14 PDS 软件中乘加器 IP 配置和使用

工程源码	
相关视频课程	

章节导读

Multiply-AccumulatorIP 支持 a,b 两个输入端口，a,b 端口数据位宽在 2 位到 36位之间任定且 a,b 端口数据类型可分为：Signed(有符号)，Unsigned（无符号）两种。Multiply-Accumulator IP 为乘累加模式时，其等效算术表达式为：p=p+/-a*b

14.1 实验任务

本章我们将向大家介绍如何使用 PDS 软件生成 Multiply-Accumulator IP，通过 PGL22G 开发板上 FPGA 的片上算术逻辑单元资源以及 PDS 软件提供的 Multiply-Accumulator IP 资源进行乘累加运算，程序执行后，判断仿真计算结果与人工计算结果是否一致。

新建 PDS 工程，首先在菜单栏里选择“Tools”然后单击 “IP Compiler”选项，如图 14-1 所示。

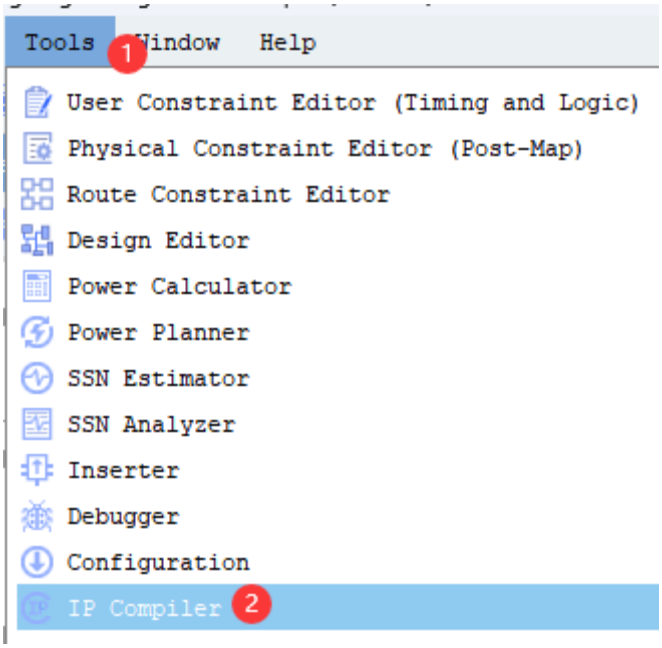


图 14-1 IP Compiler 页面

点击图 14-1 中的“IP Compiler”后会跳转图 14-2 页面。在图 14-2 中我们可以看到三个与“Multiplier”有关的 IP。标签 1“Multiply-Accumulator” 是一种乘累加运算，其算术表达式为：p=p+/-a*b；标签 2“Multiply-Adder”是一种乘加运算，其算术表达式为：p=a0*b0 +/- a1*b1 ；标签 3“Simple Multiplier”是一种简单的乘法运算，其算数表达式为：p=a*b。

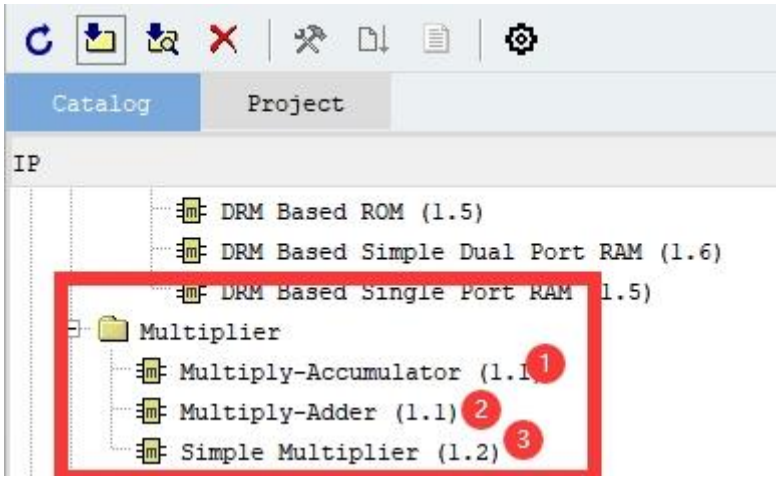


图 14-2 Multiplier IP 类型

由于本章实验章节主要讲解 Multiply-Accumulator IP 的使用方法，所以接下来我们点击图 14-3 页面中标签 1 中的“Multiply-Accumulator”按钮，选择创建 Multiply-Accumulator IP，然后将标签 2 中“Instance Name”选项命名为“acc”。最后点击标签 3 中的“Customize”按钮进入 Multiply-Accumulator IP 参数配置页面。

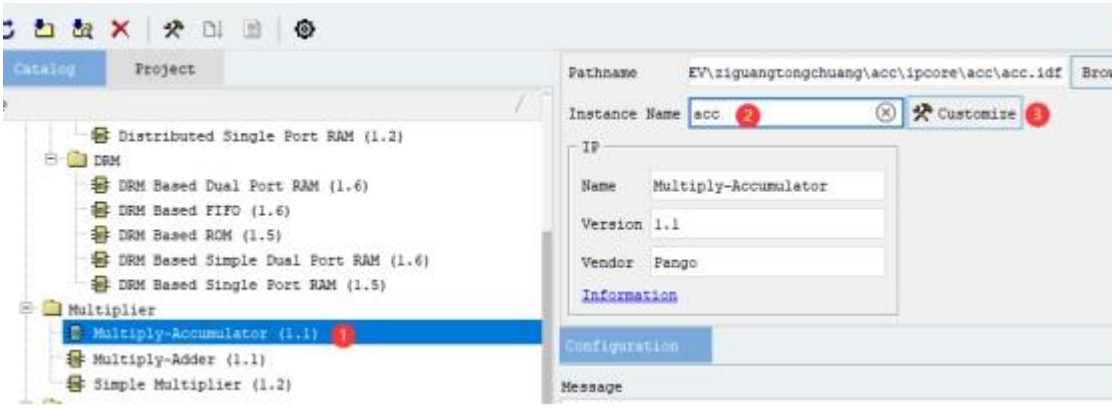


图 14-3 IP 选择页面

点击图 14-3 中的“Customize”按钮进入图 14-4 中。

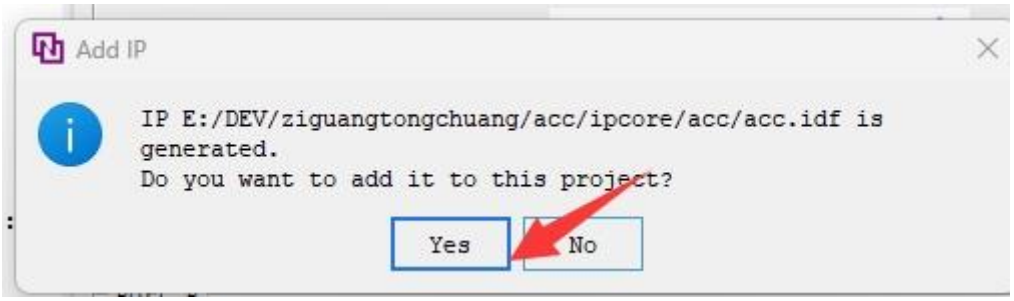


图 14-4 Add IP 弹窗

图 14-4 中表示的是 acc.idf 文件已经生成，询问我们是否将该文件加载到我们的工程中。因为这个 IP 我们后续是要在工程中用到的，因此这里直接点击图 14-4 中的“Yes”按钮即可。然后软件会自动进入 Multiply-Accumulator IP 的参数配置界面，如图 14-5 所示。图中展示了 Multiply-Accumulator IP 的若干个相关参数，包括数据位宽，数据类型，以下分别介绍：

Port A 方框中的“Data Type”(数据类型)有两种选择，一种是“Signed”(有符号类型)，实验中我们选择的是“Unsigned”(无符号类型)。“Width”（数据位宽），数据位宽的合法范围在 2~36 之间，这里我们将 Port A 中的数据位宽设置为 9，Port B 端口的设置与 Port A 设置相同。

Port P 方框中的“Width”（数据位宽）由 Port A，Port B 中的数据位宽共同决定：

如果 A，B 端口数据位宽都<=9 时，P 端口位宽可选为 24bit 或者 48bit；

如果 A，B 端口最大数据位宽<=18 时，P 端口位宽固定为 96bit；

如果 A，B 端口最大数据位宽<=36 且最小数据位宽<=18 时，P 端口位宽固定为 66bit；

如果 A，B 端口最大数据位宽<=36 时，P 端口位宽固定为 84bit

Pipeline Register Control 方框中的“Pipeline Stages ”表示为该 IP 输出寄存器，范围在 0~2 之间，为了在仿真窗口中直观清晰地观察信号，所以本实验将该选项设置为 0。

“Async Reset”（异步复位），勾选该选项是选择输出寄存器的复位方式为异步复位，未勾选时为同步复位。

Dynamic ACC ADD or ACC SUB 窗口中的“Dynamic ACC Addsub”选项表示的是“动态累加，累减”，不勾选该选项表示的是静态累加，累减。

“Subtract”表示的是静态累减运算，不勾选该选项表示的是静态累加运算，这里我们不做勾选，则该实验 IP 核的算术表达式为：p=p+a*b。

Reload Control 方框中的“Reload Dynamic Acc Init Value”表示的是动态输入初始值，不勾选该选项时表示的是静态配置初始值。

“Static Acc Init Value”表示的是配置静态初始值，当设置为静态配置初始值时有效。

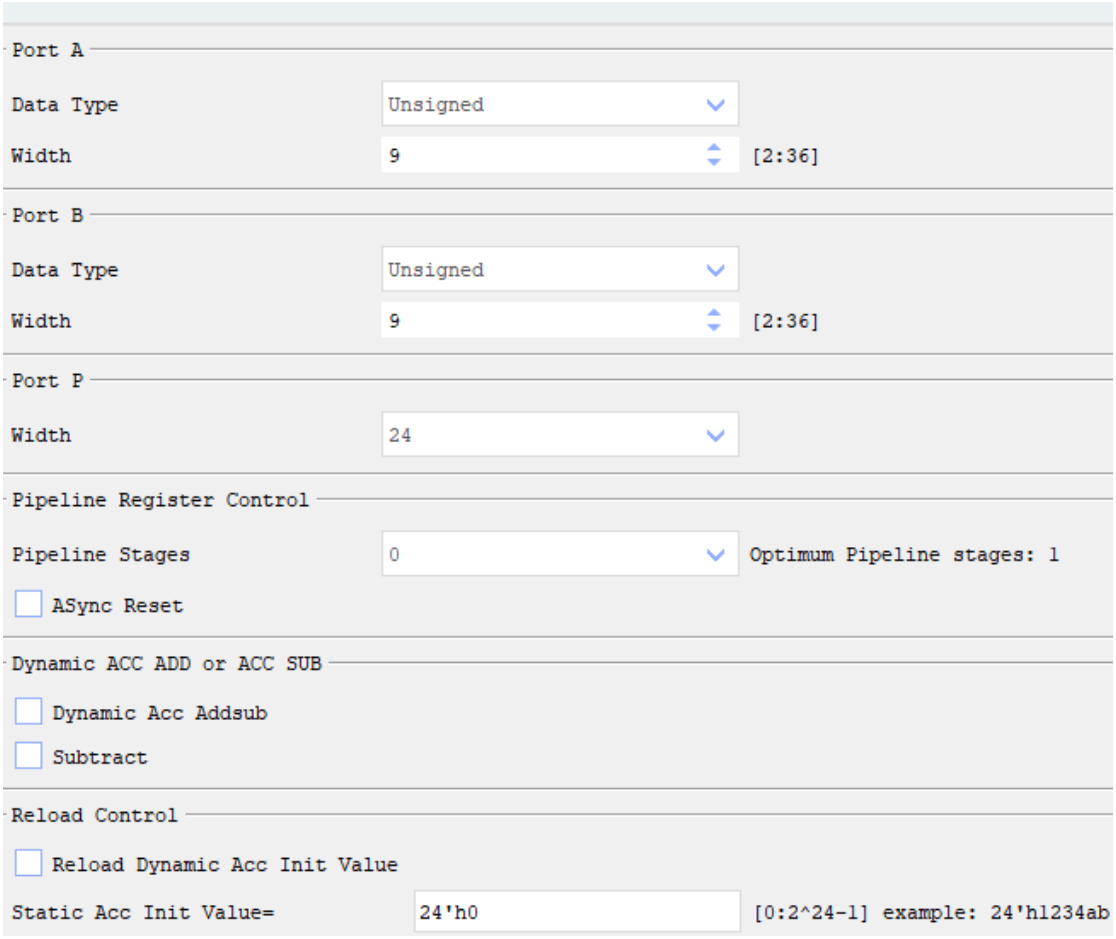


图 14-5 配置界面

至此本实验所需要的 Multiply-Accumulator IP 已介绍并配置完成，接下来点击图 14-6“Customize IP”窗口左上角箭头指向的“Generate”按钮即可。

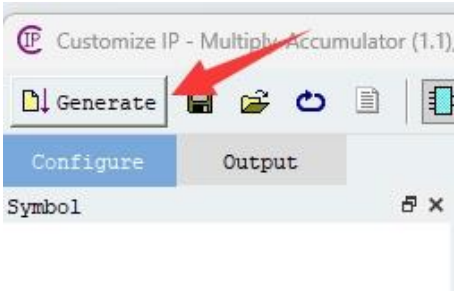


图 14-6 Customize IP 窗口

点击图 14-6 中的“Generate”按钮，进入图 14-7 界面。至此，Multiply-

Accumulator IP 核配置成功。

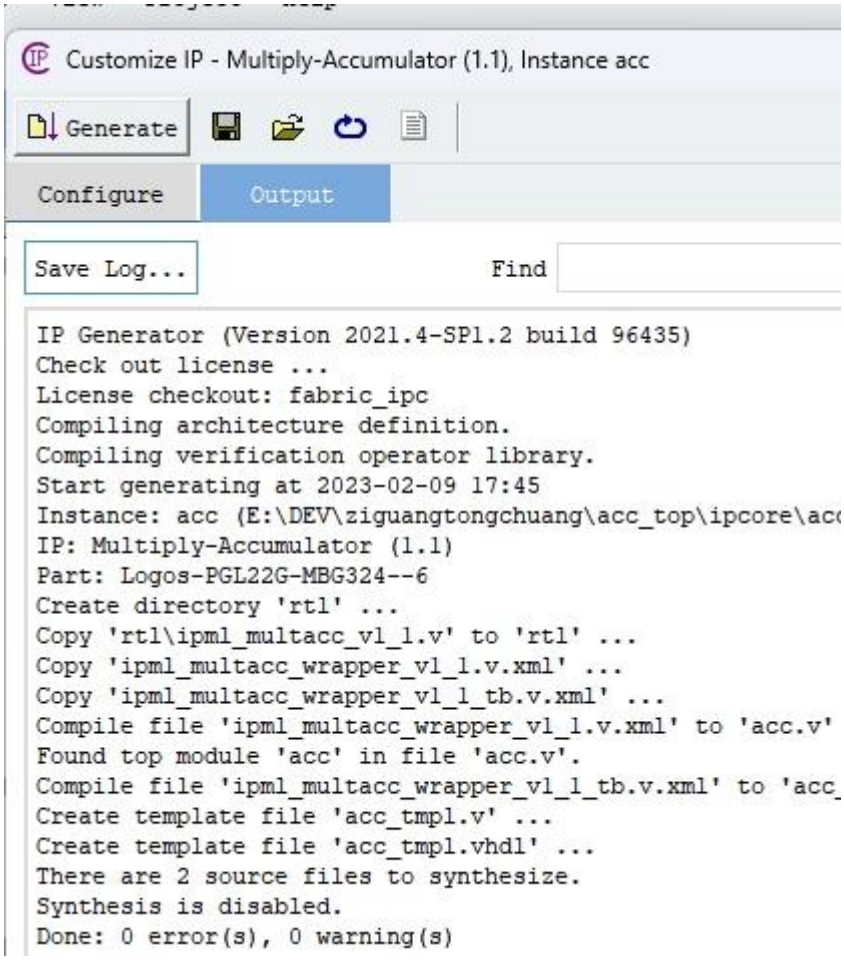


图 14-7 IP 配置成功

IP 核配置成功后会自动弹出图 14-8 IP 的例化模板文件，接下来我们需要例化这些自动生成的代码。

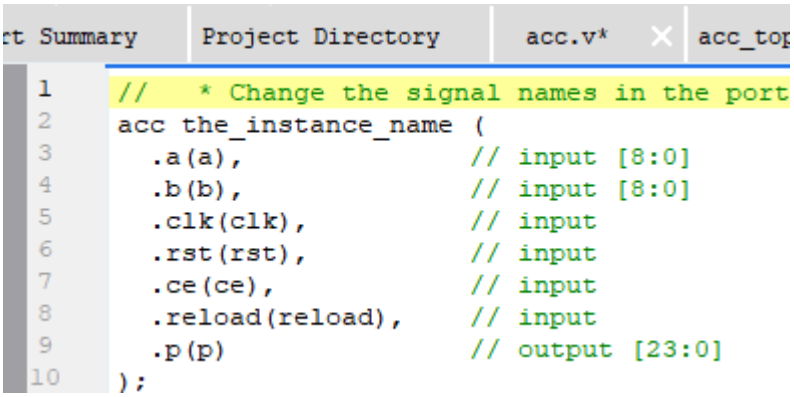


图 14-8IP 例化的模板文件

将 IP 核配置过程中弹出的页面关闭，返回 Source 面板，如图 14-9 所示。

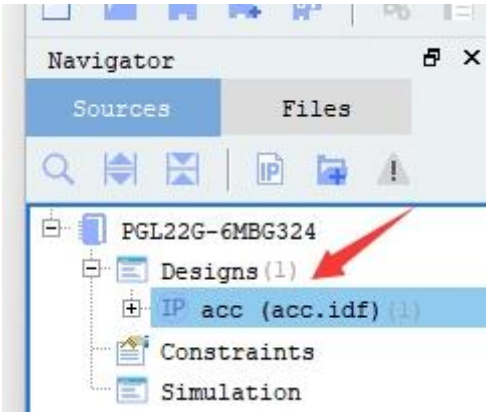


图 14-9Source 面板

点击图 14-9 页面中箭头指向的 acc.idf 文件左侧加号位置，可以看到 acc.idf 核下有两个文件，点击图 14-10 中箭头所指的位置，打开 acc.v 文件。

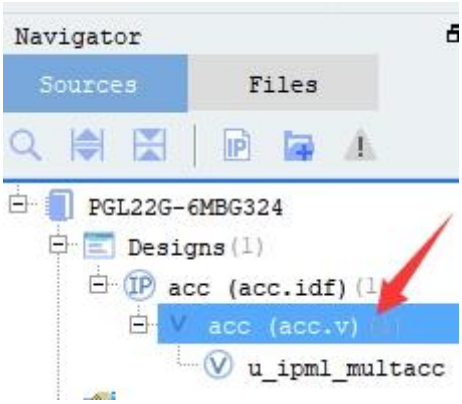


图 14-10 IP 核包含的文件

部分 acc.v 文件内容如图 14-11 所示，若要调用 Multiply Accumulator IP，我们可以选择将图 14-11 中的端口例化到需要调用 Multiply Accumulator IP 的模块中，或者是直接例化图 14-8 中内容。

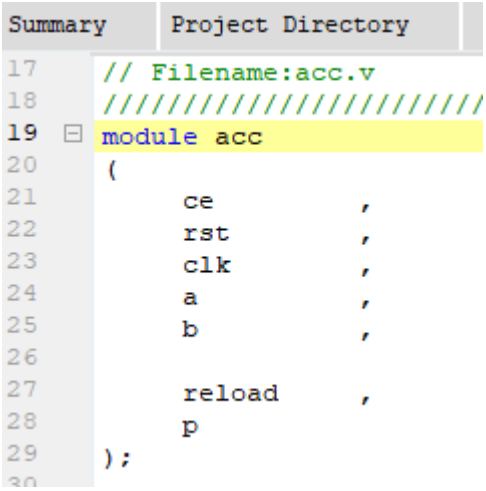


图 14-11 文件内容

14.1.1代码设计

接下来我们创建一个 verilog 源文件，其名称为 acc_top，本设计的端口列表和源码比较简单，代码的主要作用就是例化 Multiply-Accumulator IP，其中由 3-8 译码器决定输出端口 a 的值再将其通过连线赋值给 IP 的输入端口 a,输出端口 b 在系统时钟的上升沿到来后自加，为了方便我们计算这里当 b 自加到十后再归零，最后输出端口 b 的值再将其通过连线赋值给 IP 的输入端口 b,代码如下：

```
module acc_top(
   clk,
   reset_n,
   sw_0,
   sw_1,
   sw_2,
   p,
   a,
   b
   );

 input clk;
 input reset_n;
 input sw_0;
 input sw_1;
 input sw_2;
 output[23:0]p;
 output  [8:0]a;
 output  [8:0]b;
 reg[8:0] a;
 reg[8:0] b;

always@(sw_0,sw_1,sw_2)
   begin
      case({sw_0,sw_1,sw_2})
         3'b000:a = 8'b0000_0001;
         3'b001:a = 8'b0000_0010;
         3'b010:a = 8'b0000_0100;
         3'b011:a = 8'b0000_1000;
         3'b100:a = 8'b0001_0000;
         3'b101:a = 8'b0010_0000;
         3'b110:a = 8'b0100_0000;
         3'b111:a = 8'b1000_0000;
      endcase
   end
```

```
 always@(posedge clk or negedge reset_n)
    if(!reset_n)
      b<=0;
    else if(b==10)
      b<=0;
    else
      b<= b+1'b1;

acc acc  (
  .a(a),                      // input [8:0]
  .b(b),                      // input [8:0]
  .clk(clk),                  // input
  .rst(~reset_n),                  // input
  .ce(1),                    // input
  .reload(0),             // input
  .p(p)                       // output [23:0]
);

    endmodule
```

将上面的设计内容进行分析和综合直至没有错误以及警告后，按开发流程接下来进入激励创建及仿真测试环节。作为一个 FPGA 数字逻辑的完整的开发流程，仿真环节是必不可少的，这一要求请务必引起各位初学者的重视。

14.2激励创建及仿真测试

当 Multiply-Accumulator IP 创建好之后，我们可以通过仿真来对该 IP 进行测试，可以通过改变端口 a,b 的值来判断 Multiply-Accumulator IP 是否处于正常工作状态。这里针对该 IP，编写一个简单的 testbench 来进行仿真测试，testbench 代码如下所示。

```
`timescale 1ns/1ns
`define CLKA_PERIOD 20

module acc_top_tb();

reg grs_n;
 GTP_GRS GRS_INST(
 .GRS_N (grs_n)
 );

initial begin
 grs_n = 1'b0;
```

```
 #50000;
 grs_n = 1'b1;
 end

   reg clk;
   reg reset_n;
   reg sw_0;
   reg sw_1;
   reg sw_2;
   wire [23:0]p;
   wire[8:0]a;
   wire [8:0]b;


initial clk = 1'b1;
always #(`CLKA_PERIOD/2) clk = ~clk;

initial begin
   reset_n=0;
   #20;
   reset_n=1;
   #2000000;
  end

initial begin
      sw_0 = 0;sw_1 = 0;sw_2 = 0;
      #200;
      sw_0 = 0;sw_1 = 0;sw_2 = 1;
      #20000000;
      $stop;
   end

acc_top  acc_top(
  .clk(clk),
  .reset_n(reset_n),
  .sw_0(sw_0),
  .sw_1(sw_1),
  .sw_2(sw_2),
  .p(p),
  .a(a),
  .b(b)
   );
endmodule
```

testbench 代码中的`timescale 1ns/1ns 是为了控制仿真文件的置仿真时间单位以及仿真精度。格式为：timescale 单位/精度。例如： timescale 1ns/100ps 这样在

设置延时#100，就代表着延时 100*1ns。并且可以延时 100.1ns。如果想延时 10 0.0001ns 则可以将仿真精度修改为 1fs。仿真精度越高电脑所需要的仿真时间就越长，因此需要根据实际情况来进行定义。timescale 1ns/100ps，这种常用的书写方式就代表着仿真时间单位为 1ns，仿真精度为 1ns。

如何对模块仿真这里我们不再赘述，仿真波形如图 14-12 所示

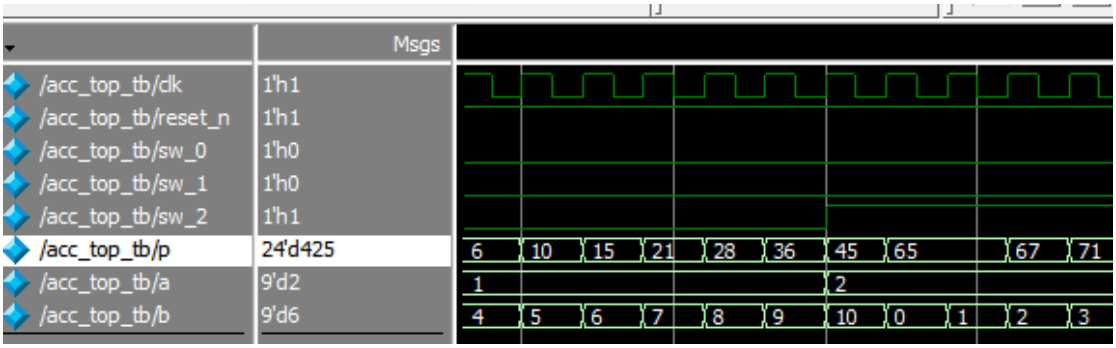


图 14-12 仿真波形

由图 14-12 可以看出当 sw_0,sw_1,sw_2 的值均为 0 时，a 端口的值为 1（与代码设计功能相符）， 当 b 端口的值为 4 时，p 端口在下一个时钟沿的值为 10，b 端口的值为 5 时，p 端口在下一个时钟沿的值为 15；当 sw_0,sw_1 的值均为 0，sw_2 的值为 1 时，a 端口的值为 2（与代码设计功能相符），， 当 b 端口的值为 10 时，p 端口在下一个时钟沿的值为 65， b 端口的值为 0 时，p 端口在下一个时钟沿的值仍为 65，依此类推，与 Multiply-Accumulator IP 的算术表达式 p=p+a*b 相符，通过仿真波形可以看到我们设置的 Multiply Accumulator IP 处于正常工作状态

14.3板级调试与验证

本实验的板级验证环节，主要验证以下几个目标：

1. 能否正确将生成的 bit 文件下载到 PGL22G 开发板；
2. Multiply Accumulator IP 输出数据是否与其算术表达式一致；

系统所需硬件：

1. PGL22G 开发板；
2. 电源电缆一根；
3. 硬件条件符合实验要求，具有完全开发功能的 PC 机一台；

14.3.1添加 I/O 约束

通常，一个设计中的 FPGA 不会是独立使用的，FPGA 一定会与其他外设、接口相连接，比如时钟，按键等。因此，FPGA 设计需要指定对应的 IO 引脚位置信息。

添加 IO 约束的方法非常简单：点击图 14-13 上方工具栏“Tools”栏中“User Constranit Editor(Timing and Logic)”后点击“Pre Synthesize USE”选项。

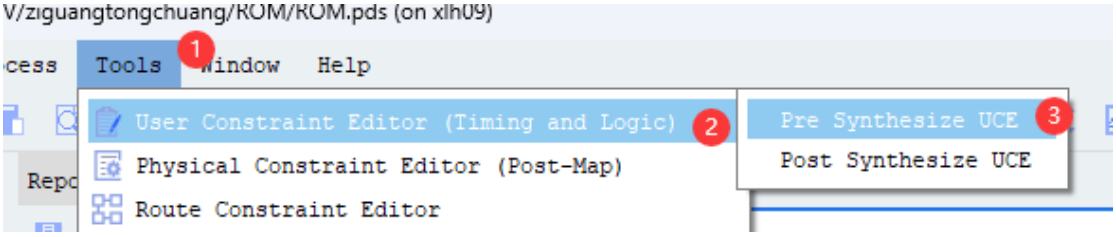


图 14-13“Tools”工具栏

点击图 14-13 中的“Pre Synthesize USE”选项后，弹出图 14-14 界面，首先点击“Pre Synthesize USE” 选项，然后点击“Device”选项，最后点击“I/O”选项，进入管脚分配页面。

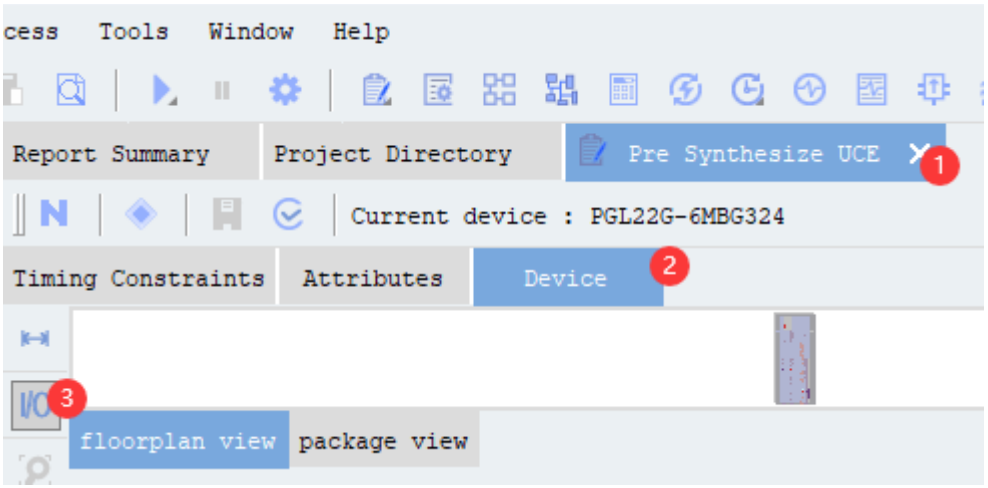
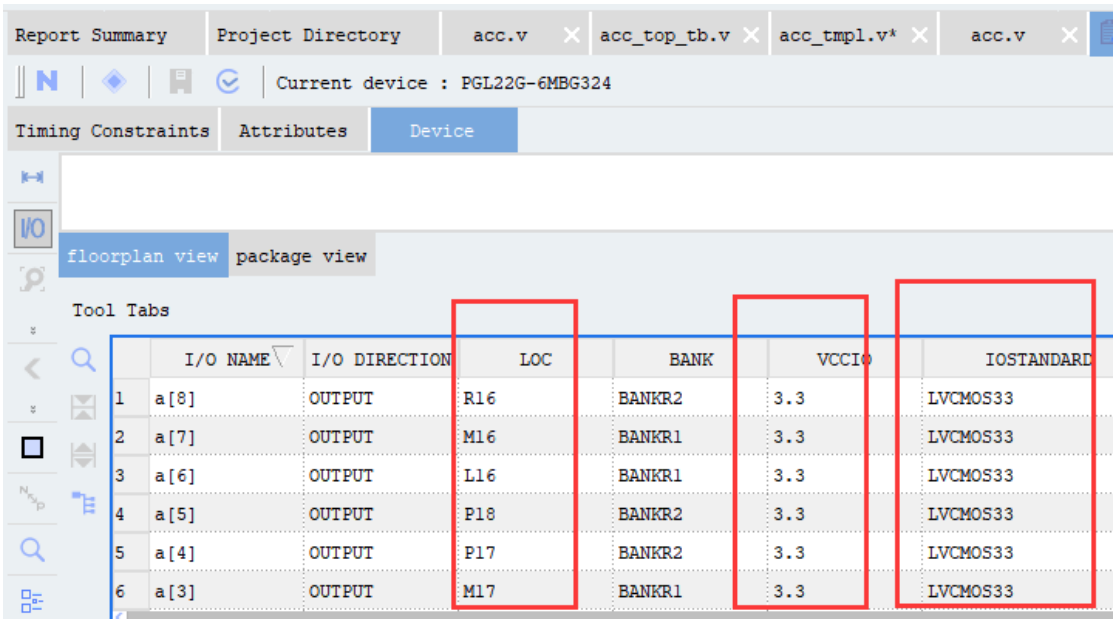


图 14-14 管脚分配入口

打开管脚约束窗口如图 14-15，其中 LOC ,VCCIO,IOSTANDARD 是我们要约束的内容。这里，参考我们提供的管脚约束表即可完成管脚绑定。



	I/O NAME	I/O DIRECTION	LOC	BANK	VCCIO	IOSTANDARD
1	a[8]	OUTPUT	R16	BANKR2	3.3	LVCMOS33
2	a[7]	OUTPUT	M16	BANKR1	3.3	LVCMOS33
3	a[6]	OUTPUT	L16	BANKR1	3.3	LVCMOS33
4	a[5]	OUTPUT	P18	BANKR2	3.3	LVCMOS33
5	a[4]	OUTPUT	P17	BANKR2	3.3	LVCMOS33
6	a[3]	OUTPUT	M17	BANKR1	3.3	LVCMOS33

图 14-15 管脚分配页面

这样管脚约束添加完成了。但是此时约束内容保存在内存中，还没有写入文件，点击工具栏保存按钮(图 14-16 中箭头指向的位置)，或者直接 Ctrl+S。

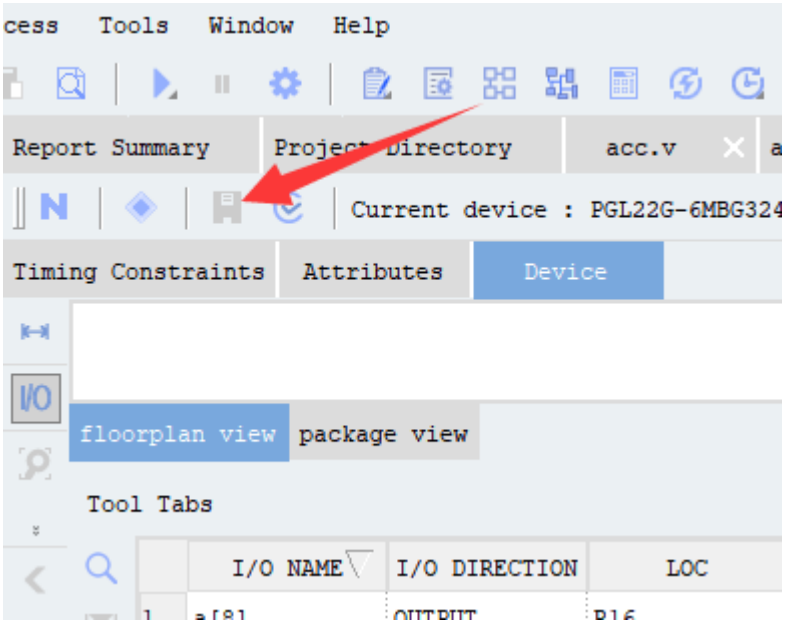


图 14-16 保存文件

将约束文件命名为 ACC_acc 后就可以在 Source 窗口的 Constraints 下可找到刚保存的 ACC_acc.fdc 文件。双击可以打开约束文件。

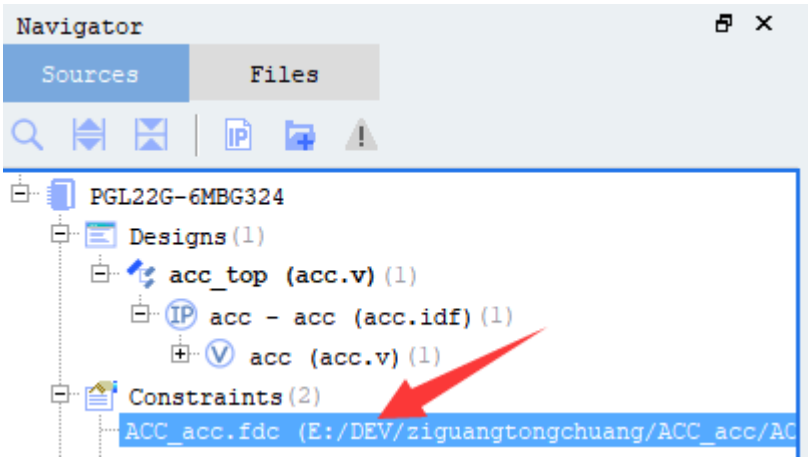


图 14-17 Source 面板

14.3.2下载验证

新建 DebugCore，将 a，b 以及 p 这些信号添加至观察列表中，新建 DebugCore 核的方法这里不再赘述。

编译工程并生成比特流.sbit 文件后，此时将下载器一端连接电脑，另一端与开发板上的 JTAG 下载口连接，连接电源线，并打开开发板的电源开关。硬件连接如图 14-18 所示。

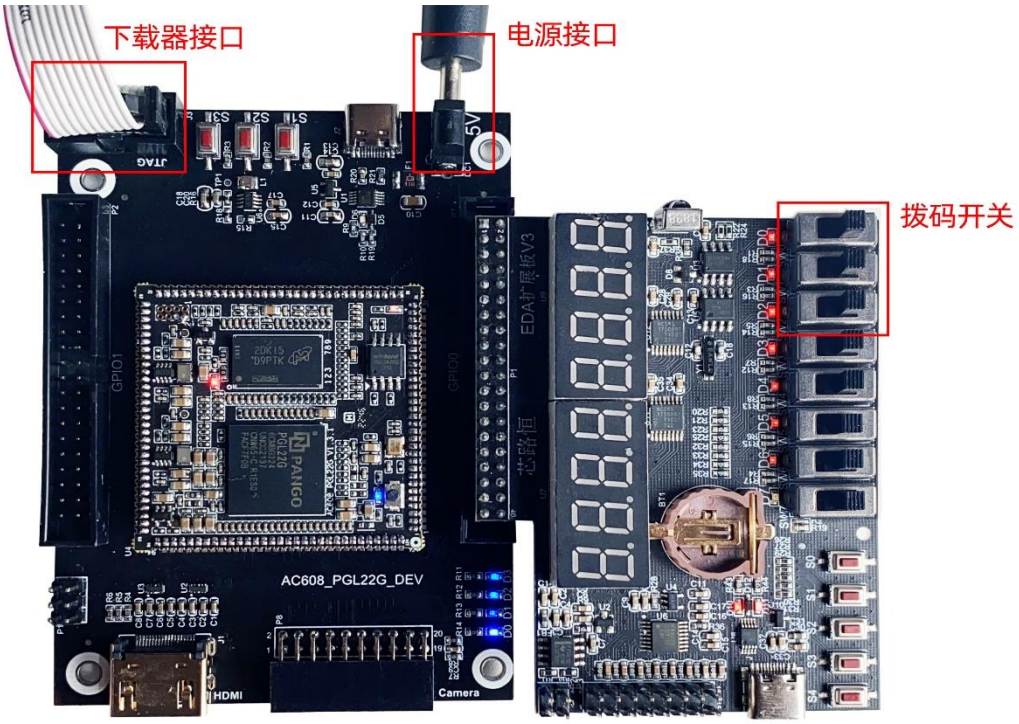


图 14-18 硬件连接

点击 PDS 工具栏的下载按钮，在弹出的 Fabric Configuration 界面中双击“Boundary Scan”，我们将生成好的 sbit 流文件下载到开发板中去, 然后点击图 14-19 中的触发按钮。

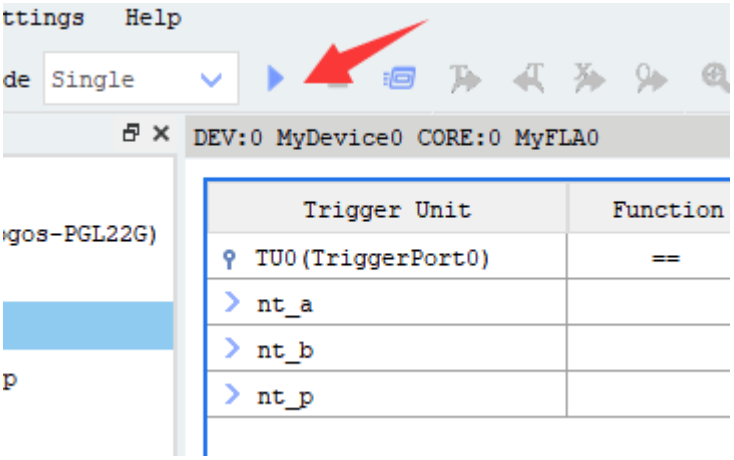


图 14-19 置触发信号

我们可以看出 当没有设置拨码开关 sw_0,sw_1,sw_2 时，sw_0,sw_1,sw_2 对应的值均为 0，此时 a 端口的对应的二进制应该为 8’0000_0001，观察波形。

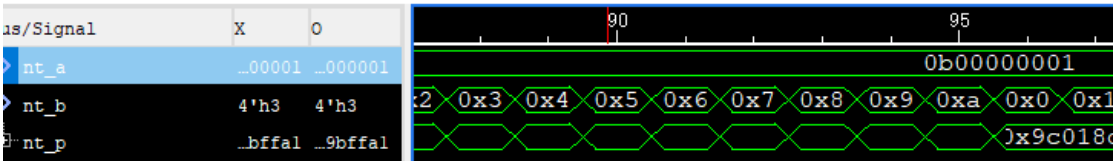


图 14-20 波形图

从图 14-20 中可以看出 a 端口对应的二进制值是 8’b0000_0001,与我们代码设计功能一致。此时，我们将拨码开关 sw_1,sw_2 上拨，sw_0 位置保持不变如图 14-21，再次点击图 14-19 中的触发按钮，抓取波形。

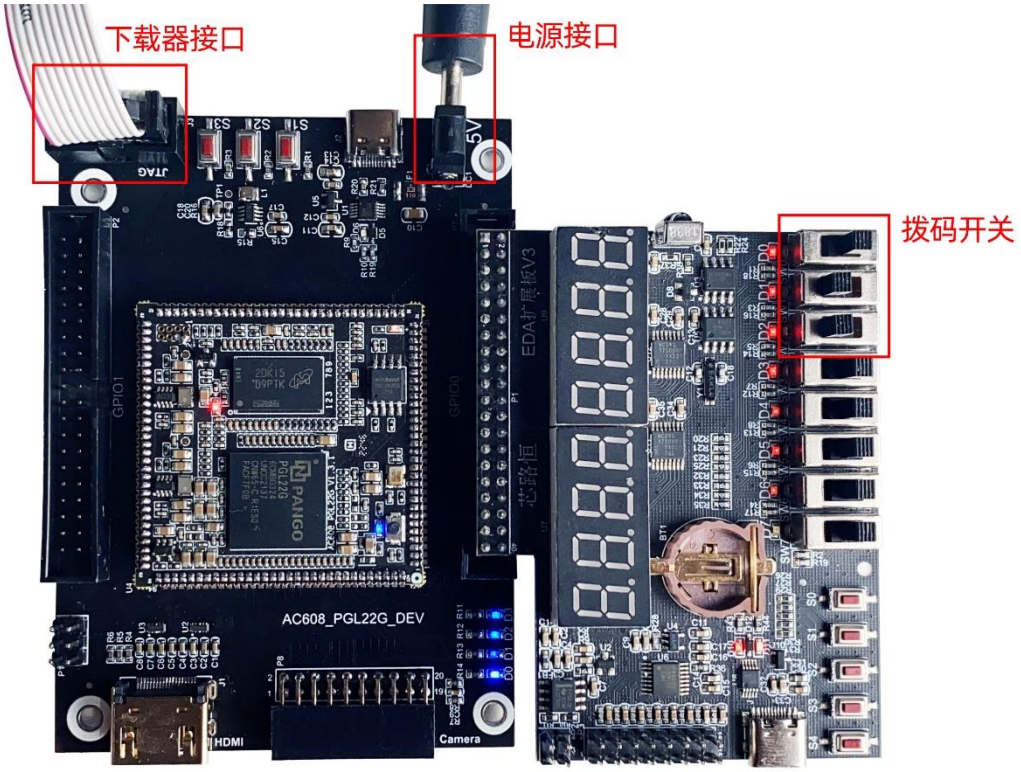


图 14-21sw_1,sw_2 上拨设置

拨码开关 sw_1,sw_2 上拨后的现象如图 14-22 所示，我们可以看出当拨开开关 sw_0 为 0，sw_1,sw_2 均为 1 时对应 a 端口的值十进制为 8，与代码设计功能一致。当 b 端口的值为 1 时，p 端口在下一个时钟沿的值为 16732784， b 端口的值为 2 时，p 端口在下一个时钟沿的值仍为 16732800，依此类推，与 Multiply-Accumulator IP 的算术表达式 p=p+a*b 相符。

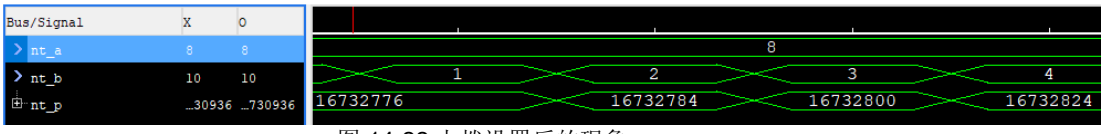


图 14-22 上拨设置后的现象

我们可以发现，上图中的数据变化同样和 PDS 仿真的波形时序是一致的，证明了本次实验的 IP 核处于正常工作状态。

14.4常见问题说明

1. 对于一个完整的 FPGA 工程开发流程，仿真是一个重要而必不可少的步骤；

2. 为了在仿真窗口中直观清晰地观察信号建议简单的功能实现不必对输出结果做延迟处理；

14.5总结

本章简单介绍了 PDS 软件 的 Multiply-Accumulator IP，通过调用 Multiply-Accumulator IP 的方式实现了乘累加运算。本章还简单复习了 3_8 译码器功能实现，进一步加深对组合逻辑的理解以及激励文件的编写能力。建议读者能够跟随本实验内容，完整的进行整个实验。