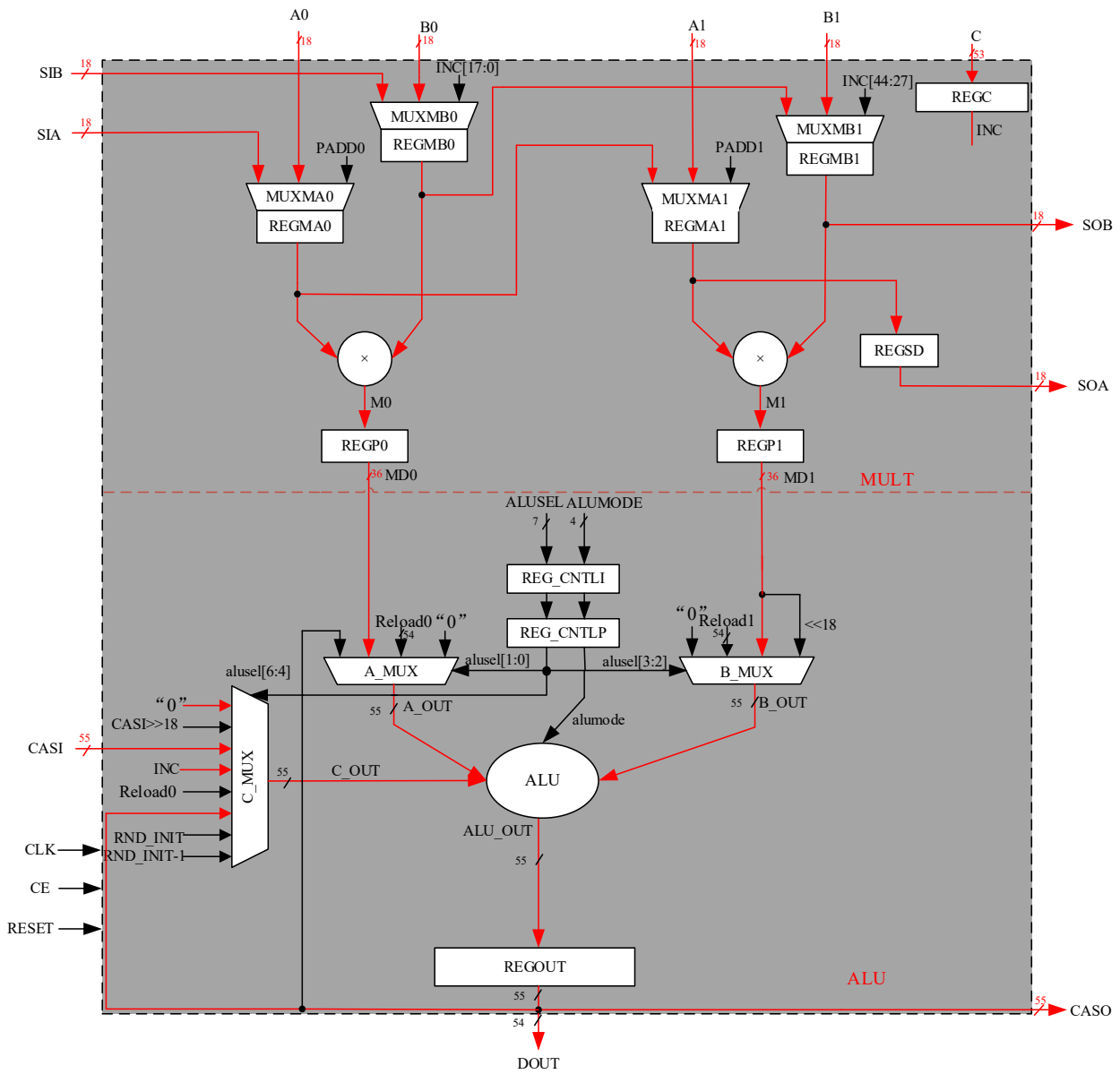
$$DOUT = A0 * B0 \pm A1 * B1 + CASI$$

原语介绍

MULTADDALU18X18 (The Sum of Two 18x18 Multipliers with ALU) 是带 ALU 功能的 18x18 乘加器，实现 18 位的乘法求和后累加或 reload 运算。

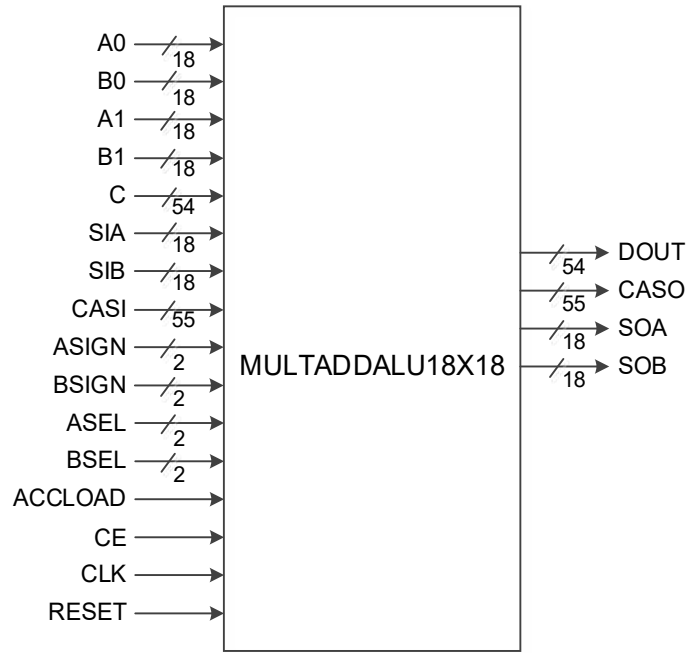
逻辑结构示意图

图 4-13 MULTADDALU18X18 逻辑结构示意图



端口示意图

图 4-14 MULTADDALU18X18 端口示意图



端口介绍

表 4-13 MULTADDALU18X18 端口介绍

端口	I/O	描述
A0[17:0]	Input	18-bit 数据输入信号 A0
B0[17:0]	Input	18-bit 数据输入信号 B0
A1[17:0]	Input	18-bit 数据输入信号 A1
B1[17:0]	Input	18-bit 数据输入信号 B1
C[53:0]	Input	54-bit Reload 数据输入信号 C
SIA[17:0]	Input	18-bit 移位数据输入信号 A
SIB[17:0]	Input	18-bit 移位数据输入信号 B
CASI[54:0]	Input	55-bit 级联输入信号
ASIGN[1:0]	Input	A1,A0 符号位输入信号
BSIGN[1:0]	Input	B1,B0 符号位输入信号
ASEL[1:0]	Input	输入 A1,A0 源选择信号
BSEL[1:0]	Input	输入 B1,B0 源选择信号
CLK	Input	时钟输入信号
CE	Input	时钟使能信号
RESET	Input	复位输入信号
ACCLOAD	Input	累加器 Reload 模式选择信号
DOUT[53:0]	Output	数据输出信号
CASO[54:0]	Output	55-bit 级联输出信号
SOA[17:0]	Output	移位数据输出信号 A
SOB[17:0]	Output	移位数据输出信号 B

参数介绍

表 4-14 MULTADDALU18X18 参数介绍

参数	范围	默认	描述
A0REG	1'b0,1'b1	1'b0	输入 A0(A0 或者 SIA)寄存器. 1'b0:旁路模式 1'b1:寄存器模式
A1REG	1'b0,1'b1	1'b0	输入 A1(A1 或者寄存器输出 A0)寄存器. 1'b0:旁路模式 1'b1:寄存器模式
B0REG	1'b0,1'b1	1'b0	输入 B0(B0 或者 SIB) 寄存器. 1'b0:旁路模式 1'b1:寄存器模式
B1REG	1'b0,1'b1	1'b0	输入 B1(B1 或者寄存器输出 B0)寄存器. 1'b0:旁路模式 1'b1:寄存器模式
CREG	1'b0,1'b1	1'b0	输入 C 寄存器 1'b0:旁路模式 1'b1:寄存器模式
PIPE0_REG	1'b0,1'b1	1'b0	Multiplier0 Pipeline 寄存器. 1'b0:旁路模式 1'b1:寄存器模式
PIPE1_REG	1'b0,1'b1	1'b0	Multiplier1 Pipeline 寄存器. 1'b0:旁路模式 1'b1:寄存器模式
OUT_REG	1'b0,1'b1	1'b0	输出寄存器 1'b0:旁路模式 1'b1:寄存器模式
ASIGN0_REG	1'b0,1'b1	1'b0	ASIGN[0]输入寄存器. 1'b0:旁路模式 1'b1:寄存器模式
ASIGN1_REG	1'b0,1'b1	1'b0	ASIGN[1]输入寄存器. 1'b0:旁路模式 1'b1:寄存器模式
ACCLOAD_REG0	1'b0,1'b1	1'b0	ACCLOAD 第一级寄存器 1'b0:旁路模式 1'b1:寄存器模式
ACCLOAD_REG1	1'b0,1'b1	1'b0	ACCLOAD 第二级寄存器 1'b0:旁路模式 1'b1:寄存器模式
BSIGN0_REG	1'b0,1'b1	1'b0	BSIGN[0] 输入寄存器.

参数	范围	默认	描述
			1'b0:旁路模式 1'b1:寄存器模式
BSIGN1_REG	1'b0,1'b1	1'b0	BSIGN[1] 输入寄存器. 1'b0:旁路模式 1'b1:寄存器模式
SOA_REG	1'b0,1'b1	1'b0	SOA 寄存器. 1'b0:旁路模式 1'b1:寄存器模式
B_ADD_SUB	1'b0,1'b1	1'b0	B_OUT 加/减选择 1'b0:加 1'b1:减
C_ADD_SUB	1'b0,1'b1	1'b0	C_OUT 加/减选择 1'b0:加 1'b1:减
MULTADDALU18X18_MODE	0,1,2	0	MULTADDALU18X18 操作模式及输入选择 0:18x18 +/- 18x18 +/- C; 1: ACC/0 + 18x18 +/- 18x18; 2:18x18 +/- 18x18 + CASI
MULT_RESET_MODE	"SYNC", "ASYN"	"SYNC"	复位模式配置 SYNC: 同步复位 ASYN: 异步复位

原语例化

可以直接实例化原语，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

Verilog 例化:

```
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(acclload),
.ASEL(asel[1:0]),
.BSEL(bsel[1:0]),
.ASIGN(assign[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 例化:

```
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=>assign,
        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 模式

PADD (Pre-adder) 是预加器，实现预加、预减和移位功能。DSP 宏单元包含两个预加器，实现预加、预减和移位功能。预加器位于 DSP 宏单元的最前端，有两个输入端，一个是并行 18-bit 输入 A 或 SIA，另一个是并行 18-bit 输入 B 或 SBI。为了增强时序功能，每个输入端都增加了对应的寄存器。另外，也可以把预加器旁路使输入端 A、B 直接作用到乘法器模块。高云半导体 FPGA 产品的预加器可以作为功能模块单独使用，按照位宽不同分为两种，分别是 9-bit 位宽的 PADD9 和 18-bit 位宽的 PADD18。

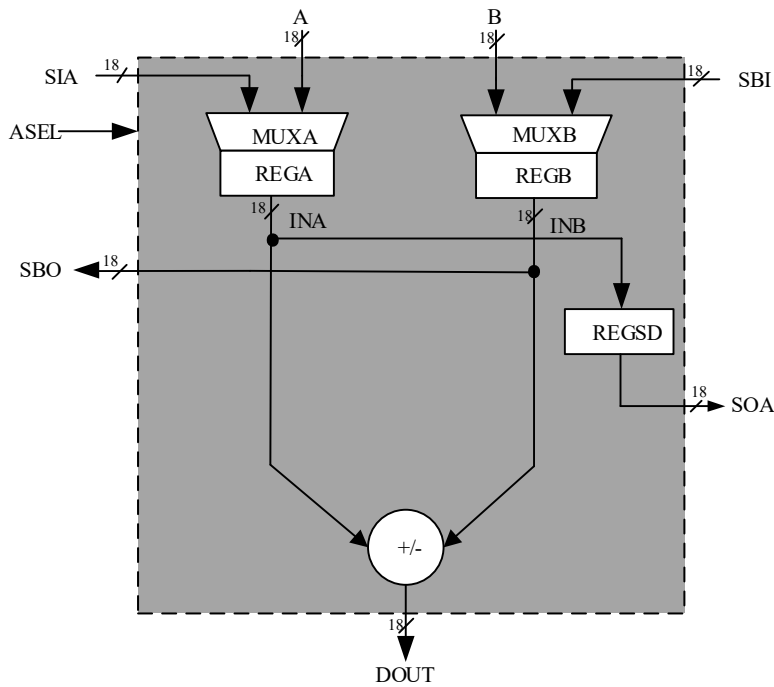
4.5.1 PADD18

原语介绍

PADD18 (18-bit Pre-Adder) 是 18 位预加器, 实现了 18 位的预加、预减或移位功能。

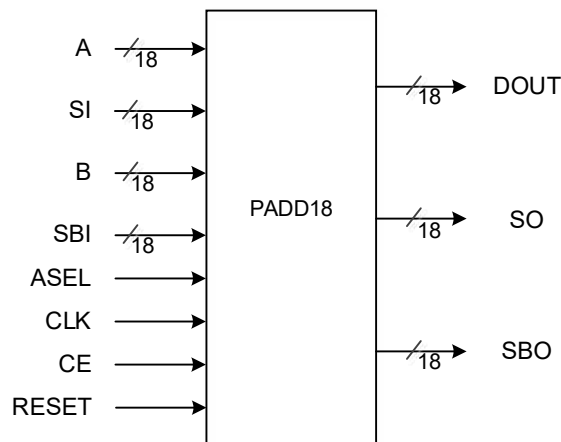
逻辑结构示意图

图 4-15 PADD18 逻辑结构示意图



端口示意图

图 4-16 PADD18 端口示意图



端口介绍

表 4-15 PADD18 端口介绍

端口	I/O	描述
A[17:0]	Input	18-bit 数据输入信号 A
B[17:0]	Input	18-bit 数据输入信号 B
SI[17:0]	Input	移位数据输入信号 A
SBI[17:0]	Input	预加器移位输入信号, 反向
ASEL	Input	源选择输入信号, SI 或者 A
CLK	Input	时钟输入信号
CE	Input	时钟使能信号
RESET	Input	复位输入信号
SO[17:0]	Output	移位数据输出信号 A
SBO[17:0]	Output	预加器移位输出信号, 反向
DOUT[17:0]	Output	数据输出信号

参数介绍

表 4-16 PADD18 参数介绍

参数	范围	默认	描述
AREG	1'b0,1'b1	1'b0	输入 A(A 或者 SI)寄存器 1'b0:旁路模式 1'b1:寄存器模式
BREG	1'b0,1'b1	1'b0	输入 B(B 或者 SBI)寄存器 1'b0:旁路模式 1'b1:寄存器模式
ADD_SUB	1'b0,1'b1	1'b0	加/减选择 1'b0:加 1'b1:减
PADD_RESET_MODE	"SYNC","ASYNC"	"SYNC"	复位模式配置 SYNC: 同步复位 ASYNC: 异步复位
BSEL_MODE	1'b1,1'b0	1'b1	输入 B 选择 1'b1: SBI 1'b0: B
S 或者 EG	1'b0,1'b1	1'b0	移位输出寄存器 1'b0:旁路模式 1'b1:寄存器模式

原语例化

可以直接实例化原语，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

Verilog 例化:

```
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 例化:

```
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  
);
```

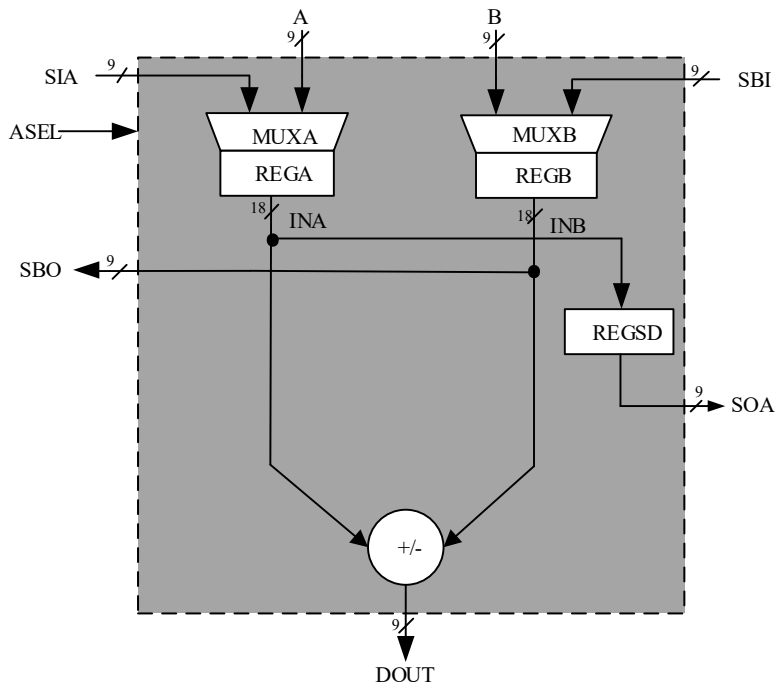
4.5.2 PADD9

原语介绍

PADD9 (9-bit Pre-Adder) 是 9 位预加器，实现了 9 位的预加、预减或移位功能。

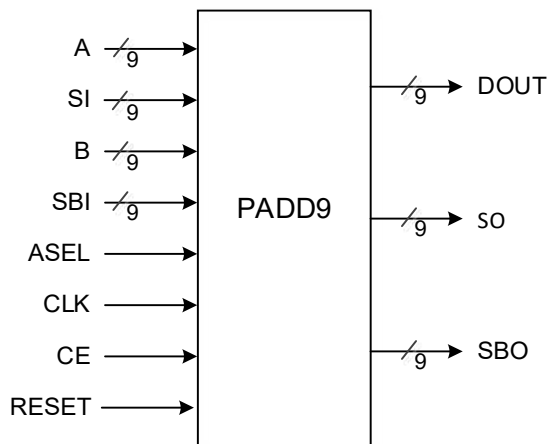
逻辑结构示意图

图 4-17 PADD9 逻辑结构示意图



端口示意图

图 4-18 PADD9 端口示意图



端口介绍

表 4-17 PADD9 端口介绍

端口	I/O	描述
A[8:0]	Input	9-bit 数据输入信号 A
B[8:0]	Input	9-bit 数据输入信号 B
SI[8:0]	Input	移位数据输入信号 A
SBI[8:0]	Input	预加器移位输入信号, 反向
ASEL	Input	源选择输入信号, SI 或者 A
CLK	Input	时钟输入信号
CE	Input	时钟使能信号
RESET	Input	复位输入信号
SO[8:0]	Output	移位数据输出信号 A
SBO[8:0]	Output	预加器移位输出信号, 反向
DOUT[8:0]	Output	数据输出信号

参数介绍

表 4-18 PADD9 参数介绍

参数	范围	默认	描述
AREG	1'b0,1'b1	1'b0	输入 A(A 或者 SI)寄存器 1'b0:旁路模式 1'b1:寄存器模式
BREG	1'b0,1'b1	1'b0	输入 B(B 或者 SBI)寄存器 1'b0:旁路模式 1'b1:寄存器模式
ADD_SUB	1'b0,1'b1	1'b0	加/减选择 1'b0:加 1'b1:减
PADD_RESET_MODE	"SYNC","ASYNC"	"SYNC"	复位模式配置 SYNC: 同步复位 ASYNC: 异步复位
BSEL_MODE	1'b1,1'b0	1'b1	输入 B 选择 1'b1: SBI 1'b0: B
SOREG	1'b0,1'b1	1'b0	移位输出寄存器 1'b0:旁路模式 1'b1:寄存器模式

原语例化

可以直接实例化原语，也可以通过 IP Core Generator 工具产生，具体可参考 5 IP 调用。

Verilog 例化:

```
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 例化:

```
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 调用

IP Core Generator 中 DSP 模块支持五种高云原语的产生：ALU54、MULT、MULTADDALU、MULTALU、PADD。

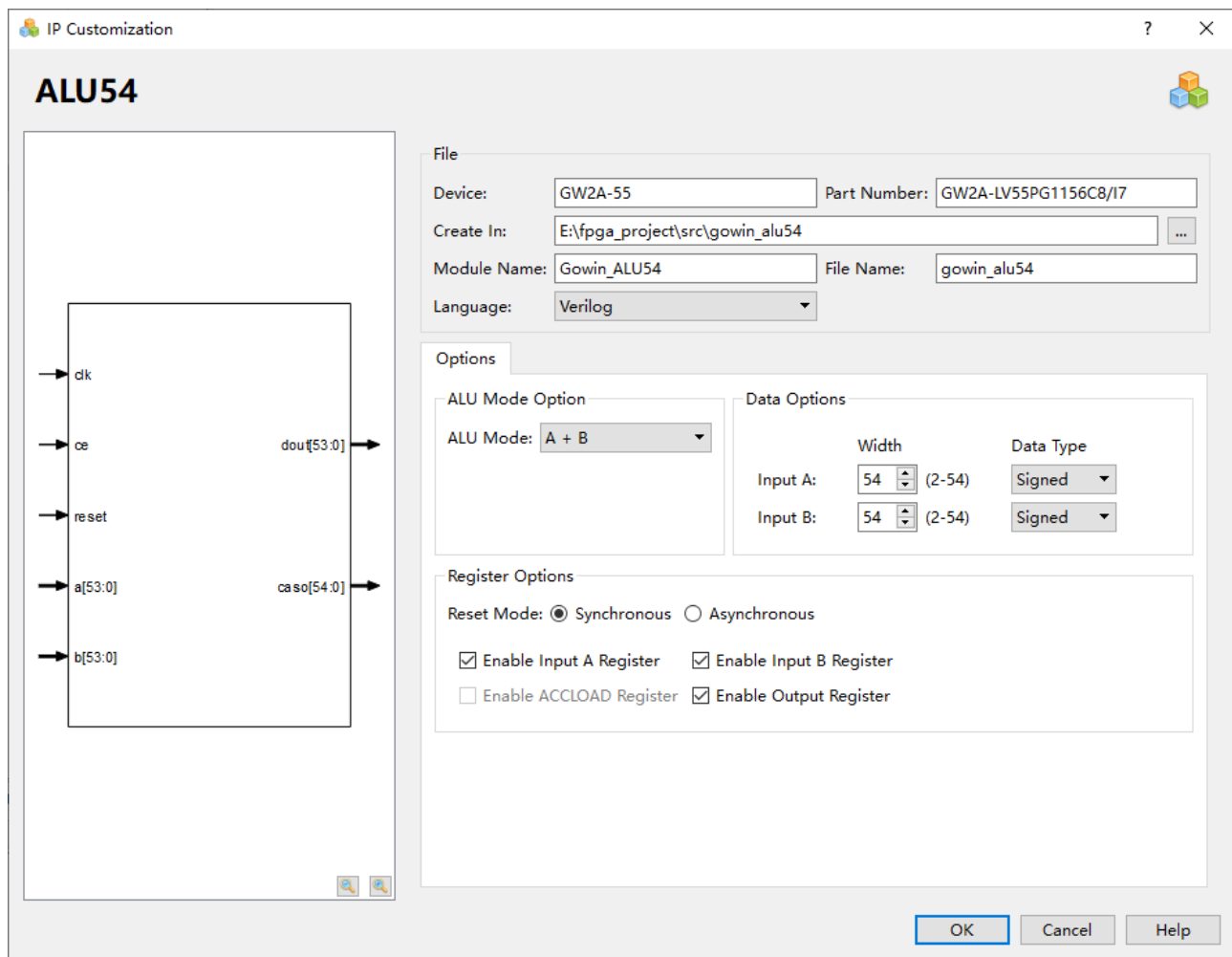
5.1 ALU54

ALU54 实现 54 位算数逻辑运算。在 IP Core Generator 界面中，单击“ALU54”，界面右侧会显示 ALU54 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“ALU54”，弹出 ALU54 的“IP Customization”窗口，如图 5-1 所示。该窗口包括“File”配置框、“Options”配置框、端口显示框图和“Help”按钮。

图 5-1 ALU54 的 IP Customization 窗口结构



1. File 配置框

File 配置框用于配置产生的 IP 设计文件的相关信息。

- **Device:** 显示已配置的 Device 信息；
- **Part Number:** 显示已配置的 Part Number 信息；
- **Language:** 配置产生的 IP 设计文件的硬件描述语言。选择右侧下拉列表框，选择目标语言，支持 Verilog 和 VHDL；
- **Module Name:** 配置产生的 IP 设计文件的 module name。在右侧文本框可重新编辑模块名称。Module Name 不能与原语名称相同，若相同，则报出 Error 提示；
- **File Name:** 配置产生的 IP 设计文件的文件名。在右侧文本框可重新编辑文件名称；
- **Create In:** 配置产生的 IP 设计文件的目标路径。可在右侧文本框中重新编辑目标路径，也可通过文本框右侧选择按钮选择目标路径。

2. Options 配置框

Options 配置框用于用户自定义配置 IP, Options 配置框如图 5-1 所示。

- **ALU Mode Option:** 配置 ALU54 的运算模式。可选择:
 - $A + B$;
 - $A - B$;
 - 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:** 配置数据选项。
 - 配置 ALU54 输入数据位宽。输入 A/B 端的数据可配置为 1-54 位;
 - 输出端口数据位宽无需用户配置, 其会根据输入位宽自动调整位宽;
 - “Data Type” 选项可配置为 Signed、Unsigned。
- **Register Options:** 配置寄存器工作模式。
 - “Reset Mode” 选项配置 ALU54 的复位模式, 支持同步模式 “Synchronous” 和异步模式 “Asynchronous”;
 - “Enable Input A Register” 配置 Input A register;
 - “Enable Input B Register” 配置 Input B register;
 - “Enable ACCLOAD Register” 配置 ACCLOAD register;
 - “Enable Output Register” 配置 Output register。

3. 端口显示框图

端口显示框图显示当前 IP Core 的配置结果示例框图, 输入输出端口

的位宽根据 Options 配置实时更新，如图 5-1 所示。

4. Help 按钮

单击“Help”，显示 IP Core 的配置信息的页面。Help 页面包括当前 IP Core 的概要介绍，以及 Options 各项配置的简要说明。

IP 生成文件

IP 窗口配置完成后，产生以配置文件“File Name”命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_alu54.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 ALU54；
- IP 设计使用模板文件 gowin_alu54_tmp.v，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_alu54.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

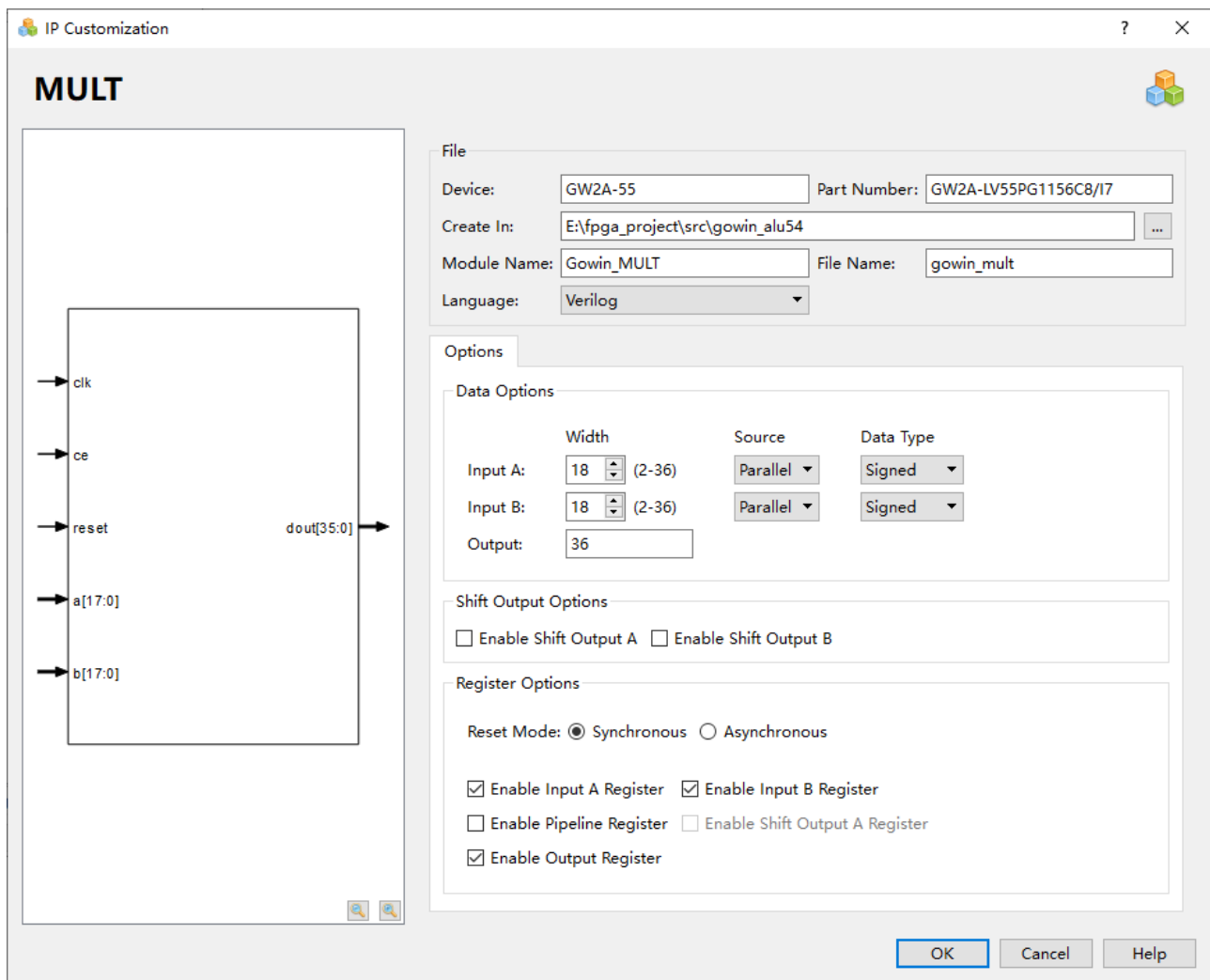
5.2 MULT

MULT 实现乘法运算功能。在 IP Core Generator 界面中单击“MULT”，界面右侧会显示 MULT 的相关信息概要。

IP 配置

在 IP Core Generator 界面中双击“MULT”，弹出 MULT 的“IP Customization”窗口，如图 5-2 所示。该窗口包括“File”配置框、“Options”配置框、端口显示框图以及“Help”按钮。

图 5-2 MULT 的 IP Customization 窗口结构



1. File 配置框

- File 配置框用于配置产生的 IP 设计文件的相关信息。
- MULT 的 File 配置框的使用和 ALU54 模块的类似, 请参考 5.1 ALU54 中的 File 配置框。

2. Options 配置框

- Options 配置框用于用户自定义配置 IP, Options 配置框如图 5-2 所示。
- Data Options: 配置数据选项。
 - 输入端口 (Input A Width/ Input B Width) 最大数据位宽为 36;
 - 输出端口数据位宽 (Output Width) 无需用户配置, 它会根据输入位宽自动调整位宽, 例化时会根据位宽生成 MULT9X9,

MULT18X18, MULT36X36。

- 输入端口 A/B 可配置为 Parallel、Shift;
- 数据类型可配置为 Unsigned、Signed。
- **Shift Output Options:** 能否使能 shift out 功能, 输入端口 (Input A Width/ Input B Width) 都小于等于 18 时, 可使用此功能。

注!

输入端口 (Input A Width/ Input B Width) 任一项大于 18 时, Shift Output Options 置灰, 不可使用。

- **Register Options:** 该选项的功能、用法与 ALU54 的 Register Options 选项相同, 请参考 5.1 ALU54 中的 Option 配置框。

3. 端口显示框图

端口显示框图显示当前 IP Core 的配置结果示例框图, 输入输出端口的个数以及位宽根据 Options 配置实时更新, 如图 5-2 所示。

4. Help 按钮

单击“Help”, 显示 IP Core 的配置信息的页面。Help 页面包括当前 IP Core 的概要介绍, 以及 Options 各项配置的简要说明。

IP 生成文件

IP 窗口配置完成后, 产生以配置文件“File Name”命名的三个文件, 以默认配置为例进行介绍:

- IP 设计文件“gowin_mult.v”为完整的 verilog 模块, 根据用户的 IP 配置, 产生实例化的 MULT;
- IP 设计使用模板文件 gowin_mult_tmp.v, 为用户提供 IP 设计使用模板文件;
- IP 配置文件: “gowin_mult.ipc”, 用户可加载该文件对 IP 进行配置。

注!

如配置中选择语言是 VHDL, 则产生的前两个文件名后缀为.vhd。

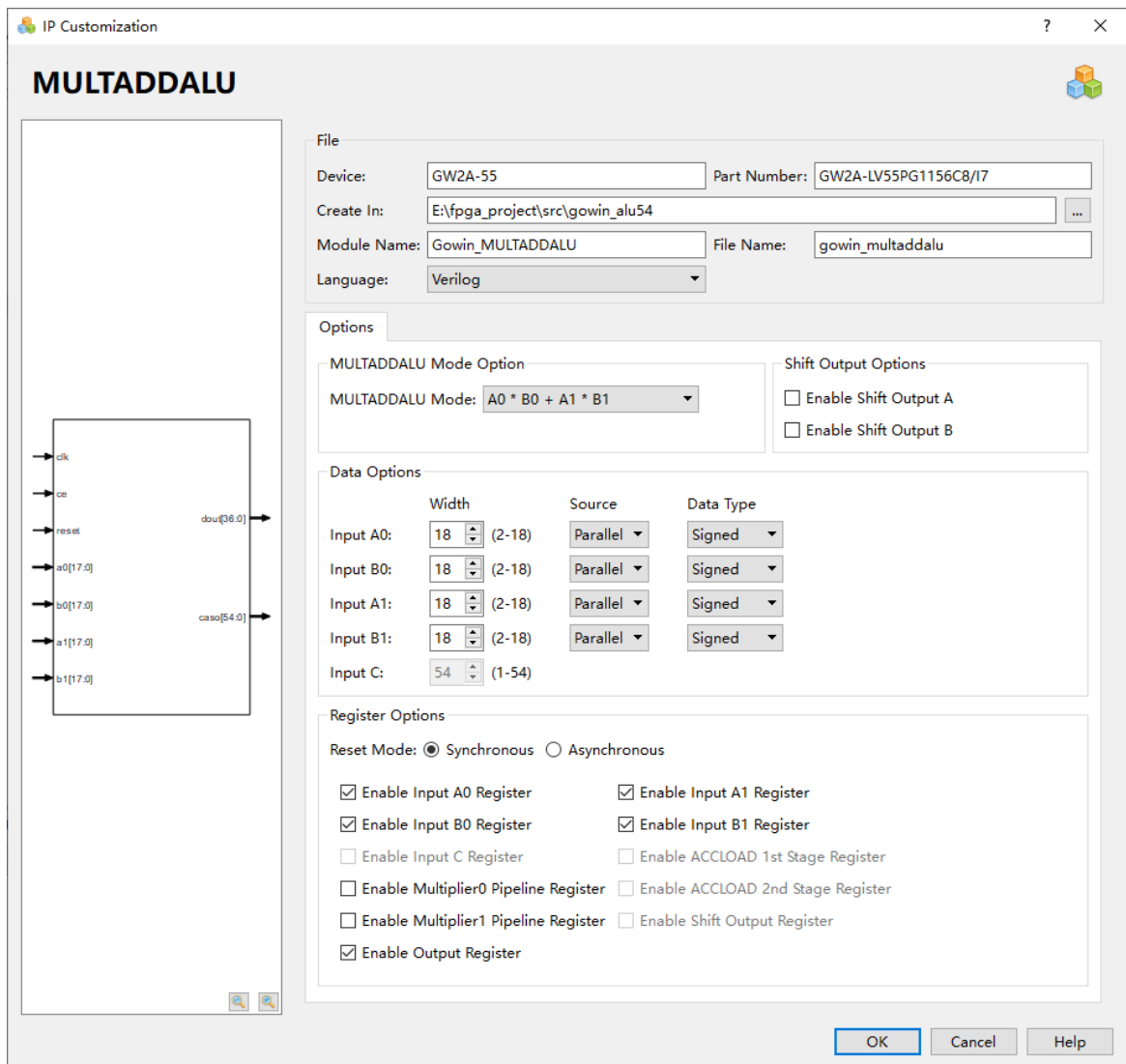
5.3 MULTADDALU

MULTADDALU 实现乘法器二次求和或累加功能。在 IP Core Generator 界面中，单击“MULTADDALU”，界面右侧会显示 MULTADDALU 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“MULTADDALU”，弹出 MULTADDALU 的“IP Customization”窗口。该窗口包括“File”配置框、“Options”配置框、端口显示框图和“Help”按钮，如图 5-3 所示。

图 5-3 MULTADDALU 的 IP Customization 窗口结构



1. File 配置框

- File 配置框用于配置产生的 IP 设计文件的相关信息。
- MULTADDALU 的 File 配置框的使用和 ALU54 模块的类似，请参考 5.1 ALU54 中的 File 配置框。

2. Options 配置框

Options 配置框用于用户自定义配置 IP，Options 配置框如图 5-3 所示。

- MULTADDALU Mode Option: 配置 MULTADDALU 的运算模式。可选择：
 - $A0*B0 + A1*B1$
 - $A0*B0 - A1*B1$
 - $A0*B0 + A1*B1 + C$
 - $A0*B0 + A1*B1 - C$
 - $A0*B0 - A1*B1 + C$
 - $A0*B0 - A1*B1 - C$
 - $Accum + A0*B0 + A1*B1$
 - $Accum + A0*B0 - A1*B1$
 - $A0*B0 + A1*B1 + CASI$
 - $A0*B0 - A1*B1 + CASI$
- MULTADDALU 的 Data Options 和 Register Options 配置框的使用和 MULT 模块的类似，请参考 5.2 MULT。

3. 端口显示框图

端口显示框图显示当前 IP Core 的配置结果示例框图，输入输出端口的位宽根据 Data Options 和 Register Options 配置实时更新，如图 5-3 所示。

4. Help 按钮

单击“Help”，显示 IP Core 的配置信息的页面。Help 页面包括 IP Core 的概要介绍，以及 Data Options 和 Register Options 各项配置的简要说明。

IP 生成文件

IP 窗口配置完成后，产生以配置文件“File Name”命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_multaddalu.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 MULTADDALU；
- IP 设计使用模板文件 gowin_multaddalu_tmp.v，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_multaddalu.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

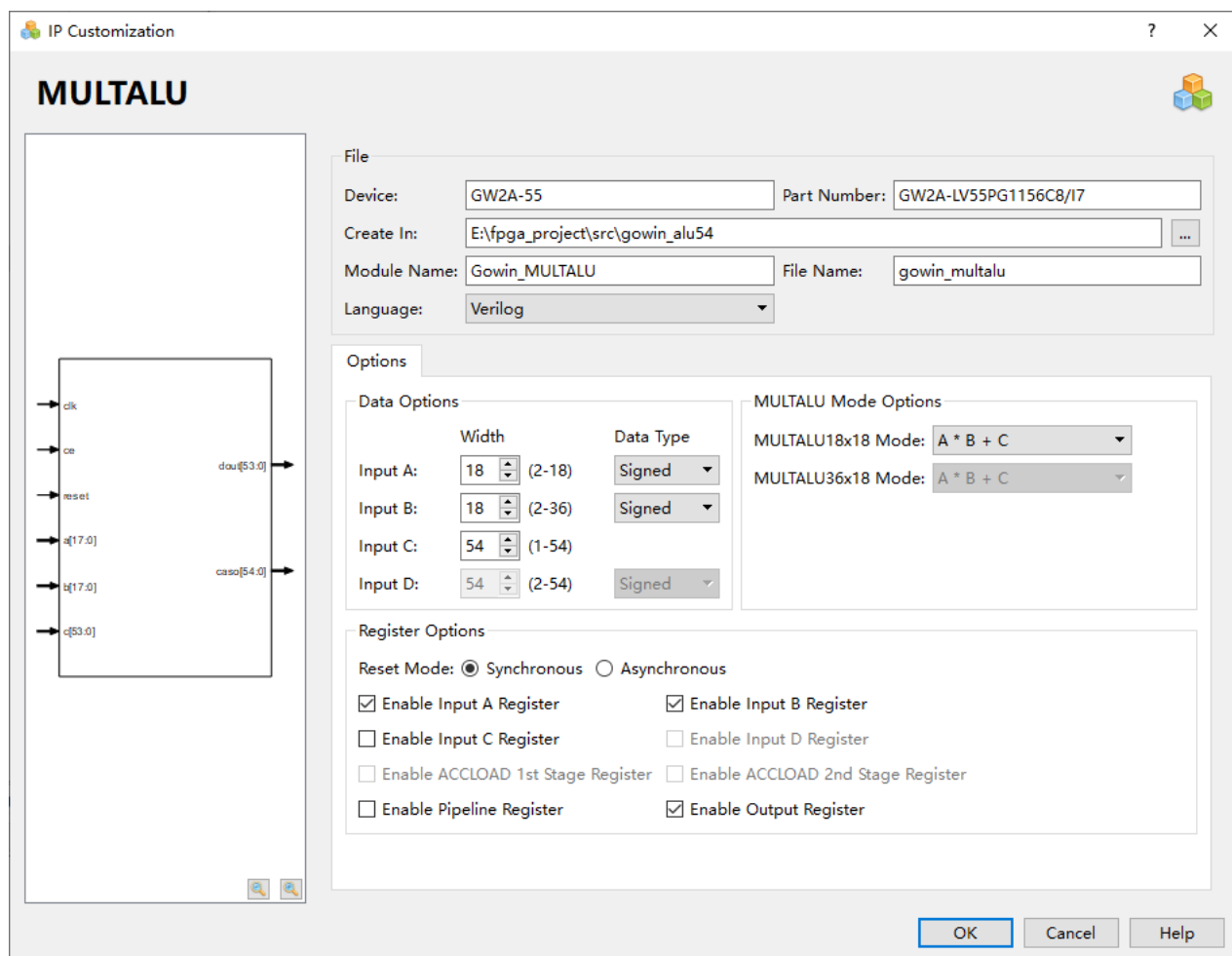
5.4 MULTALU

MULTALU 实现乘积求和或累加的功能。在 IP Core Generator 界面中，单击 MULTALU，界面右侧会显示 MULTALU 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“MULTALU”，弹出“IP Customization”窗口。该窗口包括“File”配置框、“Options”配置框、端口显示框图和“Help”按钮，如图 5-4 所示。

图 5-4 MULTALU 的 IP Customization 窗口结构



1. File 配置框

- File 配置框用于配置产生的 IP 设计文件的相关信息。
- MULTALU 的 File 配置框的使用和 ALU54 模块的类似，请参考 5.1 ALU54 中的 File 配置框。

2. Options 配置框

Options 配置框用于用户自定义配置 IP，Options 配置框如图 5-4 所示。

● MULTALU Mode Option

IP Core 中的 MULTALU 根据输入端口的位宽可以生成两种模块：MULTALU36X18 或 MULTALU18X18。当 Input A 和 Input B 的 width 都小于或等于 18 位时，Options 配置框右侧的 MULTALU Mode Options 中 MULTALU36X18 Mode 置灰，MULTALU18X18 Mode 可以配置为：

- $A * B + C$
- $A * B - C$

- $Accum + A*B + C$
 - $Accum + A*B - C$
 - $Accum - A*B + C$
 - $Accum - A*B - C$
 - $A*B + CASI$
 - $Accum + A*B + CASI$
 - $Accum - A*B + CASI$
 - $A*B + D + CASI$
 - $A*B - D + CASI$
- 当 Input B 的 width 大于 18 位时，MULTALU18X18 Mode 置灰，
 - MULTALU36X18 Mode 可以配置为：
 - $A*B + C$
 - $A*B - C$
 - $Accum + A*B$
 - $A*B + CASI$
 - MULTALU 的 Data Options 和 Register Options 配置框的使用和 MULT 模块的类似，请参考 5.2 MULT。

3. 端口显示框图

端口显示框图显示当前 IP Core 的配置结果示例框图，输入输出端口的位宽根据 Options 配置实时更新，如图 5-4 所示。

4. Help 按钮

单击“Help”，显示 IP Core 的配置信息的页面。Help 页面包括当前 IP Core 的概要介绍，以及 Options 各项配置的简要说明。

IP 生成文件

IP 窗口配置完成后，产生以配置文件“File Name”命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_multtalu.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 MULTALU；
- IP 设计使用模板文件 gowin_multtalu_tmp.v，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_multtalu.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

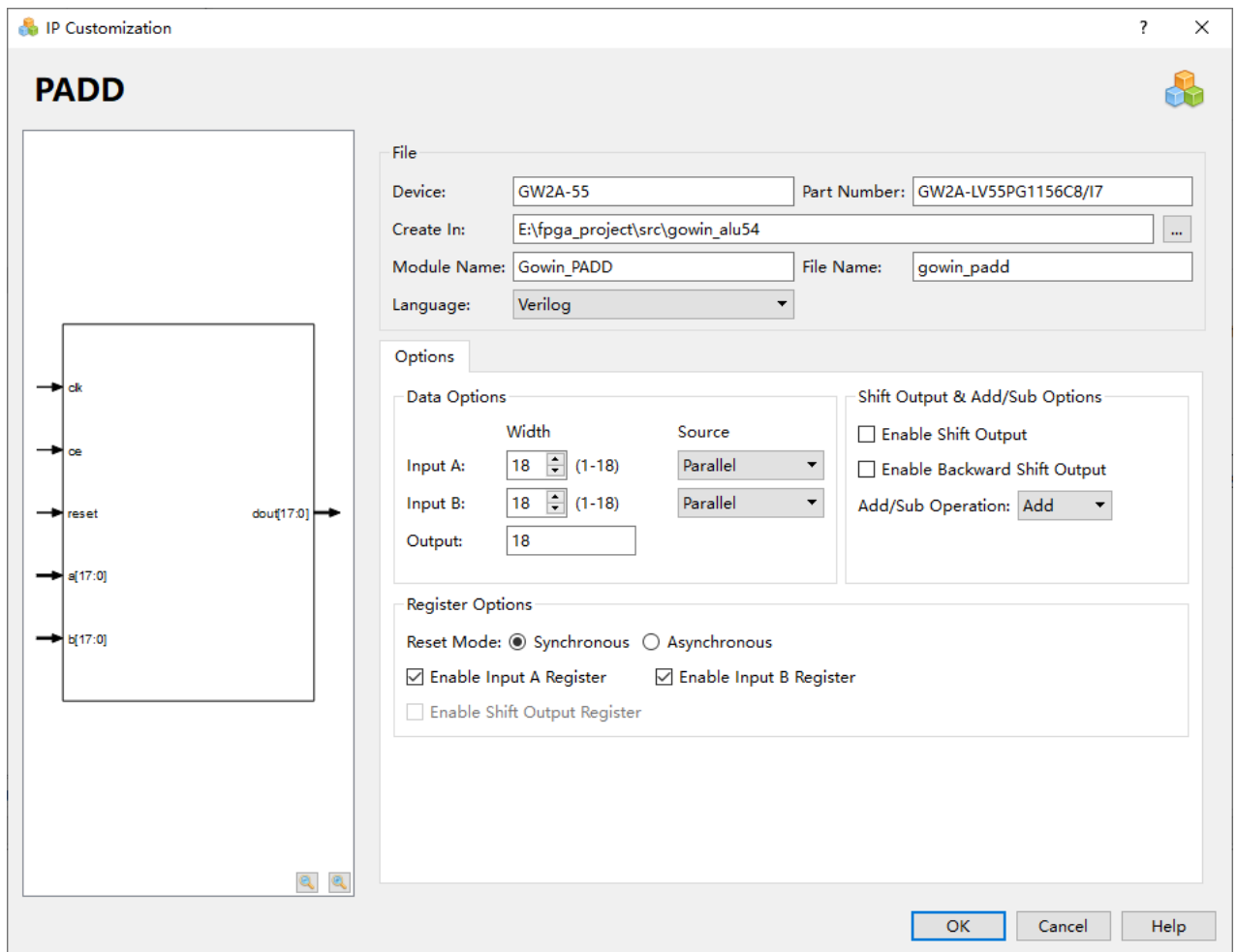
5.5 PADD

PADD 实现预加、预减或移位功能。在 IP Core Generator 界面中，单击“PADD”，界面右侧即会显示 PADD 的相关信息概要。

IP 配置

在 IP Core Generator 界面中，双击“PADD”，弹出“IP Customization”窗口。该窗口包括“File”配置框、“Options”配置框、端口显示框图和“Help”按钮，如图 5-5 所示。

图 5-5 PADD 的 IP Customization 窗口结构



1. File 配置框

- File 配置框用于配置产生的 IP 设计文件的相关信息。
- PADD 的 File 配置框的使用和 ALU54 模块的类似, 请参考 5.1 ALU54 中的 File 配置框。

2. Options 配置框

Options 配置框用于用户自定义配置 IP, Options 配置框如图 5-5 所示。

- Data Options: 配置数据选项。
 - 输入端口 (Input A Width/ Input B Width) 最大数据位宽为 18;
 - 输出端口数据位宽 (output Width) 无需用户配置, 它会根据输入位宽自动调整位宽, 例化时会根据位宽决定生成 PADD9 或 PADD18。
 - 输入端口 A 的数据来源可通过 “Input A Source” 选项配置为

Parallel 和 Shift;

- 输入端口 B 的数据来源可通过“Input B Source”选项配置为 Parallel、Backward Shift。
- **Shift Output & Add/Sub Options:** 使能 Shift Output、Backward Shift Output 和加减操作配置。
 - 使能 Shift Output 通过选中“Enable Shift Output”进行配置;
 - 使能 Backward Shift Output 通过选中“Enable Backward Shift Output”选项进行配置;
 - PADD 可通过配置“Add/Sub Operation”选项执行加法、减法。
- **Register Options:** 配置寄存器工作模式。
 - “Reset Mode”选项配置 PADD 的复位模式，支持同步模式“Synchronous”和异步模式“Asynchronous”;
 - “Enable Input A Register”配置 Input A register;
 - “Enable Input B Register”配置 Input B register;
 - “Enable Output Register”配置 Output register。

3. 端口显示框图

端口显示框图显示当前 IP Core 的配置结果示例框图，输入输出端口的个数以及位宽根据 Options 配置实时更新，如图 5-5 所示。

4. Help 按钮

单击“Help”，显示 IP Core 的配置信息的页面。Help 页面包括当前 IP Core 的概要介绍，以及 Options 各项配置的简要说明。

IP 生成文件

IP 窗口配置完成后，产生以配置文件“File Name”命名的三个文件，以默认配置为例进行介绍：

- IP 设计文件“gowin_padd.v”为完整的 verilog 模块，根据用户的 IP 配置，产生实例化的 PADD；
- IP 设计使用模板文件 gowin_padd_tmp.v，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_padd.ipc”，用户可加载该文件对 IP 进行配置。

注！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

