Arora V 数字信号处理（DSP）模块

用户指南

UG305-1.0,2023-04-20

免责声明

本文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。高云半导体对文档中包含的文字、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文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文档进行适时的更新。

版本信息

日期	版本	说明
2023/04/20	1.0	初始版本。

目录

目录 i

图目录 ii

表目录 iii

1 关于本手册 1

1.1 手册内容 1

1.2 相关文档 1

1.3 术语、缩略语 1

1.4 技术支持与反馈 2

2 概述 3

3 DSP 结构 4

4 DSP 原语 8

4.1 MULT 8

4.1.1 MULT12X12 8

4.1.2 MULT27X36 15

4.2 MULTALU 25

4.2.1 MULTALU27X18 25

4.3 MULTADDALU 50

4.3.1 MULTADDALU12X12 50

5 IP 调用 69

5.1 MULT 69

5.2 MULTALU 72

5.3 MULTADDALU 76

图目录

图 3-1 DSP 的组成结构 5
图 4-1 MULT12X12 逻辑结构示意图 9
图 4-2 MULT12X12 端口示意图 9
图 4-3 MULT27X36 逻辑结构示意图 15
图 4-4 MULT27X36 端口示意图 16
图 4-5 MULTALU27X18 逻辑结构示意图 26
图 4-6 MULTALU27X18 端口示意图 27
图 4-7 MULTADDALU12X12 逻辑结构示意图 51
图 4-8 MULTADDALU12X12 端口示意图 51
图 5-1 MULT 的 IP Customization 窗口结构 70
图 5-2 MULTALU 的 IP Customization 窗口结构 73
图 5-3 MULTADDALU 的 IP Customization 窗口结构 76

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 MULT12X12 端口介绍 10
表 4-2 MULT12X12 参数介绍 10
表 4-3 MULT27X36 端口介绍 16
表 4-4 MULT27X36 参数介绍 17
表 4-5 MULTALU27X18 端口介绍 27
表 4-6 MULTALU27X18 参数介绍 29
表 4-7 MULTADDALU12X12 端口介绍 52
表 4-8 MULTADDALU12X12 参数介绍 52

1 关于本手册

1.1 手册内容

本手册主要描述高云半导体 Arora V FPGA 产品的数字信号处理（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 Arora V DSP 的使用流程，提高设计效率。

1.2 相关文档

通过登录高云半导体网站 www.gowinsemi.com 可以下载、查看以下相关器件文档：

- DS981，GW5AT 系列 FPGA 产品数据手册
- DS1103，GW5A 系列 FPGA 产品数据手册
- DS1104，GW5AST 系列 FPGA 产品数据手册
- SUG100，Gowin 云源软件用户指南

1.3 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
CFU	Configurable Function Unit	可配置功能单元
DSP	Digital Signal Processing	数字信号处理
FIR	Finite Impulse Response	有限脉冲响应滤波器
FFT	Fast Fourier Transformation	快速傅里叶变换
MULT	Multiplier	乘法器
PADD	Pre-adder	前加器
48-bit ALU	48-bit Arithmetic Logic Unit	48 位算术逻辑单元

1.4 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 Arora V 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 Arora V DSP 的结构和使用方法。

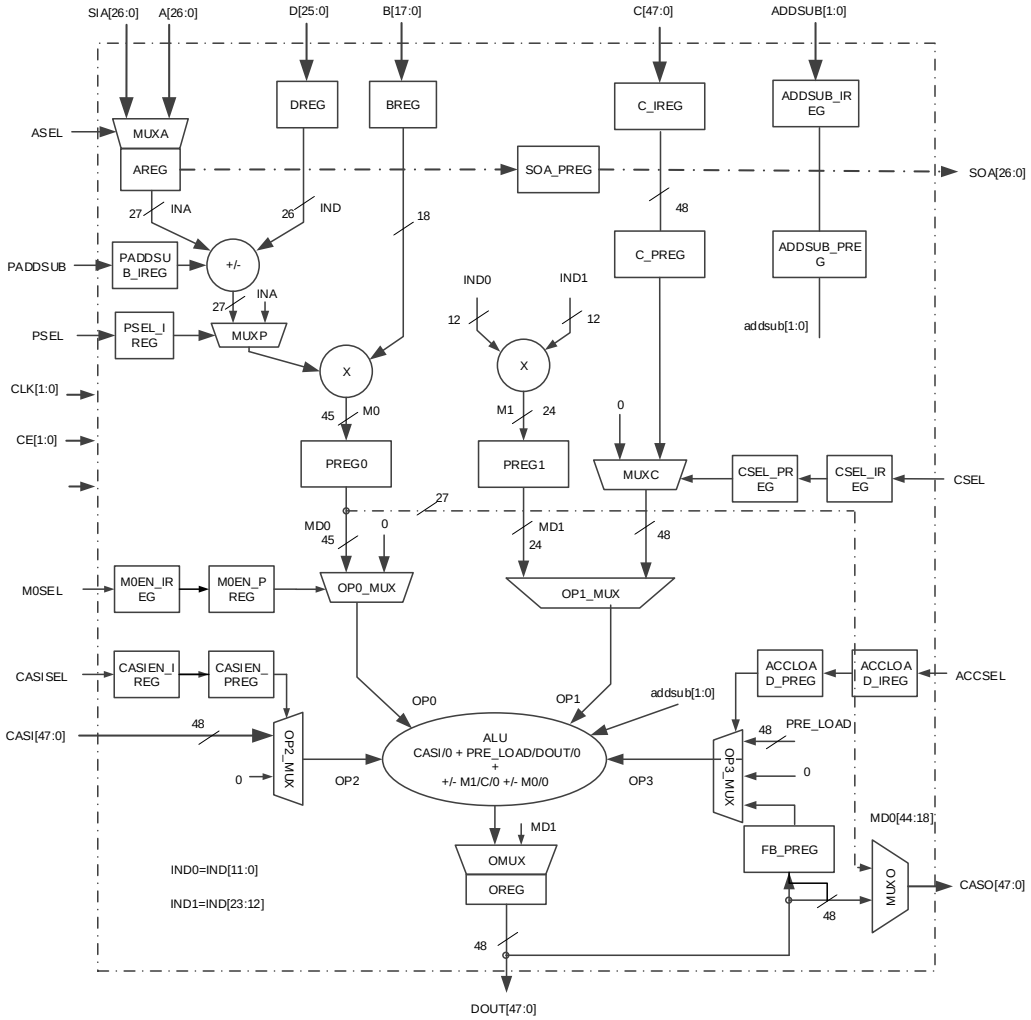
DSP 模块的功能及特性如下：

- 3 种宽度（12X12，27X18，27X36）的乘法器
- 26-bit 前加器功能
- 48-bit 的算术/逻辑运算单元
- 支持移位功能
- 多个乘法器可通过级联实现更大数据位宽的乘法
- 支持 27X18 乘法器的累加、乘加功能
- 支持两个 12X12 乘法器求和后累加功能
- 支持寄存器的流水线和旁路功能
- 数据是有符号操作

3 DSP 结构

高云半导体 Arora V FPGA 产品的 DSP 模块以行的形式分布在 FPGA 阵列中，DSP 可以实现乘法、预加、累加、移位等功能,分为 MULT、PADD 和 48-bit ALU 等功能模块。每个 DSP 占用 3 个 CFU，每个 DSP 有 2 个独立的 clock 时钟信号、2 个独立的 clock 使能信号、2 个独立的 reset 复位信号。寄存器层级最多有 4 层，即 input reg、pipe reg、out reg 和 fb preg。

图 3-1 DSP 的组成结构



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
A[26:0]	I	27-bit 数据输入 A
B[17:0]	I	18-bit 数据输入 B
C[47:0]	I	48-bit 数据输入 C
D[25:0]	I	26-bit 数据输入 D
SIA[26:0]	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。

端口名称	I/O 类型	说明
CASI[47:0]	I	来自前一个 DSP 模块的 CASO，48-bit ALU 级联输入，用于级联连接。
CASISEL	I	48-bit ALU 输入 CASI/0 控制信号
ASEL	I	前加器的 A 输入选择
PSEL	I	乘法器的 A 输入选择
PADDSUB	I	前加器的操作控制信号，用于前加器逻辑加减法选择。
CLK[1:0]	I	时钟输入
CE[1:0]	I	时钟使能信号，高电平使能。
RESET[1:0]	I	复位信号，支持同步/异步模式，高电平使能。
ADDSUB[1:0]	I	48-bit ALU 的操作控制信号，用于 M0/0,M1/C/0 的加减法选择。
CSEL	I	48-bit ALU 输入 C/0 控制信号
ACCSEL	I	48-bit ALU 输入 PRE_LOAD/DOUT 控制信号
M0SEL	I	48-bit ALU 输入 M0/0 控制信号
SOA[26:0]	O	移位数据输出 A
DOUT[47:0]	O	DSP 输出数据
CASO[47:0]	O	48-bit ALU 输出到下一个 DSP 模块进行级联连接

表 3-2 DSP 模块内部寄存器描述

寄存器	说明及相关属性
AREG	A 输入寄存器
BREG	B 输入寄存器
C_IREG	C 输入寄存器
DREG	D 输入寄存器
ADDSUB_IREG	ADDSUB 输入寄存器
PADDSUB_IREG	PADDSUB 输入寄存器
PSEL_IREG	PSEL 输入寄存器
M0SEL_IREG	M0SEL 输入寄存器
CASISEL_IREG	CASISEL 输入寄存器
CSEL_IREG	CSEL 输入寄存器
ACCSEL_IREG	ACCSEL 输入寄存器

寄存器	说明及相关属性
C_PREG	C 流水线输入寄存器
ADDSUB_PREG	ADDSUB 流水线输入寄存器
M0SEL_PREG	M0SEL 流水线输入寄存器
CASISEL_PREG	CASISEL 流水线输入寄存器
CSEL_PREG	CSEL 流水线输入寄存器
ACCSEL_PREG	ACCSEL 流水线输入寄存器
OREG	DOUT 输出寄存器
PREG0	左乘法器流水线输出寄存器
PREG1	右乘法器流水线输出寄存器
FB_PREG	反馈输出流水线寄存器
SOA_PREG	SOA 的流水线移位输出寄存器

4 DSP 原语

4.1 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：

$$DOUT = A * B$$

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

每个 DSP 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 12x12，27x36 等乘法器，分别对应原语 MULT12x12，MULT27x36。其中 27x36 乘法器需要两个 DSP 模块进行配置。

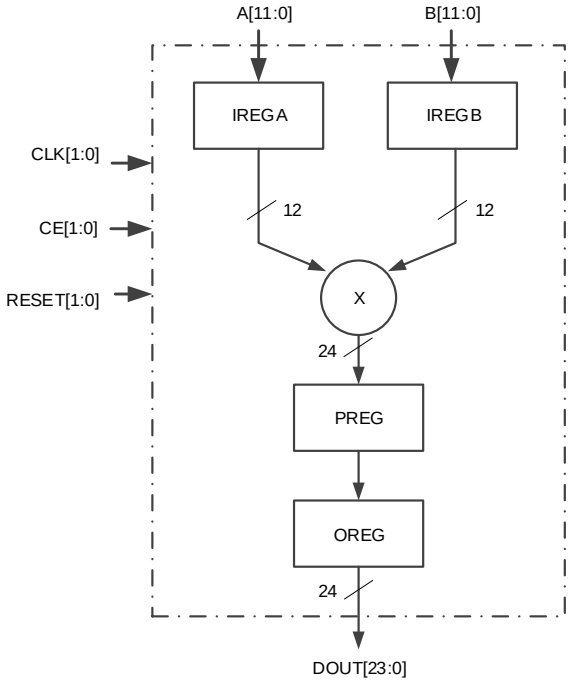
4.1.1 MULT12X12

原语介绍

MULT12X12（12x12 Multiplier）是 12x12 乘法器，实现了 12 位乘法运算。

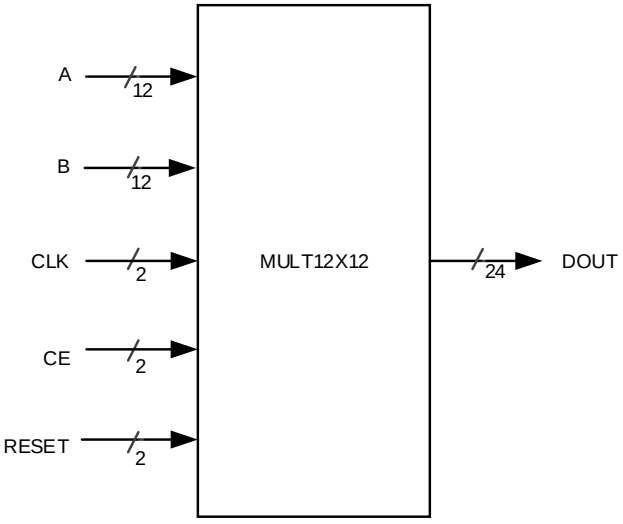
逻辑结构示意图

图 4-1 MULT12X12 逻辑结构示意图



端口示意图

图 4-2 MULT12X12 端口示意图



端口介绍

表 4-1 MULT12X12 端口介绍

端口	I/O	描述
A[11:0]	Input	12-bit 数据输入信号 A
B[11:0]	Input	12-bit 数据输入信号 B
CLK[1:0]	Input	时钟输入信号
CE[1:0]	Input	时钟使能信号，高电平有效。
RESET[1:0]	Input	复位输入信号，高电平有效。
DOUT[23:0]	Output	数据输出信号

参数介绍

表 4-2 MULT12X12 参数介绍

参数	范围	默认	描述
AREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 A 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
AREG_CE	CE0，CE1	CE0	输入 A 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
AREG_RESET	RESET0，RESET1	RESET0	输入 A 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
BREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 B 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。

参数	范围	默认	描述
BREG_CE	CE0，CE1	CE0	输入 B 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
BREG_RESET	RESET0，RESET1	RESET0	输入 B 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	Pipeline 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
PREG_CE	CE0，CE1	CE0	Pipeline 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
PREG_RESET	RESET0，RESET1	RESET0	Pipeline 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
OREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输出寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
OREG_CE	CE0，CE1	CE0	输出寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。

参数	范围	默认	描述
OREG_RESET	RESET0，RESET1	RESET0	输出寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
MULT_RESET_MODE	SYNC，ASYNC	SYNC	复位配置模式 ● 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，具体可参考第 5 章 IP 调用。

Verilog 例化：

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

defparam mult12x12_inst.AREG_CLK = "BYPASS";
defparam mult12x12_inst.AREG_CE = "CE0";
defparam mult12x12_inst.AREG_RESET = "RESET0";
defparam mult12x12_inst.BREG_CLK = "BYPASS";
defparam mult12x12_inst.BREG_CE = "CE0";
defparam mult12x12_inst.BREG_RESET = "RESET0";
defparam mult12x12_inst.PREG_CLK = "BYPASS";
defparam mult12x12_inst.PREG_CE = "CE0";
defparam mult12x12_inst.PREG_RESET = "RESET0";
```

```
defparam mult12x12_inst.OREG_CLK = "BYPASS";
defparam mult12x12_inst.OREG_CE = "CE0";
defparam mult12x12_inst.OREG_RESET = "RESET0";
defparam mult12x12_inst.MULT_RESET_MODE = "SYNC";
```

Vhdl 例化：

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

```
GENERIC MAP (
    AREG_CLK => "BYPASS",
    AREG_CE => "CE0",
    AREG_RESET => "RESET0",
    BREG_CLK => "BYPASS",
    BREG_CE => "CE0",
    BREG_RESET => "RESET0",
    PREG_CLK => "BYPASS",
    PREG_CE => "CE0",
    PREG_RESET => "RESET0",
    OREG_CLK => "BYPASS",
    OREG_CE => "CE0",
    OREG_RESET => "RESET0",
    MULT_RESET_MODE => "SYNC"
)
PORT MAP (
    DOUT => dout,
    A => a,
    B => b,
    CLK => CLK_i,
    CE => CE_i,
    RESET => RESET_i
);
```

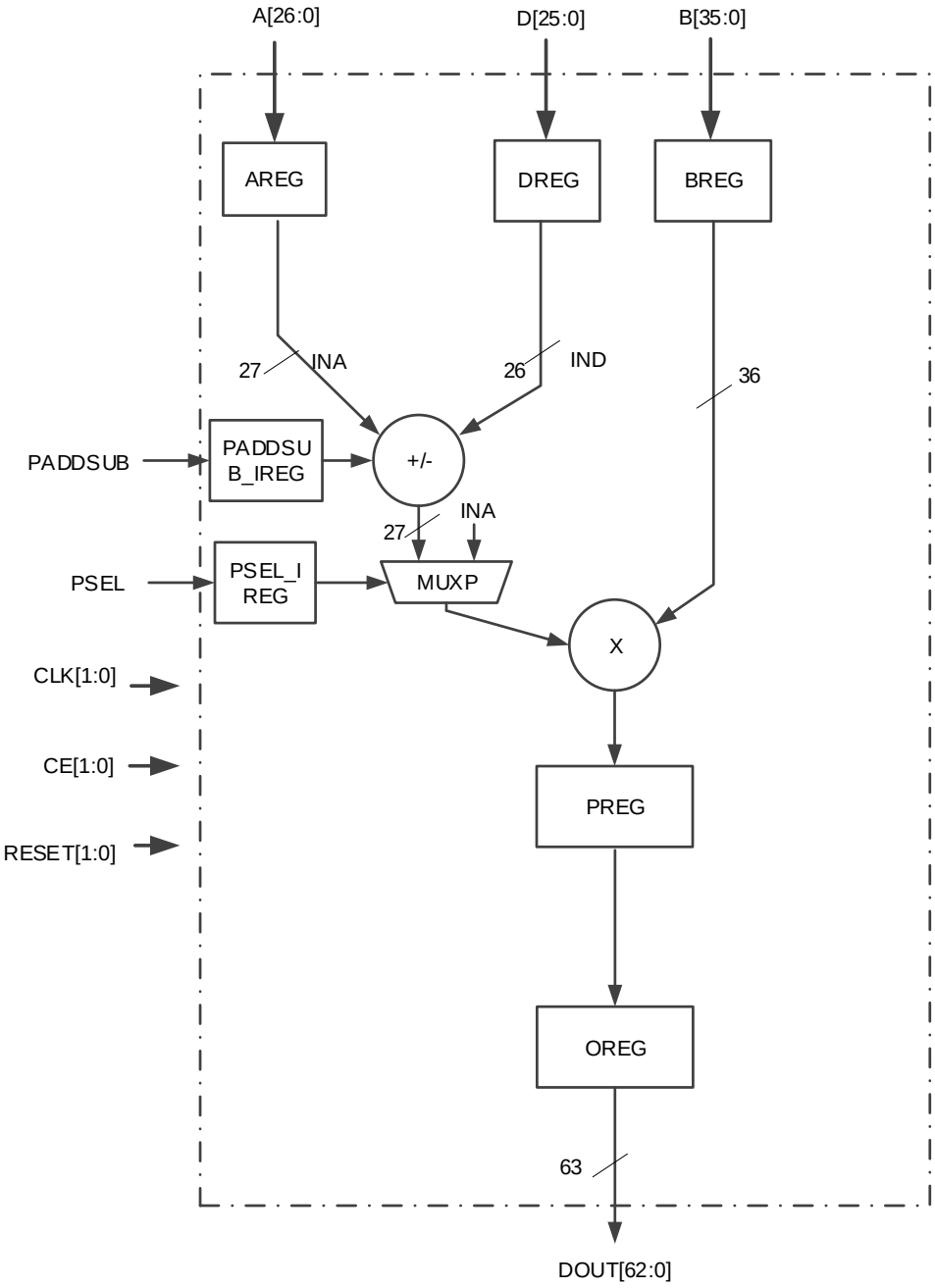
4.1.2 MULT27X36

原语介绍

MULT27X36（27x36 Multiplier）是 27x36 乘法器，实现了 27 位 X36 位的乘法运算。

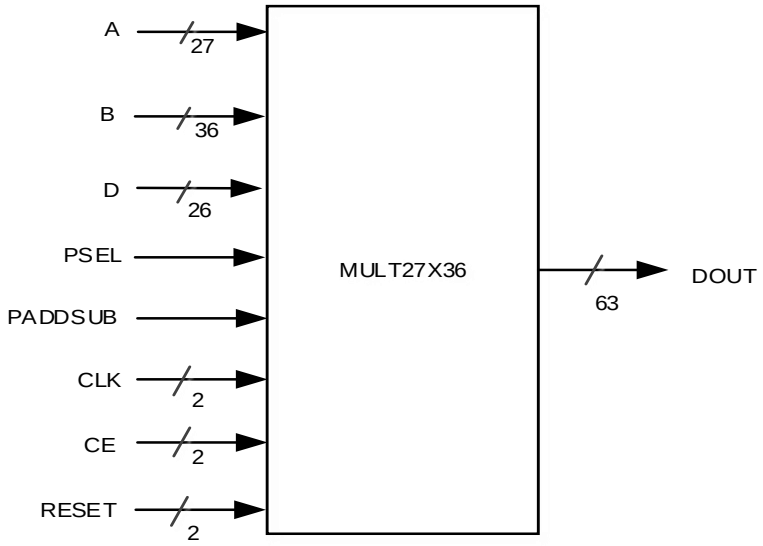
逻辑结构示意图

图 4-3 MULT27X36 逻辑结构示意图



端口示意图

图 4-4 MULT27X36 端口示意图



端口介绍

表 4-3 MULT27X36 端口介绍

端口	I/O	描述
A[26:0]	Input	27-bit 数据输入信号 A
B[35:0]	Input	36-bit 数据输入信号 B
D[25:0]	Input	26-bit 数据输入信号 D
PSEL	Input	乘法器的 A 输入源选择
PADDSUB	Input	前加器的操作控制信号，用于前加器逻辑加减法选择。
CLK[1:0]	Input	时钟输入信号
CE[1:0]	Input	时钟使能信号，高电平有效。
RESET[1:0]	Input	复位输入信号，高电平有效。
DOUT[62:0]	Output	数据输出信号

参数介绍

表 4-4 MULT27X36 参数介绍

参数	范围	默认	描述
AREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 A 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器的控制信号 clk 来自于 CLK[0]； ● CLK1：寄存器模式，寄存器的控制信号 clk 来自于 CLK[1]。
AREG_CE	CE0，CE1	CE0	输入 A 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自于 CE[0]； ● CE1：寄存器时钟使能控制信号来自于 CE[1]。
AREG_RESET	RESET0，RESET1	RESET0	输入 A 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自于 RESET[0]； ● RESET1：寄存器复位控制信号来自于 RESET[1]。
BREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 B 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器的控制信号 clk 来自于 CLK[0]； ● CLK1：寄存器模式，寄存器的控制信号 clk 来自于 CLK[1]。
BREG_CE	CE0，CE1	CE0	输入 B 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自于 CE[0]； ● CE1：寄存器时钟使能控制信号来自于 CE[1]。
BREG_RESET	RESET0，RESET1	RESET0	输入 B 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自于 RESET[0]； ● RESET1：寄存器复位控制信号来自于 RESET[1]。
DREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 D 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器的控制信号

参数	范围	默认	描述
			clk 来自于 CLK[0]； ● CLK1：寄存器模式，寄存器的控制信号 clk 来自于 CLK[1]。
DREG_CE	CE0，CE1	CE0	输入 D 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自于 CE[0]； ● CE1：寄存器时钟使能控制信号来自于 CE[1]。
DREG_RESET	RESET0，RESET1	RESET0	输入 D 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自于 RESET[0]； ● RESET1：寄存器复位控制信号来自于 RESET[1]。
PADDSUB_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 PADDSUB 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器的控制信号 clk 来自于 CLK[0]； ● CLK1：寄存器模式，寄存器的控制信号 clk 来自于 CLK[1]。
PADDSUB_IREG_CE	CE0，CE1	CE0	输入 PADDSUB 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自于 CE[0]； ● CE1：寄存器时钟使能控制信号来自于 CE[1]。
PADDSUB_IREG_RESET	RESET0，RESET1	RESET0	输入 PADDSUB 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自于 RESET[0]； ● RESET1：寄存器复位控制信号来自于 RESET[1]。
PSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 PSEL 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器的控制信号 clk 来自于 CLK[0]； ● CLK1：寄存器模式，寄存器的控制信号 clk 来自于 CLK[1]。
PSEL_IREG_CE	CE0，CE1	CE0	输入 PSEL 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自于

参数	范围	默认	描述
			CE[0]； ● CE1：寄存器时钟使能控制信号来自于 CE[1]。
PSEL_IREG_RESET	RESET0，RESET1	RESET0	输入 PSEL 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自于 RESET[0]； ● RESET1：寄存器复位控制信号来自于 RESET[1]。
PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	Mult Pipeline 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器的控制信号 clk 来自于 CLK[0]； ● CLK1：寄存器模式，寄存器的控制信号 clk 来自于 CLK[1]。
PREG_CE	CE0，CE1	CE0	Mult Pipeline 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自于 CE[0]； ● CE1：寄存器时钟使能控制信号来自于 CE[1]。
PREG_RESET	RESET0，RESET1	RESET0	Mult Pipeline 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自于 RESET[0]； ● RESET1：寄存器复位控制信号来自于 RESET[1]。
OREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输出寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器的控制信号 clk 来自于 CLK[0]； ● CLK1：寄存器模式，寄存器的控制信号 clk 来自于 CLK[1]。
OREG_CE	CE0，CE1	CE0	输出寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自于 CE[0]； ● CE1：寄存器时钟使能控制信号来自于 CE[1]。
OREG_RESET	RESET0，RESET1	RESET0	输出寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自于

参数	范围	默认	描述
			RESET[0]； ● RESET1：寄存器复位控制信号来自于 RESET[1]。
MULT_RESET_MODE	SYNC，ASYNC	SYNC	复位配置模式
P_SEL	1'b0，1'b1	1'b0	静态控制选择，直接连接 A 或 A +/- D。 ● 1'b0：选择直连 INA ● 1'b1：选择前加器
DYN_P_SEL	FALSE，TRUE	FALSE	动态控制选择 INA 或 INA +/- D ● FALSE：P_SEL 静态控制 mult0 选择 INA 或 INA +/- D ● TRUE：输入 PSEL 动态控制 mult0 选择 INA 或 INA +/- D
P_ADDSUB	1'b0，1'b1	1'b0	静态控制前加器选择加法或减法 ● 1'b0：加法 ● 1'b1：减法
DYN_P_ADDSUB	FALSE，TRUE	FALSE	动态控制前加器选择加法或减法 ● FALSE：P_ADDSUB 静态控制前加器选择加法或减法 ● TRUE：输入 PSEL 动态控制前加器选择加法或减法

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，具体可参考第 5 章 IP 调用。

Verilog 例化：

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

```
        .CE({gw_gnd,ce}),
        .RESET({gw_gnd,reset})
);

defparam mult27x36_inst.AREG_CLK = "CLK0";
defparam mult27x36_inst.AREG_CE = "CE0";
defparam mult27x36_inst.AREG_RESET = "RESET0";
defparam mult27x36_inst.BREG_CLK = "CLK0";
defparam mult27x36_inst.BREG_CE = "CE0";
defparam mult27x36_inst.BREG_RESET = "RESET0";
defparam mult27x36_inst.DREG_CLK = "CLK0";
defparam mult27x36_inst.DREG_CE = "CE0";
defparam mult27x36_inst.DREG_RESET = "RESET0";
defparam mult27x36_inst.PADDSUB_IREG_CLK = "BYPASS";
defparam mult27x36_inst.PADDSUB_IREG_CE = "CE0";
defparam mult27x36_inst.PADDSUB_IREG_RESET = "RESET0";
defparam mult27x36_inst.PREG_CLK = "BYPASS";
defparam mult27x36_inst.PREG_CE = "CE0";
defparam mult27x36_inst.PREG_RESET = "RESET0";
defparam mult27x36_inst.PSEL_IREG_CLK = "BYPASS";
defparam mult27x36_inst.PSEL_IREG_CE = "CE0";
defparam mult27x36_inst.PSEL_IREG_RESET = "RESET0";
defparam mult27x36_inst.OREG_CLK = "CLK0";
defparam mult27x36_inst.OREG_CE = "CE0";
defparam mult27x36_inst.OREG_RESET = "RESET0";
defparam mult27x36_inst.MULT_RESET_MODE = "SYNC";
defparam mult27x36_inst.DYN_P_SEL = "FALSE";
defparam mult27x36_inst.P_SEL = "1'b1";
defparam mult27x36_inst.DYN_P_ADDSUB = "FALSE";
defparam mult27x36_inst.P_ADDSUB = "1'b0";
```

Vhdl 例化：

```
COMPONENT MULT27X36
    GENERIC (AREG_CLK:string:="CLK0";
             AREG_CE:string:="CE0";
             AREG_RESET:string:="RESET0";
             BREG_CLK:string:="CLK0";
             BREG_CE:string:="CE0";
             BREG_RESET:string:="RESET0";
             DREG_CLK:string:="CLK0";
             DREG_CE:string:="CE0";
             DREG_RESET:string:="RESET0";
             PADDSUB_IREG_CLK:string:="CLK0";
             PADDSUB_IREG_CE:string:="CE0";
             PADDSUB_IREG_RESET:string:="RESET0";
             PREG_CLK:string:="CLK0";
             PREG_CE:string:="CE0";
             PREG_RESET:string:="RESET0";
             PSEL_IREG_CLK:string:="CLK0";
             PSEL_IREG_CE:string:="CE0";
             PSEL_IREG_RESET:string:="RESET0";
             OREG_CLK:string:="CLK0";
             OREG_CE:string:="CE0";
             OREG_RESET:string:="RESET0";
             MULT_RESET_MODE:string:="ASYNC";
             DYN_P_SEL:string:="FALSE";
             P_SEL:bit:='0';
             DYN_P_ADDSUB:string:="FALSE";
             P_ADDSUB:bit:='0';
    );
    PORT(
        DOUT:OUT std_logic_vector(62 downto 0);
        A:IN std_logic_vector(26 downto 0);
```

```
                B:IN std_logic_vector(35 downto 0);
                D:IN std_logic_vector(25 downto 0);
                PSEL:IN std_logic;
                PADDSUB:IN std_logic;
                CLK:IN std_logic_vector(1 downto 0);
                CE:IN std_logic_vector(1 downto 0);
                RESET:IN std_logic_vector(1 downto 0)
        );
END COMPONENT;
uut:MULT27X36
        GENERIC MAP (AREG_CLK=>"CLK0",
                        AREG_CE=>"CE0",
                        AREG_RESET=>"RESET0",

                        BREG_CLK=>"CLK0",
                        BREG_CE=>"CE0",
                        BREG_RESET=>"RESET0",
                        DREG_CLK=>"CLK0",
                        DREG_CE=>"CE0",
                        DREG_RESET=>"RESET0",
                        PADDSUB_IREG_CLK=>"CLK0",
                        PADDSUB_IREG_CE=>"CE0",
                        PADDSUB_IREG_RESET=>"RESET0",
                        PREG_CLK=>"CLK0",
                        PREG_CE=>"CE0",
                        PREG_RESET=>"RESET0",
                        PSEL_IREG_CLK=>"CLK0",
                        PSEL_IREG_CE=>"CE0",
                        PSEL_IREG_RESET=>"RESET0",
                        OREG_CLK=>"CLK0",
                        OREG_CE=>"CE0",
```

```
                    OREG_RESET=>"RESET0",
                    MULT_RESET_MODE=>"ASYNC",
                    DYN_P_SEL=>"FALSE",
                    P_SEL=>'1',
                    DYN_P_ADDSUB=>"FALSE",
                    P_ADDSUB=>'0'
    )
    PORT MAP (
        DOUT=>dout,
        A=>A_i,
        B=>b,
        D=>d,
        PSEL=>gw_gnd,
        PADDSUB=>gw_gnd,
        CLK=>clk,
        CE=>ce,
        RESET=>reset
    );
```

4.2 MULTALU

4.2.1 MULTALU27X18

MULTALU 模式实现一个乘法器输出经过 48-bit ALU 运算，对应的原语为 MULTALU27X18。MULTALU27X18 有十六种运算模式：

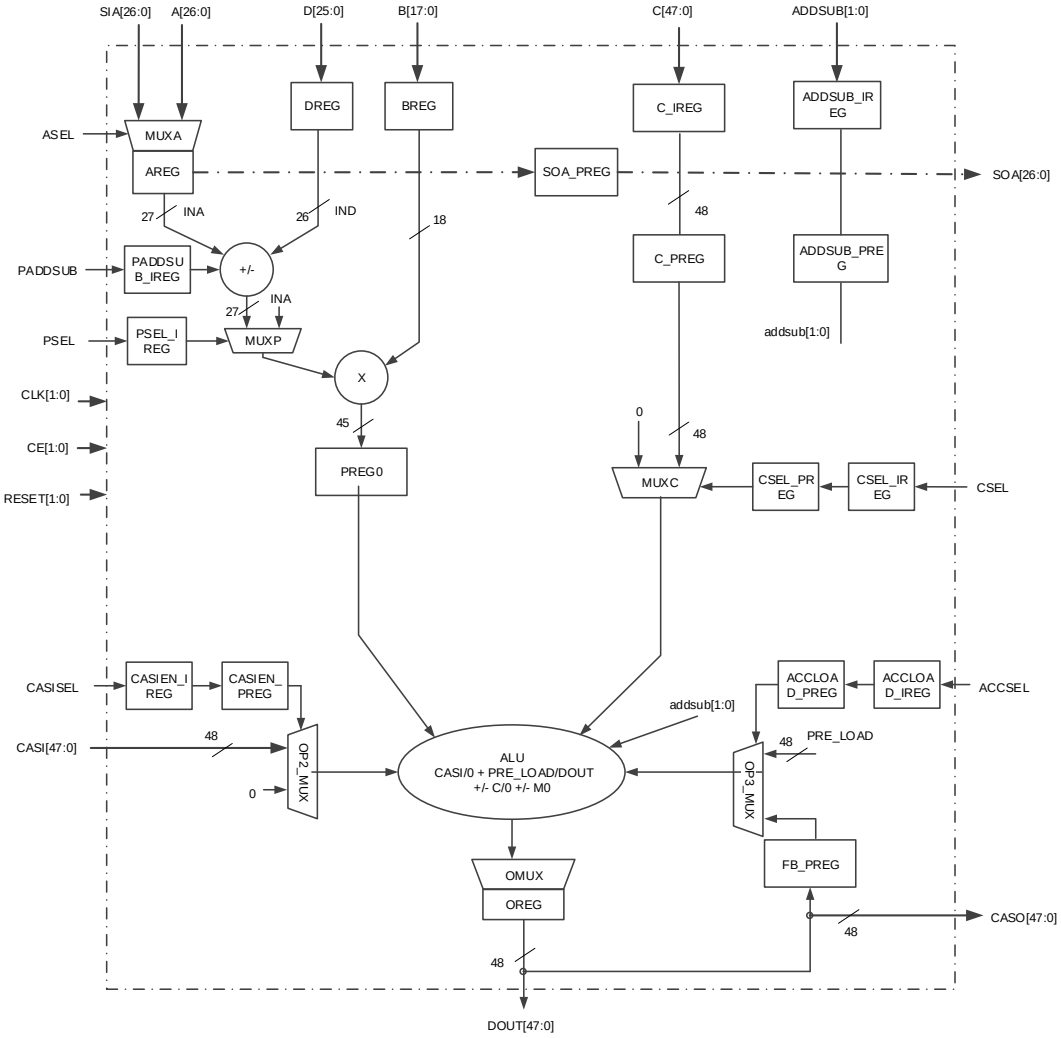
$$DOUT = \pm(A * B)$$
$$DOUT = \pm(A * B) \pm C$$
$$DOUT = \pm(A * B) + DOUT$$
$$DOUT = \pm(A * B) \pm C + DOUT$$
$$DOUT = \pm\big((A \pm D) * B\big)$$
$$DOUT = \pm\big((A \pm D) * B\big) \pm C$$
$$DOUT = \pm\big((A \pm D) * B\big) + DOUT$$
$$DOUT = \pm\big((A \pm D) * B\big) \pm C + DOUT$$
$$DOUT =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

原语介绍

MULTALU27X18（27x18 Multiplier with ALU）是带 ALU 功能的 27X18 乘法器，实现乘法、乘加、累加、乘累加、基于乘法/乘加/累加/乘累加的移位、基于乘法/乘加/累加/乘累加的级联、基于乘法/乘加/累加/乘累加的预加减等功能。

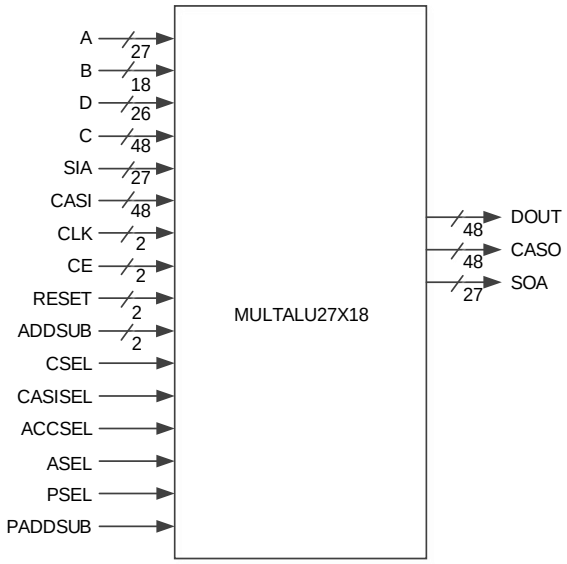
逻辑结构示意图

图 4-5 MULTALU27X18 逻辑结构示意图



端口示意图

图 4-6 MULTALU27X18 端口示意图



端口介绍

表 4-5 MULTALU27X18 端口介绍

端口	I/O	描述
A[26:0]	Input	27-bit 数据输入信号 A
B[17:0]	Input	18-bit 数据输入信号 B
D[25:0]	Input	26-bit 前加器数据输入信号 D
C[47:0]	Input	48-bit ALU 数据输入信号 C
CASI[47:0]	Input	48-bit ALU 级联输入信号
SIA[26:0]	Input	27-bit 移位数据输入信号，用于移位。输入信号 SIA 直接连接到先前相邻 DSP 的移位输出信号 SOA。
CLK[1:0]	Input	时钟输入信号
CE[1:0]	Input	时钟使能信号，高电平有效。
RESET[1:0]	Input	复位输入信号，高电平有效。
ADDSUB[1:0]	Input	动态加/减控制输入信号
CSEL	Input	48-bit ALU 输入 C，0 的控制选择输入信号
CASISEL	Input	48-bit ALU 输入 CASI，0 的控制选择输入信号
ACCSEL	Input	48-bit ALU 输入 DOUT，PRE_LOAD 的控制选择输入信号。
ASEL	Input	前加器或乘法器的 A，SIA 的控制选择输入信

端口	I/O	描述
		号。
PSEL	Input	乘法器的 INA，INA+/-D 的控制选择输入信号。
PADDSUB	Input	前加器的操作控制信号，用于前加器逻辑加减法选择。
SOA[26:0]	Output	移位数据输出信号
DOUT[47:0]	Output	数据输出信号
CASO[47:0]	Output	48-bit 级联输出信号

参数介绍

表 4-6 MULTALU27X18 参数介绍

参数	范围	默认	描述
AREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 A(A 或 SIA)寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
AREG_CE	CE0，CE1	CE0	输入 A(A 或 SIA)寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
AREG_RESET	RESET0，RESET1	RESET0	输入 A(A 或 SIA)寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
BREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 B 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
BREG_CE	CE0，CE1	CE0	输入 B 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
BREG_RESET	RESET0，RESET1	RESET0	输入 B 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
DREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 D 寄存器时钟控制信号 ● BYPASS：旁路模式；

参数	范围	默认	描述
			● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
DREG_CE	CE0，CE1	CE0	输入 D 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
DREG_RESET	RESET0，RESET1	RESET0	输入 D 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
C_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 C 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
C_IREG_CE	CE0，CE1	CE0	输入 C 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
C_IREG_RESET	RESET0，RESET1	RESET0	输入 C 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
C_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 C pipeline 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
C_PREG_CE	CE0，CE1	CE0	输入 C pipeline 寄存器时钟使能控制信

参数	范围	默认	描述
			号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
C_PREG_RESET	RESET0，RESET1	RESET0	输入 C pipeline 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
ADDSUB0_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ADDSUB[0]寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ADDSUB0_IREG_CE	CE0，CE1	CE0	输入 ADDSUB[0]寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
ADDSUB0_IREG_RESET	RESET0，RESET1	RESET0	输入 ADDSUB[0]寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
ADDSUB1_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ADDSUB[1]寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ADDSUB1_IREG_CE	CE0，CE1	CE0	输入 ADDSUB[1]寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]；

参数	范围	默认	描述
			● CE1：寄存器时钟使能控制信号来自 CE[1]。
ADDSUB1_IREG_RESET	RESET0，RESET1	RESET0	输入 ADDSUB1[1]寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
ADDSUB0_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ADDSUB[0] pipeline 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ADDSUB0_PREG_CE	CE0，CE1	CE0	输入 ADDSUB[0] pipeline 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
ADDSUB0_PREG_RESET	RESET0，RESET1	RESET0	输入 ADDSUB[0] pipeline 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
ADDSUB1_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ADDSUB[1] pipeline 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ADDSUB1_PREG_CE	CE0，CE1	CE0	输入 ADDSUB[1] pipeline 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来

参数	范围	默认	描述
			自 CE[1]。
ADDSUB1_PREG_RESET	RESET0，RESET1	RESET0	输入 ADDSUB[1] pipeline 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
PADDSUB_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 PADDSUB 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
PADDSUB_IREG_CE	CE0，CE1	CE0	输入 PADDSUB 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
PADDSUB_IREG_RESET	RESET0，RESET1	RESET0	输入 PADDSUB 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
PSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 PSEL 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
PSEL_IREG_CE	CE0，CE1	CE0	输入 PSEL 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
PSEL_IREG_RESET	RESET0，RESET1	RESET0	输入 PSEL 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自

参数	范围	默认	描述
			RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
CSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 CSEL 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
CSEL_IREG_CE	CE0，CE1	CE0	输入 CSEL 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
CSEL_IREG_RESET	RESET0，RESET1	RESET0	输入 CSEL 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
CSEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 CSEL pipeline 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
CSEL_PREG_CE	CE0，CE1	CE0	输入 CSEL pipeline 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
CSEL_PREG_RESET	RESET0，RESET1	RESET0	输入 CSEL pipeline 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
CASISEL_IREG_	BYPASS，CLK0，CLK1	BYPASS	输入 CASISEL 寄存器时钟控制信号

参数	范围	默认	描述
CLK			● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
CASISEL_IREG_CE	CE0，CE1	CE0	输入 CASISEL 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
CASISEL_IREG_RESET	RESET0，RESET1	RESET0	输入 CASISEL 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
CASISEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 CASISEL pipeline 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
CASISEL_PREG_CE	CE0，CE1	CE0	输入 CASISEL pipeline 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
CASISEL_PREG_RESET	RESET0，RESET1	RESET0	输入 CASISEL pipeline 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
ACCSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ACCSEL 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控

参数	范围	默认	描述
			制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ACCSEL_IREG_CE	CE0，CE1	CE0	输入 ACCSEL 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
ACCSEL_IREG_RESET	RESET0，RESET1	RESET0	输入 ACCSEL 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
ACCSEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ACCSEL pipeline 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ACCSEL_PREG_CE	CE0，CE1	CE0	输入 ACCSEL pipeline 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
ACCSEL_PREG_RESET	RESET0，RESET1	RESET0	输入 ACCSEL pipeline 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	M0 Pipeline 寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控

参数	范围	默认	描述
			制信号来自 CLK[1]。
PREG_CE	CE0，CE1	CE0	M0 Pipeline 寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
PREG_RESET	RESET0，RESET1	RESET0	M0 Pipeline 寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
FB_PREG_EN	FALSE，TRUE	FALSE	反馈输出 pipeline 寄存器控制参数 ● FALSE：旁路模式； ● TRUE：寄存器模式，控制信号 clk/ce/reset 与 OREG 一致。
SOA_PREG_EN	FALSE，TRUE	FALSE	移位输出 SOA pipeline 寄存器控制参数 ● FALSE：旁路模式； ● TRUE：寄存器模式，控制信号 clk/ce/reset 与 AREG 一致。
OREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输出寄存器时钟控制信号 ● BYPASS：旁路模式； ● CLK0：寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1：寄存器模式，寄存器时钟控制信号来自 CLK[1]。
OREG_CE	CE0，CE1	CE0	输出寄存器时钟使能控制信号 ● CE0：寄存器时钟使能控制信号来自 CE[0]； ● CE1：寄存器时钟使能控制信号来自 CE[1]。
OREG_RESET	RESET0，RESET1	RESET0	输出寄存器复位控制信号 ● RESET0：寄存器复位控制信号来自 RESET[0]； ● RESET1：寄存器复位控制信号来自 RESET[1]。
MULT_RESET_MODE	SYNC，ASYNC	SYNC	复位模式配置 ● SYNC：同步复位

参数	范围	默认	描述
			● ASYNC：异步复位
PRE_LOAD	48'h000000000000 ~48'hFFFFFFFFFFFF	48'h0	PRE_LOAD 初始值
A_SEL	1'b0，1'b1	1'b0	静态控制 A，SIA 源选择 ● 1'b0：选择 A ● 1'b1：选择 SIA
DYN_A_SEL	FALSE，TRUE	FALSE	动态控制 A，SIA 源选择 ● FALSE：A_SEL 静态控制 A，SIA 源选择； ● TRUE：输入 ASEL 动态控制 A，SIA 源选择。
P_SEL	1'b0，1'b1	1'b0	静态控制 INA，INA+/-D 选择 ● 1'b0：选择 INA ● 1'b1：选择 INA+/-D
DYN_P_SEL	FALSE，TRUE	FALSE	动态控制 INA，INA+/-D 选择 ● FALSE：P_SEL 静态控制 INA，INA+/-D 选择 ● TRUE：输入 PSEL 动态控制 INA，INA+/-D 选择
P_ADDSUB	1'b0，1'b1	1'b0	静态控制前加器加/减选择 ● 1'b0：加 ● 1'b1：减
DYN_P_ADDSUB	FALSE，TRUE	FALSE	动态控制前加器加/减选择 ● FALSE：P_ADDSUB 静态控制前加器加/减选择 ● TRUE：输入 PADDSUB 动态控制前加器加/减选择
ADD_SUB_0	1'b0，1'b1	1'b0	静态控制 M0，0 加/减选择 ● 1'b0：加 ● 1'b1：减
DYN_ADD_SUB_0	FALSE，TRUE	FALSE	动态控制 M0，0 加/减选择 ● FALSE：ADD_SUB_0 静态控制 M0/0 的加/减选择 ● TRUE：输入 ADDSUB[0]动态控制 M0/0 加/减选择
ADD_SUB_1	1'b0，1'b1	1'b0	静态控制 C，0 加/减选择

参数	范围	默认	描述
			● 1’b0：加 ● 1’b1：减
DYN_ADD_SUB_1	FALSE，TRUE	FALSE	动态控制 C，0 加/减选择 ● FALSE：ADD_SUB_1 静态控制 C，0 的加/减选择； ● TRUE：输入 ADDSUB[1]动态控制 M1，C，0 的加/减选择。
CASI_SEL	1’b0，1’b1	1’b0	静态控制 CASI，0 选择 ● 1’b0：选择 0 ● 1’b1：选择 CASI
DYN_CASI_SEL	FALSE，TRUE	FALSE	动态控制 CASI，0 选择 ● FALSE：CASI_SEL 静态控制 CASI，0 源选择； ● TRUE：输入 CASISEL 动态控制 CASI，0 源选择。
ACC_SEL	1’b0，1’b1	1’b0	静态控制 PRE_LOAD，DOUT 选择 ● 1’b0：选择 PRE_LOAD ● 1’b1：选择输出反馈
DYN_ACC_SEL	FALSE，TRUE	FALSE	动态控制 PRE_LOAD，DOUT 选择 ● FALSE：ACC_SEL 静态控制 PRE_LOAD，输出反馈源选择； ● TRUE：输入 ACCSEL 动态控制 PRE_LOAD，输出反馈源选择。
C_SEL	1’b0，1’b1	1’b0	静态控制 C，0 选择 ● 1’b0：选择 0 ● 1’b1：选择 C
DYN_C_SEL	FALSE，TRUE	FALSE	动态控制 C，0 选择 ● FALSE：C_SEL 静态控制 C，0 源选择； ● TRUE：输入 CSEL 动态控制 C，0 源选择。
MULT12X12_EN	FALSE，TRUE	FALSE	控制 M0 模式 ● FALSE：27X18 模式 ● TRUE：12X12 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，具体可参考第 5 章 IP 调用。

Verilog 例化：

```
MULTALU27X18 multalu27x18_inst (
    .DOUT(dout),
    .CASO(caso),
    .SOA(soa),
    .A({a[25],a[25:0]}),
    .B(b),
    .C(c),
    .D(d),
    .SIA({gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd,gw_gnd}),
    .CASI(casi),
    .ACCSEL(gw_gnd),
    .CASISEL(gw_gnd),
    .ASEL(gw_gnd),
    .PSEL(gw_gnd),
    .CSEL(gw_gnd),
    .ADDSUB({gw_gnd,gw_gnd}),
    .PADDSUB(gw_gnd),
    .CLK({gw_gnd,clk}),
    .CE({gw_gnd,ce}),
    .RESET({gw_gnd,reset})
);
defparam multalu27x18_inst.AREG_CLK = "CLK0";
defparam multalu27x18_inst.AREG_CE = "CE0";
defparam multalu27x18_inst.AREG_RESET = "RESET0";
defparam multalu27x18_inst.BREG_CLK = "CLK0";
```

```
defparam multalu27x18_inst.BREG_CE = "CE0";
defparam multalu27x18_inst.BREG_RESET = "RESET0";
defparam multalu27x18_inst.DREG_CLK = "CLK0";
defparam multalu27x18_inst.DREG_CE = "CE0";
defparam multalu27x18_inst.DREG_RESET = "RESET0";
defparam multalu27x18_inst.C_IREG_CLK = "CLK0";
defparam multalu27x18_inst.C_IREG_CE = "CE0";
defparam multalu27x18_inst.C_IREG_RESET = "RESET0";
defparam multalu27x18_inst.PSEL_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.PSEL_IREG_CE = "CE0";
defparam multalu27x18_inst.PSEL_IREG_RESET = "RESET0";
defparam multalu27x18_inst.PADDSUB_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.PADDSUB_IREG_CE = "CE0";
defparam multalu27x18_inst.PADDSUB_IREG_RESET = "RESET0";
defparam multalu27x18_inst.ADDSUB0_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.ADDSUB0_IREG_CE = "CE0";
defparam multalu27x18_inst.ADDSUB0_IREG_RESET = "RESET0";
defparam multalu27x18_inst.ADDSUB1_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.ADDSUB1_IREG_CE = "CE0";
defparam multalu27x18_inst.ADDSUB1_IREG_RESET = "RESET0";
defparam multalu27x18_inst.CSEL_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.CSEL_IREG_CE = "CE0";
defparam multalu27x18_inst.CSEL_IREG_RESET = "RESET0";
defparam multalu27x18_inst.CASISEL_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.CASISEL_IREG_CE = "CE0";
defparam multalu27x18_inst.CASISEL_IREG_RESET = "RESET0";
defparam multalu27x18_inst.ACCSEL_IREG_CLK = "BYPASS";
defparam multalu27x18_inst.ACCSEL_IREG_CE = "CE0";
defparam multalu27x18_inst.ACCSEL_IREG_RESET = "RESET0";
defparam multalu27x18_inst.PREG_CLK = "BYPASS";
defparam multalu27x18_inst.PREG_CE = "CE0";
```

```
defparam multalu27x18_inst.PREG_RESET = "RESET0";
defparam multalu27x18_inst.ADDSUB0_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.ADDSUB0_PREG_CE = "CE0";
defparam multalu27x18_inst.ADDSUB0_PREG_RESET = "RESET0";
defparam multalu27x18_inst.ADDSUB1_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.ADDSUB1_PREG_CE = "CE0";
defparam multalu27x18_inst.ADDSUB1_PREG_RESET = "RESET0";
defparam multalu27x18_inst.CSEL_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.CSEL_PREG_CE = "CE0";
defparam multalu27x18_inst.CSEL_PREG_RESET = "RESET0";
defparam multalu27x18_inst.CASISEL_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.CASISEL_PREG_CE = "CE0";
defparam multalu27x18_inst.CASISEL_PREG_RESET = "RESET0";
defparam multalu27x18_inst.ACCSEL_PREG_CLK = "BYPASS";
defparam multalu27x18_inst.ACCSEL_PREG_CE = "CE0";
defparam multalu27x18_inst.ACCSEL_PREG_RESET = "RESET0";
defparam multalu27x18_inst.C_PREG_CLK = "CLK0";
defparam multalu27x18_inst.C_PREG_CE = "CE0";
defparam multalu27x18_inst.C_PREG_RESET = "RESET0";
defparam multalu27x18_inst.FB_PREG_EN = "FALSE";
defparam multalu27x18_inst.SOA_PREG_EN = "FALSE";
defparam multalu27x18_inst.OREG_CLK = "CLK0";
defparam multalu27x18_inst.OREG_CE = "CE0";
defparam multalu27x18_inst.OREG_RESET = "RESET0";
defparam multalu27x18_inst.MULT_RESET_MODE = "SYNC";
defparam multalu27x18_inst.PRE_LOAD = 48'h000000000000;
defparam multalu27x18_inst.DYN_P_SEL = "FALSE";
defparam multalu27x18_inst.P_SEL = 1'b0;
defparam multalu27x18_inst.DYN_P_ADDSUB = "FALSE";
defparam multalu27x18_inst.P_ADDSUB = 1'b0;
defparam multalu27x18_inst.DYN_A_SEL = "FALSE";
```

```
defparam multalu27x18_inst.A_SEL = 1'b0;
defparam multalu27x18_inst.DYN_ADD_SUB_0 = "FALSE";
defparam multalu27x18_inst.ADD_SUB_0 = 1'b0;
defparam multalu27x18_inst.DYN_ADD_SUB_1 = "FALSE";
defparam multalu27x18_inst.ADD_SUB_1 = 1'b0;
defparam multalu27x18_inst.DYN_C_SEL = "FALSE";
defparam multalu27x18_inst.C_SEL = 1'b1;
defparam multalu27x18_inst.DYN_CASI_SEL = "FALSE";
defparam multalu27x18_inst.CASI_SEL = 1'b1;
defparam multalu27x18_inst.DYN_ACC_SEL = "FALSE";
defparam multalu27x18_inst.ACC_SEL = 1'b0;
defparam multalu27x18_inst.MULT12X12_EN = "FALSE";
```

Vhdl 例化：

```
COMPONENT MULTALU27X18
    GENERIC (
        AREG_CLK : string := "BYPASS";
        AREG_CE : string := "CE0";
        AREG_RESET : string := "RESET0";
        BREG_CLK : string := "BYPASS";
        BREG_CE : string := "CE0";
        BREG_RESET : string := "RESET0";
        DREG_CLK : string := "BYPASS";
        DREG_CE : string := "CE0";
        DREG_RESET : string := "RESET0";
        C_IREG_CLK : string := "BYPASS";
        C_IREG_CE : string := "CE0";
        C_IREG_RESET : string := "RESET0";
        PSEL_IREG_CLK : string := "BYPASS";
        PSEL_IREG_CE : string := "CE0";
        PSEL_IREG_RESET : string := "RESET0";
        PADDSUB_IREG_CLK : string := "BYPASS";
```

```
PADDSUB_IREG_CE : string := "CE0";
PADDSUB_IREG_RESET : string := "RESET0";
ADDSUB0_IREG_CLK : string := "BYPASS";
ADDSUB0_IREG_CE : string := "CE0";
ADDSUB0_IREG_RESET : string := "RESET0";
ADDSUB1_IREG_CLK : string := "BYPASS";
ADDSUB1_IREG_CE : string := "CE0";
ADDSUB1_IREG_RESET : string := "RESET0";
CSEL_IREG_CLK : string := "BYPASS";
CSEL_IREG_CE : string := "CE0";
CSEL_IREG_RESET : string := "RESET0";
CASISEL_IREG_CLK : string := "BYPASS";
CASISEL_IREG_CE : string := "CE0";
CASISEL_IREG_RESET : string := "RESET0";
ACCSEL_IREG_CLK : string := "BYPASS";
ACCSEL_IREG_CE : string := "CE0";
ACCSEL_IREG_RESET : string := "RESET0";
PREG_CLK : string := "BYPASS";
PREG_CE : string := "CE0";
PREG_RESET : string := "RESET0";
ADDSUB0_PREG_CLK : string := "BYPASS";
ADDSUB0_PREG_CE : string := "CE0";
ADDSUB0_PREG_RESET : string := "RESET0";
ADDSUB1_PREG_CLK : string := "BYPASS";
ADDSUB1_PREG_CE : string := "CE0";
ADDSUB1_PREG_RESET : string := "RESET0";
CSEL_PREG_CLK : string := "BYPASS";
CSEL_PREG_CE : string := "CE0";
CSEL_PREG_RESET : string := "RESET0";
CASISEL_PREG_CLK : string := "BYPASS";
CASISEL_PREG_CE : string := "CE0";
```

```
CASISEL_PREG_RESET : string := "RESET0";
ACCSEL_PREG_CLK : string := "BYPASS";
ACCSEL_PREG_CE : string := "CE0";
ACCSEL_PREG_RESET : string := "RESET0";
C_PREG_CLK : string := "BYPASS";
C_PREG_CE : string := "CE0";
C_PREG_RESET : string := "RESET0";
FB_PREG_EN : string := "FALSE";
SOA_PREG_EN : string := "FALSE";
OREG_CLK : string := "BYPASS";
OREG_CE : string := "CE0";
OREG_RESET : string := "RESET0";
MULT_RESET_MODE : string := "SYNC";
PRE_LOAD : bit_vector := X"000000000000";
DYN_P_SEL : string := "FALSE";
P_SEL : bit := '0';
DYN_P_ADDSUB : string := "FALSE";
P_ADDSUB : bit := '0';
DYN_A_SEL : string := "FALSE";
A_SEL : bit := '0';
DYN_ADD_SUB_0 : string := "FALSE";
ADD_SUB_0 : bit := '0';
DYN_ADD_SUB_1 : string := "FALSE";
ADD_SUB_1 : bit := '0';
DYN_C_SEL : string := "FALSE";
C_SEL : bit := '0';
DYN_CASI_SEL : string := "FALSE";
CASI_SEL : bit := '0';
DYN_ACC_SE : string := "FALSE";
ACC_SEL : bit := '0';
MULT12X12_EN : string := "FALSE"
```

```
        );
        PORT (
            DOUT: out std_logic_vector(47 downto 0);
            CASO: out std_logic_vector(47 downto 0);
            SOA: out std_logic_vector(26 downto 0);
            A: in std_logic_vector(26 downto 0);
            B: in std_logic_vector(17 downto 0);
            C: in std_logic_vector(47 downto 0);
            D: in std_logic_vector(25 downto 0);
            SIA: in std_logic_vector(26 downto 0);
            CASI: in std_logic_vector(47 downto 0);
            ACCSEL: in std_logic;
            CASISEL: in std_logic;
            ASEL: in std_logic;
            PSEL: in std_logic;
            CSEL: in std_logic;
            ADDSUB: in std_logic_vector(1 downto 0);
            PADDSUB: in std_logic;
            CLK: in std_logic_vector(1 downto 0);
            CE: in std_logic_vector(1 downto 0);
            RESET: in std_logic_vector(1 downto 0)
        );
    end COMPONENT;
begin
    gw_gnd <= '0';

    A_i <= a[25] & a(25 downto 0);
    SIA_i <= gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd &
gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd &
gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd &
gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd & gw_gnd &
gw_gnd;
```

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

multalu27x18_inst: MULTALU27X18
    GENERIC MAP (
        AREG_CLK => "CLK0",
        AREG_CE => "CE0",
        AREG_RESET => "RESET0",
        BREG_CLK => "CLK0",
        BREG_CE => "CE0",
        BREG_RESET => "RESET0",
        DREG_CLK => "CLK0",
        DREG_CE => "CE0",
        DREG_RESET => "RESET0",
        C_IREG_CLK => "CLK0",
        C_IREG_CE => "CE0",
        C_IREG_RESET => "RESET0",
        PSEL_IREG_CLK => "BYPASS",
        PSEL_IREG_CE => "CE0",
        PSEL_IREG_RESET => "RESET0",
        PADDSUB_IREG_CLK => "BYPASS",
        PADDSUB_IREG_CE => "CE0",
        PADDSUB_IREG_RESET => "RESET0",
        ADDSUB0_IREG_CLK => "BYPASS",
        ADDSUB0_IREG_CE => "CE0",
        ADDSUB0_IREG_RESET => "RESET0",
        ADDSUB1_IREG_CLK => "BYPASS",
        ADDSUB1_IREG_CE => "CE0",
        ADDSUB1_IREG_RESET => "RESET0",
```

```
CSEL_IREG_CLK => "BYPASS",
CSEL_IREG_CE => "CE0",
CSEL_IREG_RESET => "RESET0",
CASISEL_IREG_CLK => "BYPASS",
CASISEL_IREG_CE => "CE0",
CASISEL_IREG_RESET => "RESET0",
ACCSEL_IREG_CLK => "BYPASS",
ACCSEL_IREG_CE => "CE0",
ACCSEL_IREG_RESET => "RESET0",
PREG_CLK => "BYPASS",
PREG_CE => "CE0",
PREG_RESET => "RESET0",
ADDSUB0_PREG_CLK => "BYPASS",
ADDSUB0_PREG_CE => "CE0",
ADDSUB0_PREG_RESET => "RESET0",
ADDSUB1_PREG_CLK => "BYPASS",
ADDSUB1_PREG_CE => "CE0",
ADDSUB1_PREG_RESET => "RESET0",
CSEL_PREG_CLK => "BYPASS",
CSEL_PREG_CE => "CE0",
CSEL_PREG_RESET => "RESET0",
CASISEL_PREG_CLK => "BYPASS",
CASISEL_PREG_CE => "CE0",
CASISEL_PREG_RESET => "RESET0",
ACCSEL_PREG_CLK => "BYPASS",
ACCSEL_PREG_CE => "CE0",
ACCSEL_PREG_RESET => "RESET0",
C_PREG_CLK => "CLK0",
C_PREG_CE => "CE0",
C_PREG_RESET => "RESET0",
FB_PREG_EN => "FALSE",
```

```
        SOA_PREG_EN => "FALSE",
        OREG_CLK => "CLK0",
        OREG_CE => "CE0",
        OREG_RESET => "RESET0",
        MULT_RESET_MODE => "SYNC",
        PRE_LOAD => X"000000000000",
        DYN_P_SEL => "FALSE",
        P_SEL => '0',
        DYN_P_ADDSUB => "FALSE",
        P_ADDSUB => '0',
        DYN_A_SEL => "FALSE",
        A_SEL => '0',
        DYN_ADD_SUB_0 => "FALSE",
        ADD_SUB_0 => '0',
        DYN_ADD_SUB_1 => "FALSE",
        ADD_SUB_1 => '0',
        DYN_C_SEL => "FALSE",
        C_SEL => '1',
        DYN_CASI_SEL => "FALSE",
        CASI_SEL => '1',
        DYN_ACC_SEL => "FALSE",
        ACC_SEL => '0',
        MULT12X12_EN => "FALSE"
    )
    PORT MAP (
        DOUT => dout,
        CASO => caso,
        SOA => soa,
        A => A_i,
        B => b,
        C => c,
```

```
    D => d,
    SIA => SIA_i,
    CASI => casi,
    ACCSEL => gw_gnd,
    CASISEL => gw_gnd,
    ASEL => gw_gnd,
    PSEL => gw_gnd,
    CSEL => gw_gnd,
    ADDSUB => ADDSUB_i,
    PADDSUB => gw_gnd,
    CLK => CLK_i,
    CE => CE_i,
    RESET => RESET_i
);
```

4.3 MULTADDALU

4.3.1 MULTADDALU12X12

MULTADDALU 模式实现两个 12 x 12 乘法器输出经过 48-bit ALU 运算，对应原语为 MULTADDALU12x12。

MULTADDALU12x12 有四种运算模式：

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

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

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

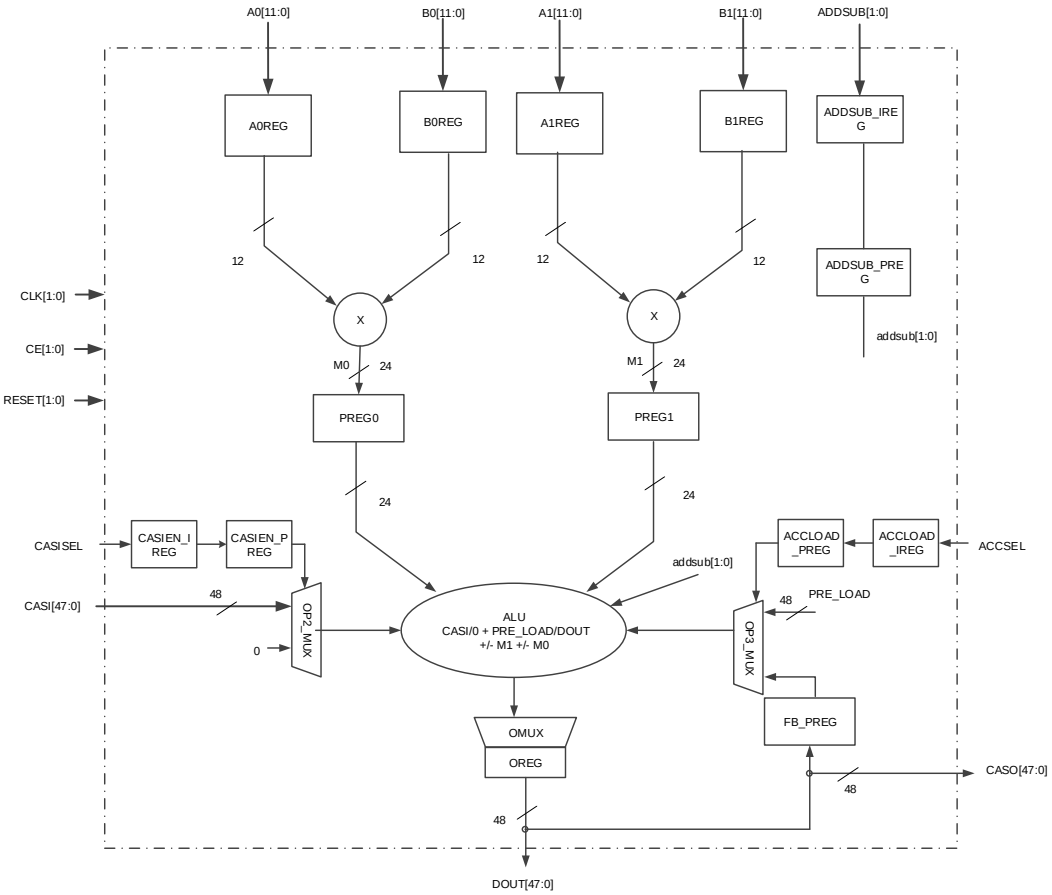
$$DOUT = CASI \pm (A0 * B0 \pm A1 * B1) + DOUT$$

原语介绍

MULTADDALU12x12（The Sum of Two 12x12 Multipliers with ALU）是带 ALU 功能的 12x12 乘加器，实现 12 位的乘法求和后累加运算。

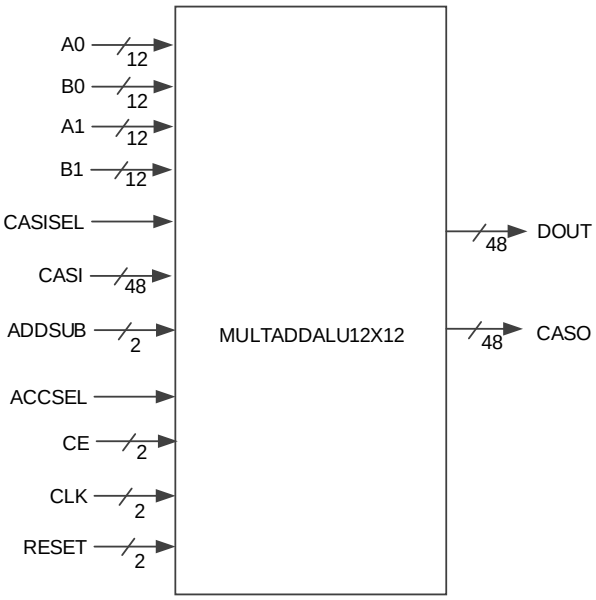
逻辑结构示意图

图 4-7 MULTADDALU12X12 逻辑结构示意图



端口示意图

图 4-8 MULTADDALU12X12 端口示意图



端口介绍

表 4-7 MULTADDALU12X12 端口介绍

端口	I/O	描述
A0[11:0]	Input	12-bit 数据输入信号 A0
B0[11:0]	Input	12-bit 数据输入信号 B0
A1[11:0]	Input	12-bit 数据输入信号 A1
B1[11:0]	Input	12-bit 数据输入信号 B1
CASI[47:0]	Input	自前一级 DSP 的 48-bit 级联输入信号
CASISEL	Input	48-bit ALU 的 CASI/0 的控制输入信号
ADDSUB[1:0]	Input	动态+/-控制输入信号
ACCSEL	Input	48-bit ALU 的 DOUT/PRE_LOAD 的控制输入信号
CLK[1:0]	Input	时钟输入信号
CE[1:0]	Input	时钟使能信号
RESET[1:0]	Input	复位输入信号
DOUT[53:0]	Output	数据输出信号
CASO[54:0]	Output	48-bit 级联输出信号

参数介绍

表 4-8 MULTADDALU12X12 参数介绍

参数	范围	默认	描述
A0REG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 A0 寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
A0REG_CE	CE0，CE1	CE0	输入 A0 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
A0REG_RESET	RESET0，RESET1	RESET0	输入 A0 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自

参数	范围	默认	描述
			RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
B0REG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 B0 寄存器时钟控制信号 ● BYPASS: 旁路模式 ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
B0REG_CE	CE0，CE1	CE0	输入 B0 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
B0REG_RESET	RESET0，RESET1	RESET0	输入 B0 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
A1REG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 A1 寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
A1REG_CE	CE0，CE1	CE0	输入 A1 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
A1REG_RESET	RESET0，RESET1	RESET0	输入 A1 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
B1REG_CLK	BYPASS，CLK0，CLK1	CLK0	输入 B1 寄存器时钟控制信号 ● BYPASS: 旁路模式；

参数	范围	默认	描述
			● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
B1REG_CE	CE0，CE1	CE0	输入 B1 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
B1REG_RESET	RESET0，RESET1	RESET0	输入 B1 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
ADDSUB0_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ADDSUB[0]寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ADDSUB0_IREG_CE	CE0，CE1	CE0	输入 ADDSUB[0]寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
ADDSUB0_IREG_RESET	RESET0，RESET1	RESET0	输入 ADDSUB[0]寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
ADDSUB1_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ADDSUB[1]寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ADDSUB1_IREG	CE0，CE1	CE0	输入 ADDSUB[1]寄存器时钟使能控制信号

参数	范围	默认	描述
_CE			● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
ADDSUB1_IREG_RESET	RESET0，RESET1	RESET0	输入 ADDSUB[1]寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
ADDSUB0_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ADDSUB[0] Pipeline 寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ADDSUB0_PREG_CE	CE0，CE1	CE0	输入 ADDSUB[0] Pipeline 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
ADDSUB0_PREG_RESET	RESET0.RESET1	RESET0	输入 ADDSUB[0] Pipeline 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
ADDSUB1_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ADDSUB[1] Pipeline 寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ADDSUB1_PREG_CE	CE0，CE1	CE0	输入 ADDSUB[1] Pipeline 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自

参数	范围	默认	描述
			CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
ADDSUB1_PREG_RESET	RESET0，RESET1	RESET0	输入 ADDSUB[1] Pipeline 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
CASISEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 CASISEL 寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
CASISEL_IREG_CE	CE0，CE1	CE0	输入 CASISEL 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
CASISEL_IREG_RESET	RESET0，RESET1	RESET0	输入 CASISEL 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
CASISEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 CASISEL Pipeline 寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]。 ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
CASISEL_PREG_CE	CE0，CE1	CE0	输入 CASISEL Pipeline 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。

参数	范围	默认	描述
CASISEL_PREG_RESET	RESET0，RESET1	RESET0	输入 CASISEL Pipeline 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
ACCSEL_IREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ACCSEL 寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ACCSEL_IREG_CE	CE0，CE1	CE0	输入 ACCSELSEL 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
ACCSEL_IREG_RESET	RESET0，RESET1	RESET0	输入 ACCSEL 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
ACCSEL_PREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输入 ACCSEL Pipeline 寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
ACCSEL_PREG_CE	CE0，CE1	CE0	输入 ACCSEL Pipeline 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
ACCSEL_PREG_RESET	RESET0，RESET1	RESET0	输入 ACCSEL Pipeline 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自

参数	范围	默认	描述
			RESET[1]。
PREG0_CLK	BYPASS，CLK0，CLK1	BYPASS	Mult0 Pipeline 寄存器时钟控制信号. ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
PREG0_CE	CE0，CE1	CE0	Mult0 Pipeline 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
PREG0_RESET	RESET0，RESET1	RESET0	Mult0 Pipeline 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
PREG1_CLK	BYPASS，CLK0，CLK1	BYPASS	Mult1 Pipeline 寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
PREG1_CE	CE0，CE1	CE0	Mult1 Pipeline 寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
PREG1_RESET	RESET0，RESET1	RESET0	Mult1 Pipeline 寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
FB_PREG_EN	FALSE，TRUE	FALSE	反馈输出 Pipeline 寄存器模式控制参数 ● FALSE: 旁路模式； ● TRUE: 该寄存器使能，控制信号 clk/ce/reset 与 OREG 一致。

参数	范围	默认	描述
OREG_CLK	BYPASS，CLK0，CLK1	BYPASS	输出寄存器时钟控制信号 ● BYPASS: 旁路模式； ● CLK0: 寄存器模式，寄存器时钟控制信号来自 CLK[0]； ● CLK1: 寄存器模式，寄存器时钟控制信号来自 CLK[1]。
OREG_CE	CE0，CE1	CE0	输出寄存器时钟使能控制信号 ● CE0: 寄存器时钟使能控制信号来自 CE[0]； ● CE1: 寄存器时钟使能控制信号来自 CE[1]。
OREG_RESET	RESET0，RESET1	RESET0	输出寄存器复位控制信号 ● RESET0: 寄存器复位控制信号来自 RESET[0]； ● RESET1: 寄存器复位控制信号来自 RESET[1]。
MULT_RESET_MODE	SYNC，ASYNC	SYNC	同步或异步
PRE_LOAD	48bits value	48’h0	PRE_LOAD 初始化值
ADD_SUB_0	1’b0，1’b1	1’b0	静态控制 M0/0 的加/减选择 ● 1’b0: 加 ● 1’b1: 减
DYN_ADD_SUB_0	FALSE，TRUE	FALSE	动态控制 M0/0 的加/减选择 ● FALSE: 由 ADD_SUB_0 静态控制 M0/0 的加/减选择； ● TRUE: 由输入 ADDSUB[0]动态控制 M0/0 的加/减选择。
ADD_SUB_1	1’b0，1’b1	1’b0	静态控制 M1/0 的加/减选择 ● 1’b0: 加 ● 1’b1: 减
DYN_ADD_SUB_1	FALSE，TRUE	FALSE	动态控制 M1/0 的加/减选择 ● FALSE: 由 ADD_SUB_1 静态控制 M1/0 的加/减选择； ● TRUE: 由输入 ADDSUB[1]动态控制 M1/0 的加/减选择。
CASI_SEL	1’b0，1’b1	1’b0	静态控制 CASI/0 选择

参数	范围	默认	描述
			● 1’b0: 0 ● 1’b1: CASI
DYN_CASI_SEL	FALSE，TRUE	FALSE	动态控制 CASI/0 选择 ● FALSE: 由 CASI_SEL 静态控制 CASI/0 选择； ● TRUE: 由输入 CASISEL 动态控制 CASI/0 选择。
ACC_SEL	1’b0，1’b1	1’b0	静态控制 PRE_LOAD/DOUT 选择. ● 1’b0：PRE_LOAD ● 1’b1：反馈的 DOUT
DYN_ACC_SEL	FALSE，TRUE	FALSE	动态控制 PRE_LOAD/DOUT 选择 ● FALSE: 由 ACC_SEL 静态控制 PRE_LOAD/DOUT 选择 ● TRUE: 由输入 ACCSEL 动态控制 PRE_LOAD/DOUT 选择

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，具体可参考第 5 章 IP 调用。

Verilog 例化：

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

```
        .RESET({gw_gnd,reset})
);

defparam multaddalu12x12_inst.A0REG_CLK = "CLK0";
defparam multaddalu12x12_inst.A0REG_CE = "CE0";
defparam multaddalu12x12_inst.A0REG_RESET = "RESET0";
defparam multaddalu12x12_inst.A1REG_CLK = "CLK0";
defparam multaddalu12x12_inst.A1REG_CE = "CE0";
defparam multaddalu12x12_inst.A1REG_RESET = "RESET0";
defparam multaddalu12x12_inst.B0REG_CLK = "CLK0";
defparam multaddalu12x12_inst.B0REG_CE = "CE0";
defparam multaddalu12x12_inst.B0REG_RESET = "RESET0";
defparam multaddalu12x12_inst.B1REG_CLK = "CLK0";
defparam multaddalu12x12_inst.B1REG_CE = "CE0";
defparam multaddalu12x12_inst.B1REG_RESET = "RESET0";
defparam multaddalu12x12_inst.ACCSEL_IREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ACCSEL_IREG_CE = "CE0";
defparam multaddalu12x12_inst.ACCSEL_IREG_RESET = "RESET0";
defparam multaddalu12x12_inst.CASISEL_IREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.CASISEL_IREG_CE = "CE0";
defparam multaddalu12x12_inst.CASISEL_IREG_RESET = "RESET0";
defparam multaddalu12x12_inst.ADDSUB0_IREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ADDSUB0_IREG_CE = "CE0";
defparam multaddalu12x12_inst.ADDSUB0_IREG_RESET = "RESET0";
defparam multaddalu12x12_inst.ADDSUB1_IREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ADDSUB1_IREG_CE = "CE0";
defparam multaddalu12x12_inst.ADDSUB1_IREG_RESET = "RESET0";
defparam multaddalu12x12_inst.PREG0_CLK = "BYPASS";
defparam multaddalu12x12_inst.PREG0_CE = "CE0";
defparam multaddalu12x12_inst.PREG0_RESET = "RESET0";
defparam multaddalu12x12_inst.PREG1_CLK = "BYPASS";
```

```
defparam multaddalu12x12_inst.PREG1_CE = "CE0";
defparam multaddalu12x12_inst.PREG1_RESET = "RESET0";
defparam multaddalu12x12_inst.FB_PREG_EN = "FALSE";
defparam multaddalu12x12_inst.ACCSEL_PREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ACCSEL_PREG_CE = "CE0";
defparam multaddalu12x12_inst.ACCSEL_PREG_RESET = "RESET0";
defparam multaddalu12x12_inst.CASISEL_PREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.CASISEL_PREG_CE = "CE0";
defparam multaddalu12x12_inst.CASISEL_PREG_RESET = "RESET0";
defparam multaddalu12x12_inst.ADDSUB0_PREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ADDSUB0_PREG_CE = "CE0";
defparam multaddalu12x12_inst.ADDSUB0_PREG_RESET =
"RESET0";
defparam multaddalu12x12_inst.ADDSUB1_PREG_CLK = "BYPASS";
defparam multaddalu12x12_inst.ADDSUB1_PREG_CE = "CE0";
defparam multaddalu12x12_inst.ADDSUB1_PREG_RESET =
"RESET0";
defparam multaddalu12x12_inst.OREG_CLK = "CLK0";
defparam multaddalu12x12_inst.OREG_CE = "CE0";
defparam multaddalu12x12_inst.OREG_RESET = "RESET0";
defparam multaddalu12x12_inst.MULT_RESET_MODE = "SYNC";
defparam multaddalu12x12_inst.PRE_LOAD = 48'h000000000000;
defparam multaddalu12x12_inst.DYN_ADD_SUB_0 = "FALSE";
defparam multaddalu12x12_inst.ADD_SUB_0 = 1'b0;
defparam multaddalu12x12_inst.DYN_ADD_SUB_1 = "FALSE";
defparam multaddalu12x12_inst.ADD_SUB_1 = 1'b0;
defparam multaddalu12x12_inst.DYN_CASI_SEL = "FALSE";
defparam multaddalu12x12_inst.CASI_SEL = 1'b1;
defparam multaddalu12x12_inst.DYN_ACC_SEL = "FALSE";
defparam multaddalu12x12_inst.ACC_SEL = 1'b0;
```

Vhdl 例化：

```
COMPONENT MULTADDALU12X12
    GENERIC (
        A0REG_CLK : string := "BYPASS";
        A0REG_CE : string := "CE0";
        A0REG_RESET : string := "RESET0";
        A1REG_CLK : string := "BYPASS";
        A1REG_CE : string := "CE0";
        A1REG_RESET : string := "RESET0";
        B0REG_CLK : string := "BYPASS";
        B0REG_E : string := "CE0";
        B0REG_RESET : string := "RESET0";
        B1REG_CLK : string := "BYPASS";
        B1REG_CE : string := "CE0";
        B1REG_RESET : string := "RESET0";
        ACCSEL_IREG_CLK : string := "BYPASS";
        ACCSEL_IREG_CE : string := "CE0";
        ACCSEL_IREG_RESET : string := "RESET0";
        CASISEL_IREG_CLK : string := "BYPASS";
        CASISEL_IREG_CE : string := "CE0";
        CASISEL_IREG_RESET : string := "RESET0";
        ADDSUB0_IREG_CLK : string := "BYPASS";
        ADDSUB0_IREG_CE : string := "CE0";
        ADDSUB0_IREG_RESET : string := "RESET0";
        ADDSUB1_IREG_CLK : string := "BYPASS";
        ADDSUB1_IREG_CE : string := "CE0";
        ADDSUB1_IREG_RESET : string := "RESET0";
        PREG0_CLK : string := "BYPASS";
        PREG0_CE : string := "CE0";
        PREG0_RESET : string := "RESET0";
        PREG1_CLK : string := "BYPASS";
        PREG1_CE : string := "CE0";
```

```
        PREG1_RESET : string := "RESET0";
        FB_PREG_EN : string := "FALSE";
        ACCSEL_PREG_CLK : string := "BYPASS";
        ACCSEL_PREG_CE : string := "CE0";
        ACCSEL_PREG_RESET : string := "RESET0";
        CASISEL_PREG_CLK : string := "BYPASS";
        CASISEL_PREG_CE : string := "CE0";
        CASISEL_PREG_RESET : string := "RESET0";
        ADDSUB0_PREG_CLK : string := "BYPASS";
        ADDSUB0_PREG_CE : string := "CE0";
        ADDSUB0_PREG_RESET : string := "RESET0";
        ADDSUB1_PREG_CLK : string := "BYPASS";
        ADDSUB1_PREG_CE : string := "CE0";
        ADDSUB1_PREG_RESET : string := "RESET0";
        OREG_CLK : string := "BYPASS";
        OREG_CE : string := "CE0";
        OREG_RESET : string := "RESET0";
        MULT_RESET_MODE : string := "SYNC";
        PRE_LOAD : bit_vector := X"000000000000";
        DYN_ADD_SUB_0 : string := "FALSE";
        ADD_SUB_0 : bit := '0';
        DYN_ADD_SUB_1 : string := "FALSE";
        ADD_SUB_1 : bit := '0';
        DYN_CASI_SEL : string := "FALSE";
        CASI_SEL : bit := '0';
        DYN_ACC_SE : string := "FALSE";
        ACC_SEL : bit := '0';
    );
    PORT (
        DOUT: out std_logic_vector(47 downto 0);
        CASO: out std_logic_vector(47 downto 0);
```

```
            A0: in std_logic_vector(11 downto 0);
            B0: in std_logic_vector(11 downto 0);
            A1: in std_logic_vector(11 downto 0);
            B1: in std_logic_vector(11 downto 0);
            CASI: in std_logic_vector(47 downto 0);
            ACCSEL: in std_logic;
            CASISEL: in std_logic;
            ADDSUB: in std_logic_vector(1 downto 0);
            CLK: in std_logic_vector(1 downto 0);
            CE: in std_logic_vector(1 downto 0);
            RESET: in std_logic_vector(1 downto 0)
        );
    end COMPONENT;
begin
    gw_gnd <= '0';

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

    multaddalu12x12_inst: MULTADDALU12X12
        GENERIC MAP (
            A0REG_CLK => "CLK0",
            A0REG_CE => "CE0",
            A0REG_RESET => "RESET0",
            A1REG_CLK => "CLK0",
            A1REG_CE => "CE0",
            A1REG_RESET => "RESET0",
            B0REG_CLK => "CLK0",
            B0REG_CE => "CE0",
```

```
B0REG_RESET => "RESET0",
B1REG_CLK => "CLK0",
B1REG_CE => "CE0",
B1REG_RESET => "RESET0",
ACCSEL_IREG_CLK => "BYPASS",
ACCSEL_IREG_CE => "CE0",
ACCSEL_IREG_RESET => "RESET0",
CASISEL_IREG_CLK => "BYPASS",
CASISEL_IREG_CE => "CE0",
CASISEL_IREG_RESET => "RESET0",
ADDSUB0_IREG_CLK => "BYPASS",
ADDSUB0_IREG_CE => "CE0",
ADDSUB0_IREG_RESET => "RESET0",
ADDSUB1_IREG_CLK => "BYPASS",
ADDSUB1_IREG_CE => "CE0",
ADDSUB1_IREG_RESET => "RESET0",
PREG0_CLK => "BYPASS",
PREG0_CE => "CE0",
PREG0_RESET => "RESET0",
PREG1_CLK => "BYPASS",
PREG1_CE => "CE0",
PREG1_RESET => "RESET0",
FB_PREG_EN => "FALSE",
ACCSEL_PREG_CLK => "BYPASS",
ACCSEL_PREG_CE => "CE0",
ACCSEL_PREG_RESET => "RESET0",
CASISEL_PREG_CLK => "BYPASS",
CASISEL_PREG_CE => "CE0",
CASISEL_PREG_RESET => "RESET0",
ADDSUB0_PREG_CLK => "BYPASS",
ADDSUB0_PREG_CE => "CE0",
```

```
        ADDSUB0_PREG_RESET => "RESET0",
        ADDSUB1_PREG_CLK => "BYPASS",
        ADDSUB1_PREG_CE => "CE0",
        ADDSUB1_PREG_RESET => "RESET0",
        OREG_CLK => "CLK0",
        OREG_CE => "CE0",
        OREG_RESET => "RESET0",
        MULT_RESET_MODE => "SYNC"
        PRE_LOAD => X"000000000000",
        DYN_ADD_SUB_0 => "FALSE",
        ADD_SUB_0 => '0',
        DYN_ADD_SUB_1 => "FALSE",
        ADD_SUB_1 => '0',
        DYN_CASI_SEL => "FALSE",
        CASI_SEL => '1',
        DYN_ACC_SEL => "FALSE",
        ACC_SEL => '0',
    )
    PORT MAP (
        DOUT => dout,
        CASO => caso,
        A0 => a0,
        B0 => b0,
        A1 => a1,
        B1 => b1,
        CASI => casi,
        ACCSEL => gw_gnd,
        CASISEL => gw_gnd,
        ADDSUB => ADDSUB_i,
        CLK => CLK_i,
        CE => CE_i,
```

```
        RESET => RESET_i
    );
```

5 IP 调用

IP Core Generator 中 DSP 模块支持三种高云原语的产生：MULT、MULTALU、MULTADDALU。

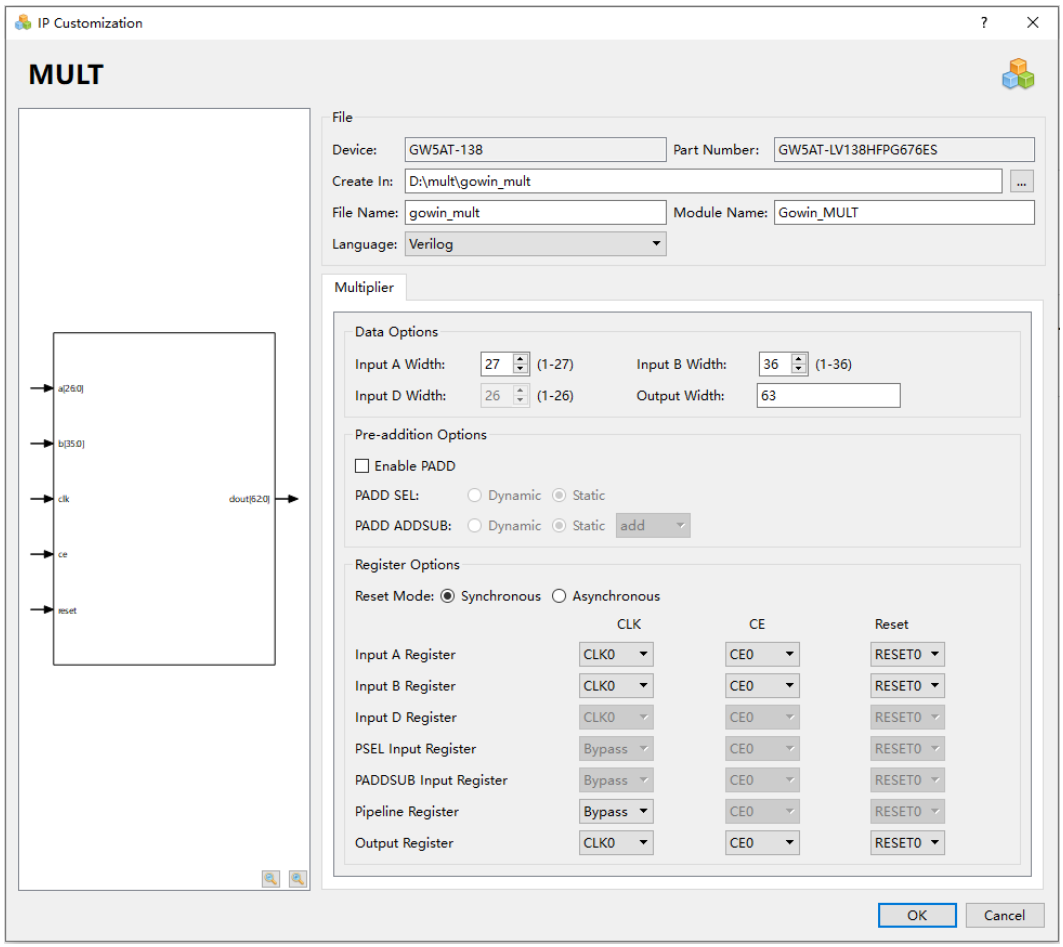
5.1 MULT

MULT 实现乘法运算、基于乘法的预加减功能。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 5-1 所示。该窗口包括“File”配置框、“Multiplier”配置框和端口显示框图。

图 5-1 MULT 的 IP Customization 窗口结构



1. 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. Multiplier 配置框：用于用户自定义配置 IP，Multiplier 配置框如图 5-1 所示。

- Data Options：配置数据选项。
  - 输入端口（Input A Width）最大数据位宽为 27；
  - 输入端口（Input B Width）最大数据位宽为 36；
  - 输入端口（Input D Width）最大数据位宽为 26；
  - 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 MULT12X12，MULT27X36。
- Pre-addition Options：配置预加选项。
  - “Enable PADD”配置 PADD；
  - “PADD SEL”选项配置 PADD 使能为动态控制或者静态控制；
  - “PADD ADDSUB”选项配置 PADD 预加减为动态控制或静态控制；
  - 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 MULT12X12，MULT27X36。
- Register Options：配置寄存器工作模式。
  - “Reset Mode”选项配置 MULT 的复位模式，支持同步模式“Synchronous”和异步模式“Asynchronous”；
  - “Input A Register”选项配置 Input A Register 的 clk、ce、reset 信号源。CLK 选项可配置为 Bypass、CLK0、CLK1，CE 选项可配置为 CE0、CE1，RESET 选项可配置为 RESET0、RESET1；
  - “Input B Register”选项配置 Input B Register 的 clk、ce、reset 信号源。配置选项同上；
  - “Input D Register”选项配置 Input D Register 的 clk、ce、reset 信号源。配置选项同上：
  - “PSEL Input Register”选项配置 PSEL Input Register 的 clk、ce、reset 信号源。配置选项同上；
  - “PADDSUB Input Register”选项配置 PADDSUB Input Register 的 clk、ce、reset 信号源。配置选项同上；
  - “Pipeline Register”选项配置 Pipeline Register 的 clk、ce、reset 信号源。配置选项同上；
  - “Output Register”选项配置 Output Register 的 clk、ce、reset 信号源。配置选项同上；

3. 端口显示框图：显示当前 IP Core 的配置结果示例框图，输入输出端口以及位宽根据 Options 配置实时更新，如图 5-1 所示。

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！

如配置中选择的语言是 VHDL，则产生的前两个文件名后缀为.vhd。

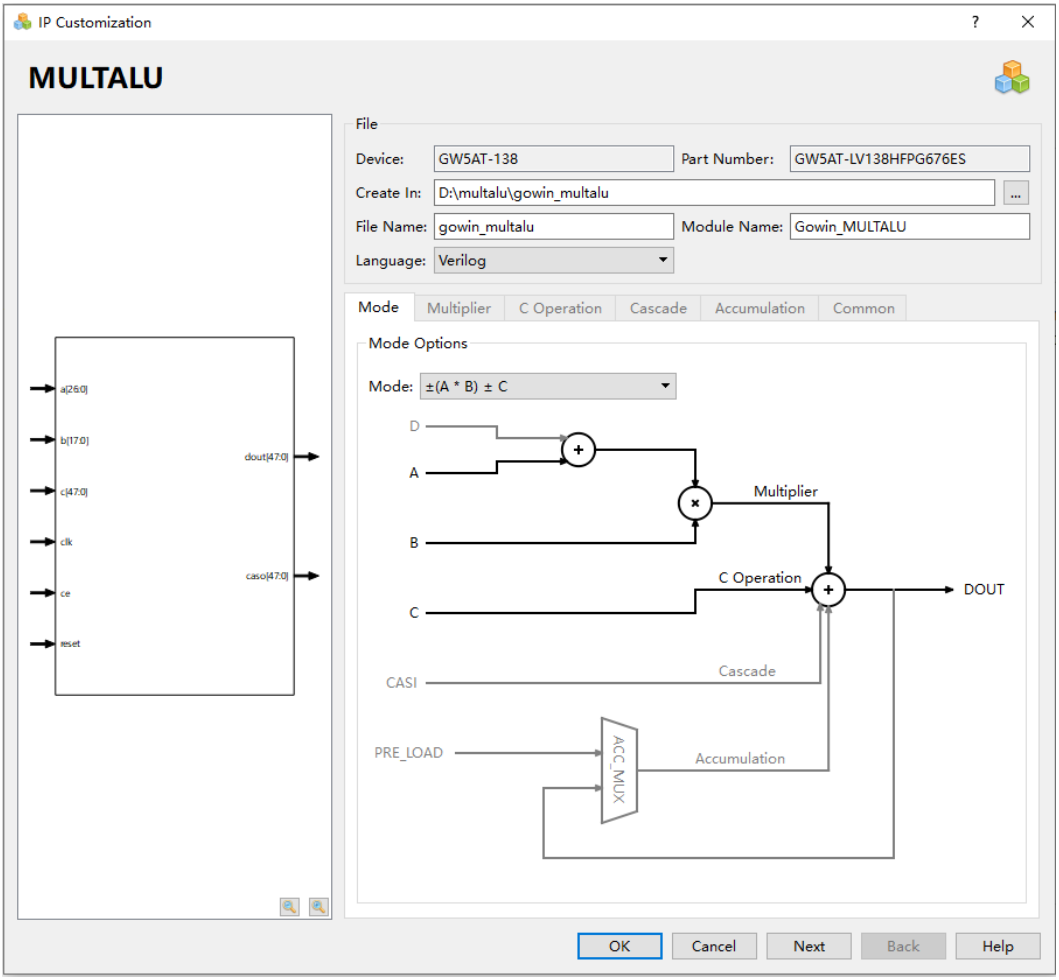
5.2 MULTALU

MULTALU 实现乘法、乘加、累加、乘累加、基于乘法/乘加/累加/乘累加的移位、基于乘法/乘加/累加/乘累加的级联、基于乘法/乘加/累加/乘累加的预加减等功能。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，“Mode”、“Multiplier”、“C Operation”、“Cascade”、“Accumulation”、“Common”配置框，如图 5-2 所示。

图 5-2 MULTALU 的 IP Customization 窗口结构



1. 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 MULT 模块的类似，请参考 5.1 MULT 中的 File 配置框。
2. Mode 配置框：配置 MULTALU27X18 的运算模式。
   - Mode Options：配置 MULTALU27X18 的运算模式。可选择：
     - ±(A*B)
     - ±((A±D) *B)
     - ±(A*B) ± C
     - ±((A±D) *B) ± C
     - Accum ±(A*B)
     - Accum ±((A±D) *B)
     - Accum ±(A*B) ± C
     - Accum ±((A±D) *B) ± C

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

3. Multiplier 配置框：配置乘法器，包括 Data Options,Pre-addition Options,ASEL Option,ADDSUB0 Option,Shift Option,Register Options。
   - MULTALU 的 Data Options,Pre-addition Options,Register Options 配置的使用和 MULT 模块的类似，请参考 5.1 MULT。
   - ASEL Option：配置 A,SIA 源选择的控制模式。
     - 支持动态控制“Dynamic”和静态控制“Static”。
     - 控制模式为动态时，ASEL 输入端口使能。
     - 控制模式为静态时，可配置为“Parallel”（选择 A）和“Shift”（选择 SIA）
   - ADDSUB0 Option：配置 M0/0 加/减选择的控制模式。
     - 支持动态控制“Dynamic”和静态控制“Static”。
     - 控制模式为动态时，ADDSUB0 输入端口使能。
     - 控制模式为静态时，可配置为“add”（选择加）和“sub”（选择减）
   - Shift Option：使能 shift out 功能。
4. C Opreation 配置框：配置输入 C，包括 Data Options,CSEL Option,ADDSUB1 Option,Register Options。
   - MULTALU 的 Data Options,Register Options 配置的使用和 MULT 模块的类似，请参考 5.1 MULT。
   - CSEL Option：配置 C,0 源选择的控制模式。
     - 支持动态控制“Dynamic”和静态控制“Static”。
   - ADDSUB1 Option：配置 M1,C,0 加/减选择的控制模式。
     - 支持动态控制“Dynamic”和静态控制“Static”。

- 控制模式为动态时，ADDSUB1 输入端口使能。
- 控制模式为静态时，可配置为“add”（选择加）和“sub”（选择减）

5. Cascade 配置框：配置级联输入 CASI，包括 CASISEL Option,Register Options。
   - MULTALU 的 Register Options 配置的使用和 MULT 模块的类似，请参考 5.1 MULT。
   - CASISEL Option：配置 CASI,0 源选择的控制模式。
     - 支持动态控制“Dynamic”和静态控制“Static”。
     - 控制模式为动态时，CASISEL 输入端口使能。
6. Accumulation 配置框：配置 ACCSEL 和 PRE_LOAD，包括 ACCSEL Option,Initialization Option,Register Options。
   - MULTALU 的 Register Options 配置的使用和 MULT 模块的类似，请参考 5.1 MULT。
   - ACCSEL Option：配置 PRE_LOAD,输出反馈源选择的控制模式。
     - 支持动态控制“Dynamic”和静态控制“Static”。
     - 控制模式为动态时，ACCSEL 输入端口使能。
     - 控制模式为静态时，可配置为“PRE_LOAD”（选择 PRE_LOAD）和 DOUT（选择输出反馈）。
   - Initialization Option：设定 PRE_LOAD 的初值。
     - Preload Value 的范围为 48’h000000000000~48’hFFFFFFFFFFFF。
7. Common 配置框：配置输出和复位模式，包括 Data Options,Register Options。
   - MULTALU 的 Data Options,Register Options 配置的使用和 MULT 模块的类似，请参考 5.1 MULT。
8. 端口显示框图：显示当前 IP Core 的配置结果示例框图，输入输出端口及其位宽根据 Options 配置实时更新，如图 5-2 所示。

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_multalu.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_multalu_tmp.v，为用户提供 IP 设计使用模板文件；
- IP 配置文件：“gowin_multalu.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。

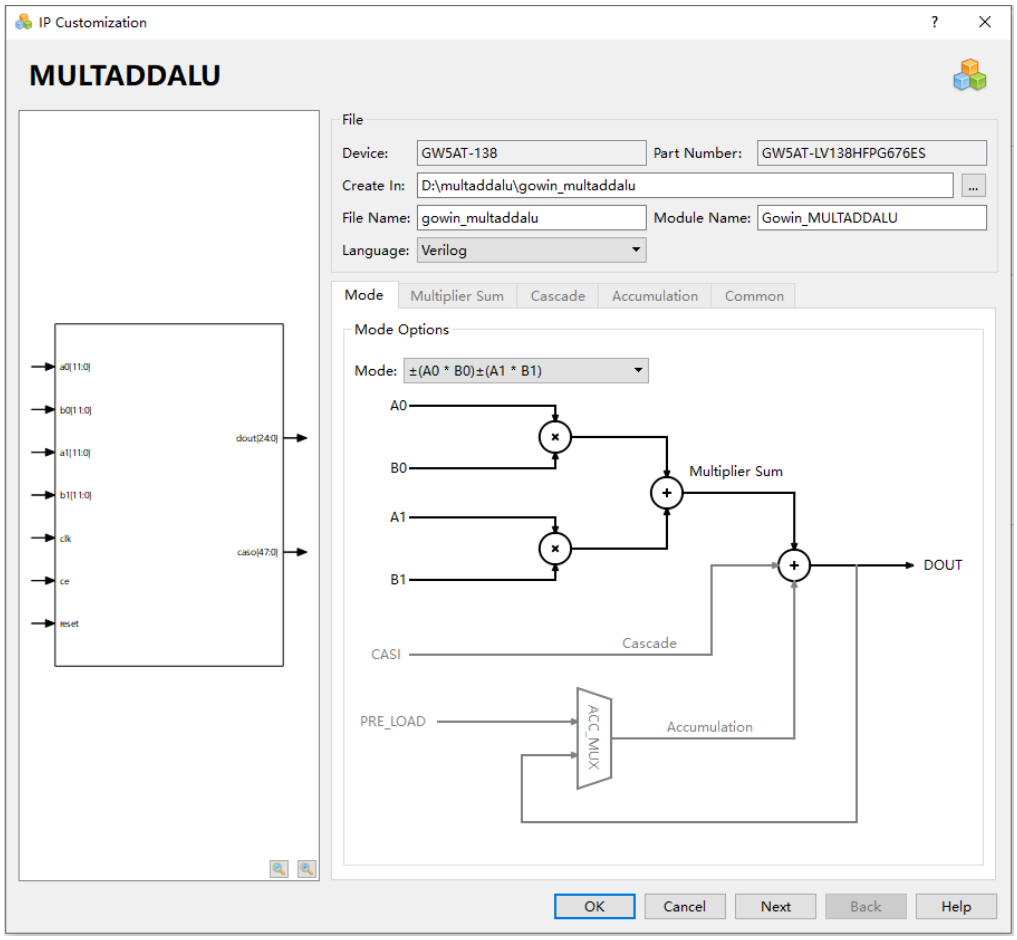
5.3 MULTADDALU

MULTADDALU 实现乘法器二次求和或累加功能。在 IP Core Generator 界面中，选择 Hard Module 中的 DSP 模块部分，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”配置框和端口显示框图，如图 5-3 所示。

图 5-3 MULTADDALU 的 IP Customization 窗口结构



1. 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 MULT 模块的类似，请参考 5.1 MULT 中的 File 配置框。
2. Mode Option：配置 MULTADDALU 的运算模式。可选择：
   - ± (A0 * B0) ± (A1 * B1)
   - CASI ± (A0 * B0) ± (A1 * B1)
   - Accum ± (A0 * B0) ± (A1 * B1)
   - Accum ± CASI ± (A0 * B0) ± (A1 * B1)
3. Multiplier Sum，Cascade, Accumulation, Common 配置框里的参数的配置与 MULTALU 里的配置类似，请参考 5.2 MULTALU 中的配置框。
4. 端口显示框图：显示当前 IP Core 的配置结果示例框图，输入输出端口及其位宽根据 Options 配置实时更新，如图 5-3 所示。

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。


GOWIN高云
智慧逻辑 定制未来