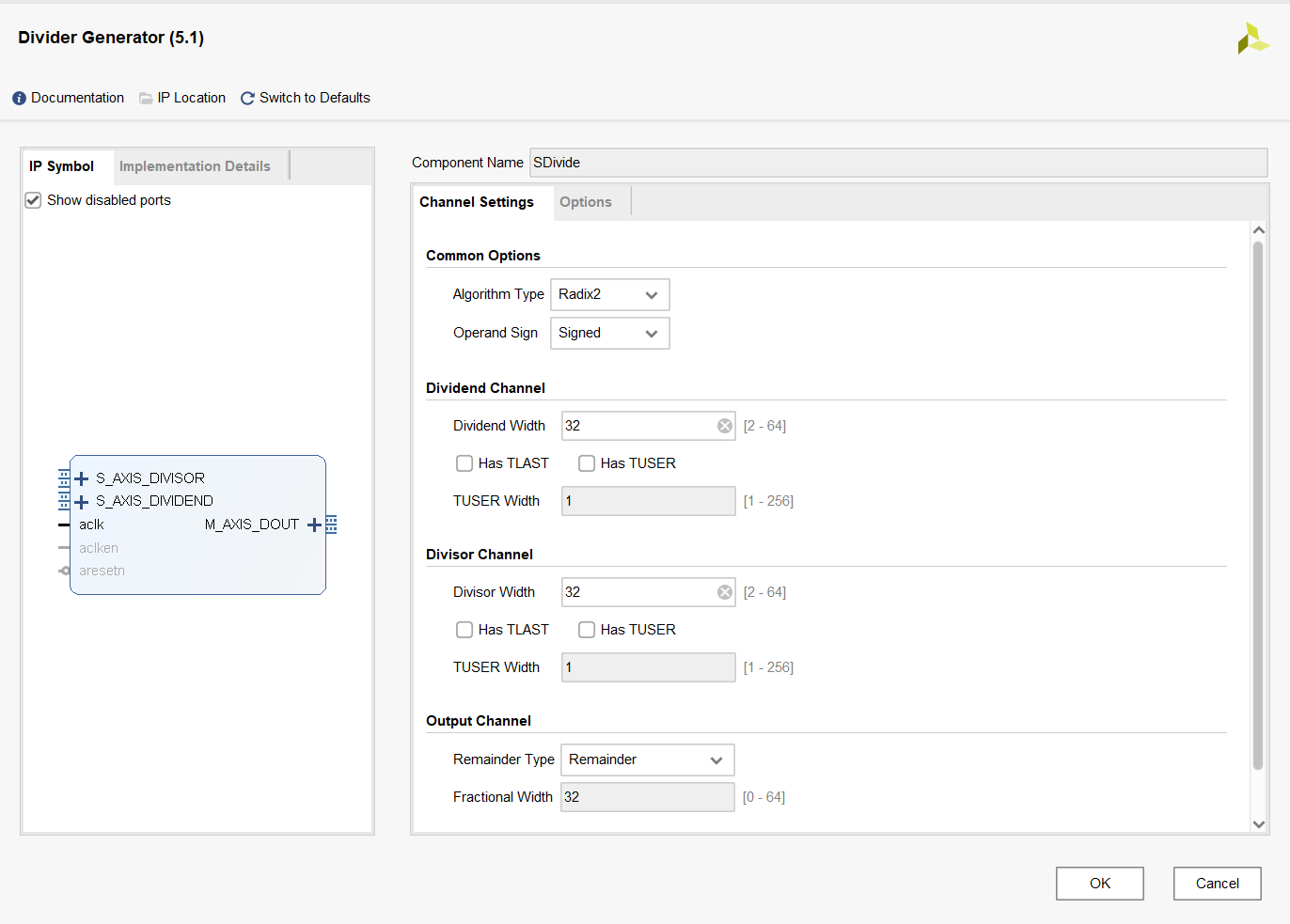
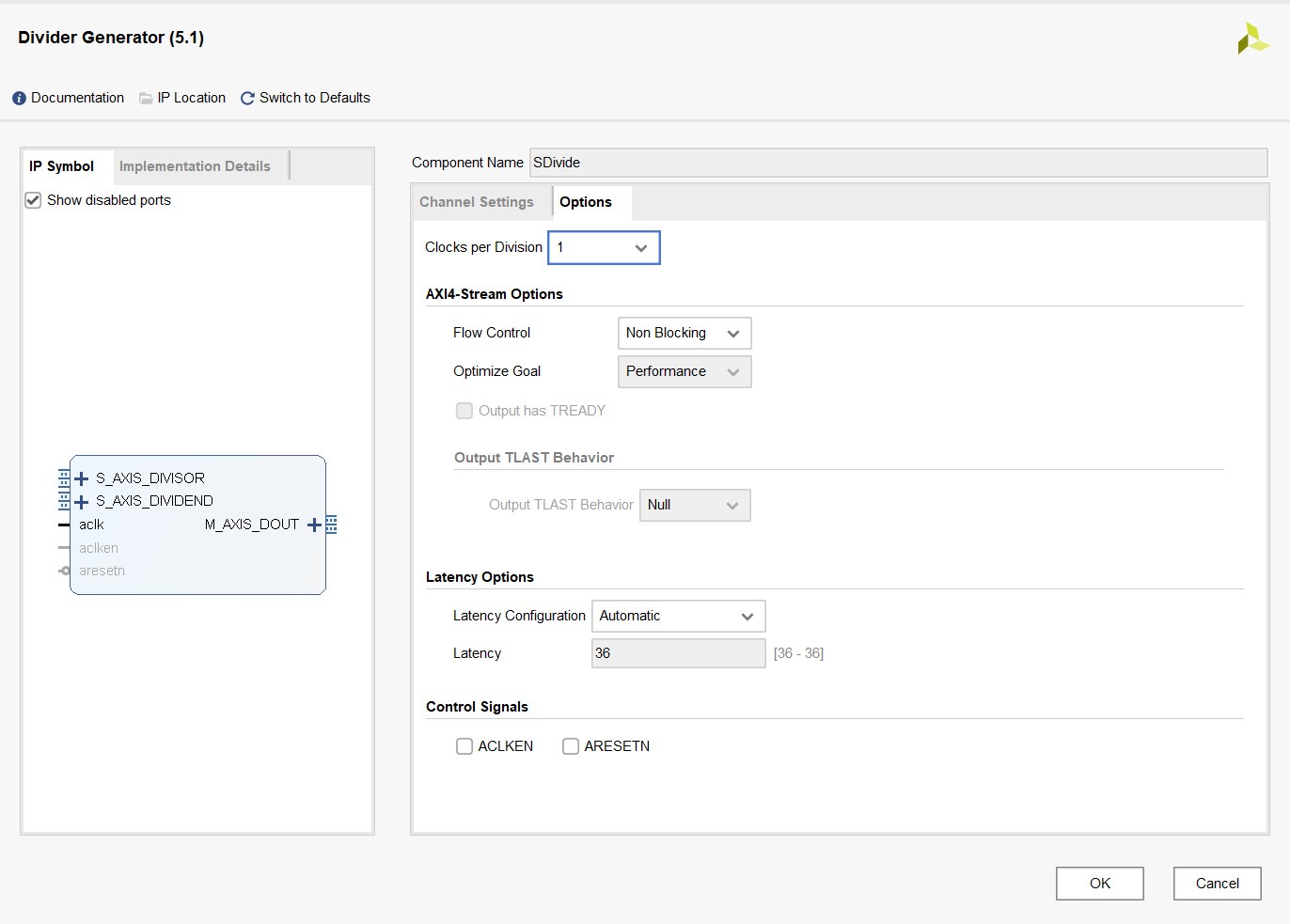
# 1. 除法ip测试

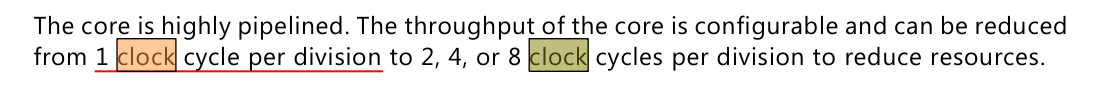
## 1 除法ip文档解读

* 文档下载：
* ip核配置选项：



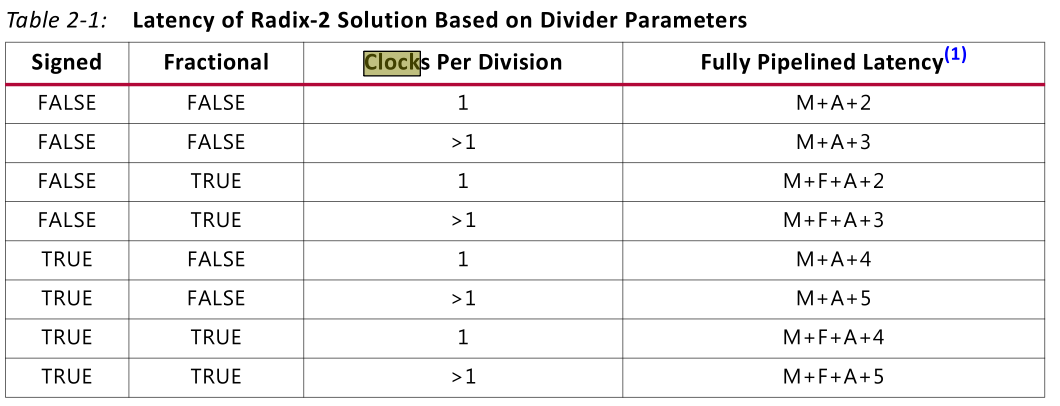


* ip核有三种算法模式：LUT，Radix2，以及高基数算法；由于LUT适合于2-17位，High Radix仅能计算分数，而Radix2能够计算2-64位的操作数，且输出提供余数或者分数模式。所以我们采用Radix2算法。
* Clock per Division：在Radix2算法下，其实是进行多次除法，所以提供了每次除法运算执行多少个周期。有兴趣的同学可以继续尝试。



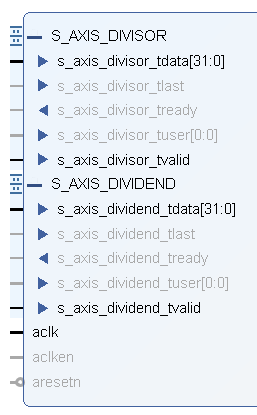
* Blocking（阻塞模式）是为AXI接口提供的选项；
* 关于Latency选项：

文档内截图：不过关于手动设置的选项，有兴趣的同学可以自己下去尝试。



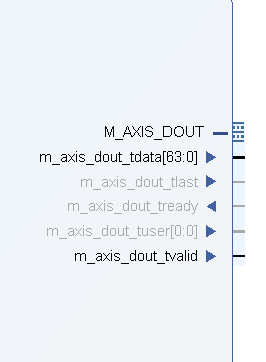
## 2 除法ip接口解释以及时序分析

* 输入接口：



1个时钟沿触发，两个数据接口，两个数据有效接口。

* 输出接口：



1个输出接口，1个输出有效接口。

* 时序分析：

假设对于无符号除法，流程：输入数据同时输入数据有效---除法计算33个周期计算出结果---第34个周期输出有效。

注意：必须控制输入有效的时间，若输入有效持续了两个周期，则输出有效也将持续两个周期，若在第二个周期内输入数据变化，则在输出有效的第二个周期内数据将发生对应的变化。正确的时序控制如下图所示。

图片包含 监视器, 屏幕截图, 室内, 墙壁

描述已自动生成

## 3 除法ip替换除法模块：

* 源码文件：
* 生成1个32位的有符号除法ip与1个32位的无符号除法ip，或者生成一个33位的有符号除法ip，将32位无符号数或者有符号数有符号扩展后带入计算。现采取的是第一种方案；
* 除法计算状态机

定义数据有效信号为：operand\_valid；除法暂停信号：div\_stall；输出有效信号：div\_out;

我们需要关注operand\_valid以及div\_stall，且必须使operand\_valid保持为一个周期的置为1的状态，否则输出可能出错；因为我们需要控制div\_stall信号，而div\_stall可以 由div\_out控制，即当openrand\_valid == 1时，div\_stall = 1; 当div\_out == 1时，div\_stall = 0；但是我们发现，div\_out是由operand\_valid决定的，（ps：其实这儿可以考虑operand\_valid有效两个周期，但是后续仍旧存在问题）所以我们需要控制operand\_valid信号。即通过控制oparend\_valid来控制div\_out，继而控制div\_stall信号；

关键代码：

reg s\_operand\_valid, uns\_operand\_valid; *// 有符号除法与无符号除法输入数据是否有效*

    reg flag;                               *// 辅助判断除法开始还是结束；*

*// 否则，DIV指令后接B族指令将会发生错误；*

*// DIV指令Memory阶段，仍会产生一次stall信号，因为正边沿触发，所以*

*// Memory阶段识别的alu\_control == `ALU\_DIV*

*// 同时此时div\_out == 0, 所以产生一次stall；*

*// 在下一周期后，alu\_control 才被正确识别，此时复位所有信号*

    wire div\_out;                           *// 除法输出有效*

    wire sign\_dout\_valid, unsign\_dout\_valid;*// 有符号输出有效 与 无符号输出有效*

    assign div\_out = sign\_dout\_valid || unsign\_dout\_valid; *//*

    wire [63:0] y\_s;

    wire [63:0] y\_uns;

    always@(posedge clk) begin

*// 同步复位；将所有信号量值*

        if(rst) begin

            stall = 0;

            s\_operand\_valid = 0;

            uns\_operand\_valid = 0;

            flag = 0;

        end

*// 判断当前指令是否为`ALU\_SIGNED\_DIV*

*// 会在执行阶段牺牲一个周期判断，在访存阶段牺牲一个周期结束状态机*

*// 待优化。*

        else if(alu\_control == `ALU\_SIGNED\_DIV && flag == 0) begin

*// 若当前没有除法暂停信号且除法没有输出.*

*// 则表示可以进行除法，输出除法暂停，并使得输入数据有效。*

            if(stall == 0 && div\_out == 0) begin

                stall = 1;

                s\_operand\_valid = 1;

            end

*// 控制数据输入有效持续一个周期，持续多个周期将会造成浪费；*

            else if(s\_operand\_valid == 1) begin

                s\_operand\_valid = 0;

            end

*// 第36个周期，除法输出有效，将flag 置为 1：表示除法已计算一次；同时结束暂停信号，*

*// stall置0，数据输入有效置0；*

*// 其实第35个周期已经计算出结果了，可惜只能通过div\_out判断除法是否运算结束。*

            else if(div\_out == 1 ) begin

                flag = 1;

                stall = 0;

                s\_operand\_valid = 0;

            end

*//其他情况：除法运算阶段，保持不变即可*

            else begin

                stall = 1;

                s\_operand\_valid = 0;

            end

        end

*// 以下代码为无符号除法状态机，由于计算周期不同，最开始分开书写状态机，可以合并。*

        else if(alu\_control == `ALU\_UNSIGNED\_DIV && flag == 0) begin

            if(stall == 0 && div\_out == 0) begin

                stall = 1;

                uns\_operand\_valid = 1;

            end

*// 确定除法开始周期*

            else if(uns\_operand\_valid == 1) begin

                uns\_operand\_valid = 0;

            end

*// 除法结束阶段*

            else if(div\_out == 1 ) begin

                flag = 1;

                stall = 0;

                uns\_operand\_valid = 0;

            end

*//其他情况：除法运算阶段*

            else begin

                stall = 1;

                uns\_operand\_valid = 0;

            end

        end

        else begin

            stall = 0;

            flag = 0;

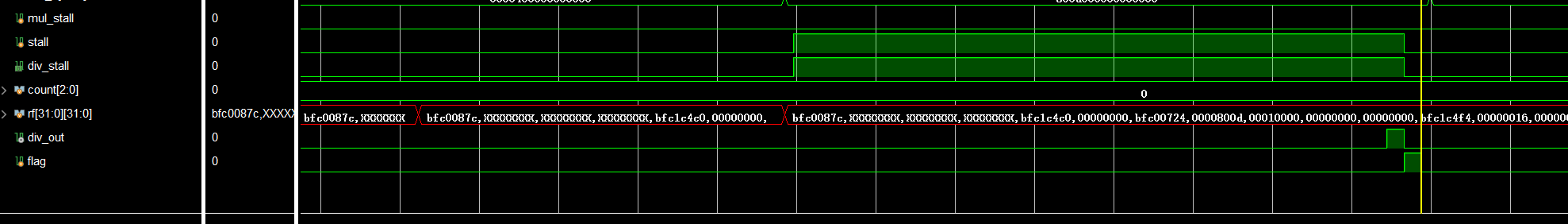
            s\_operand\_valid = 0;

            uns\_operand\_valid = 0;

        end

    end

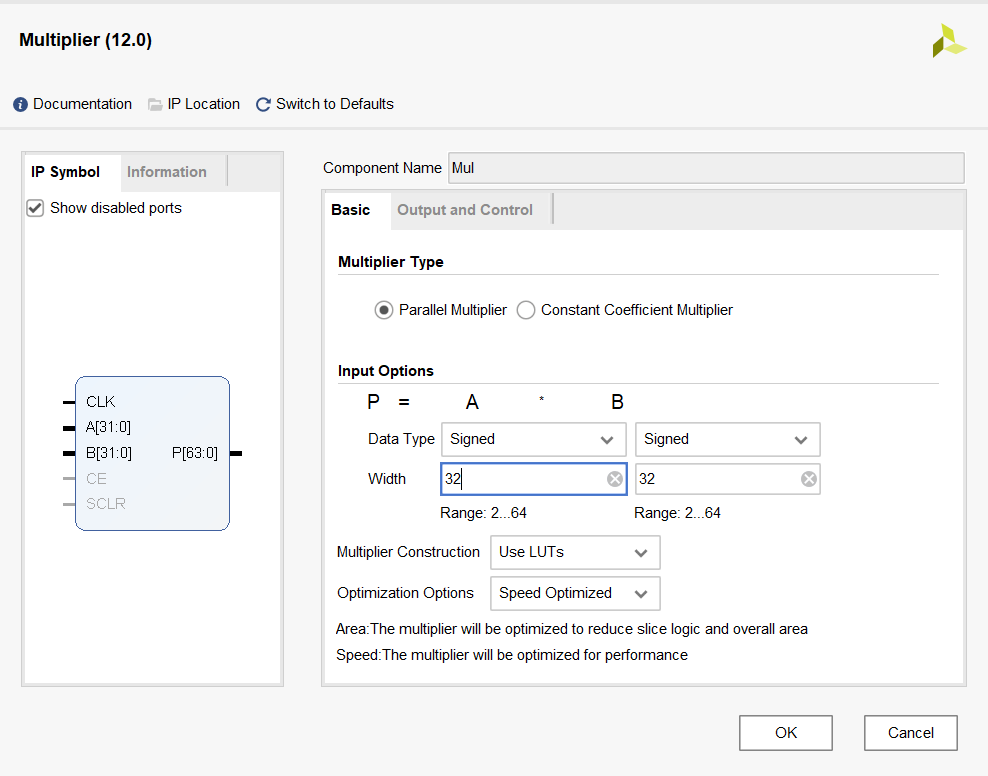
时序示意图：



# 2. 乘法ip测试

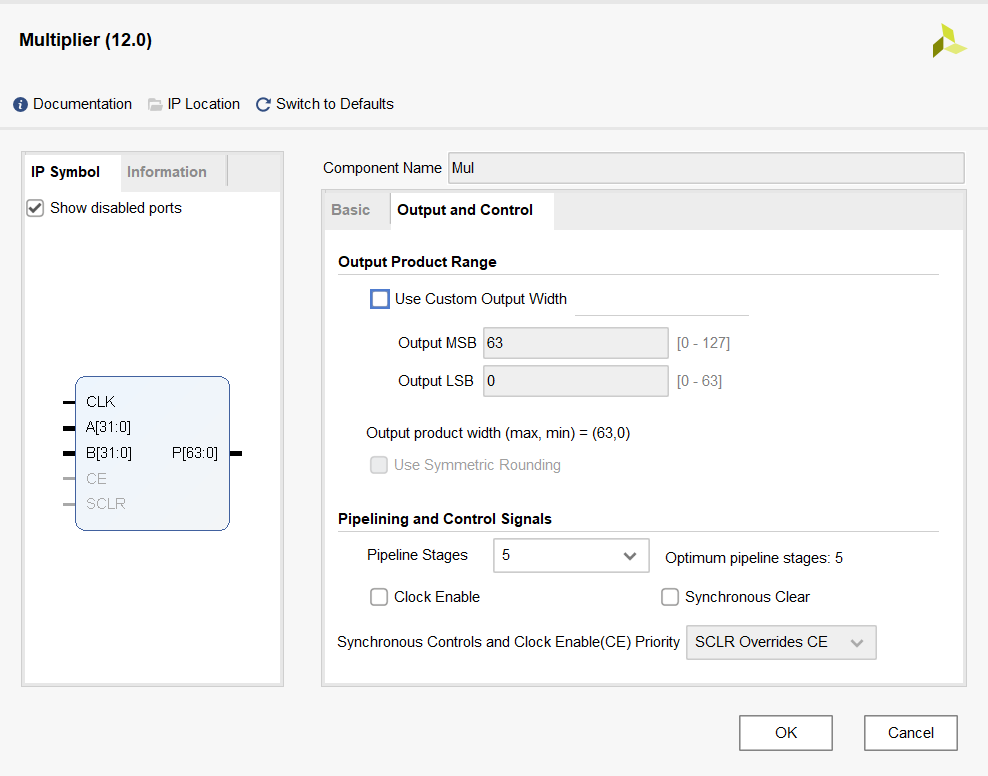
## 1 乘法ip文档解读

* 文档下载：
* ip核配置选项：



Constant Coefficient Multiplier：表示用一个常数与操作数相乘；

LUTS与MULTS实现乘法器的区别暂不详（后续可以探索）；

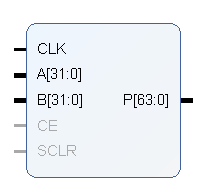


流水级数：若为0，则为组合逻辑，即表示当个周期内返回数据；

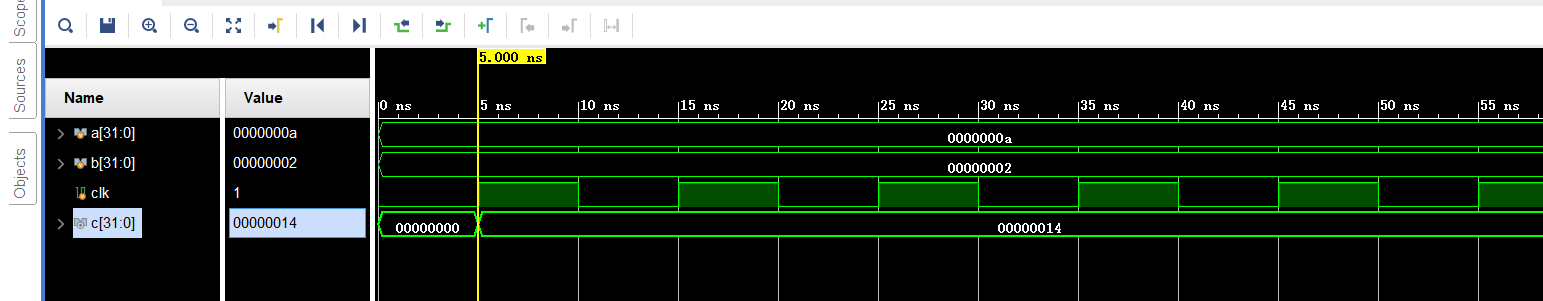
若为1，则将输出保存到寄存器中，下一个周期内输出数据，此处根据提示选择5级流水。

## 2 乘法ip接口以及时序分析：

* 接口：



* 时序分析：以一级流水为例



## 3 乘法ip替换乘法模块

1. 源码文件：
2. 状态机设计：

乘法ip的输入接口为1个时钟输入，两个数据输入，以及一个数据输出；因为我们采取的是流水线乘法ip，所以需要额外控制乘法运算状态；因为乘法运算在多个周期内实现，因此我们需要向外面输出乘法停止信号（mul\_stall），那么同时我们需要确定乘法的开始与结束状态；

    always@(posedge clk) begin

        if (rst) begin

            mul\_stall = 0;

            count = 0;

            mul\_out = 0;

        end

        else if(alu\_control == `ALU\_SIGNED\_MULT || alu\_control == `ALU\_UNSIGNED\_MULT) begin

            if(count == 5) begin

                mul\_stall = 0;

                count = 0;

                mul\_out = 1;

            end

            else if(mul\_out == 0) begin

                mul\_stall = 1;

                count = count + 1;

            end

            else begin

                mul\_stall = mul\_stall;

                count = count;

            end

        end

        else begin

            mul\_stall = 0;

            count = 0;

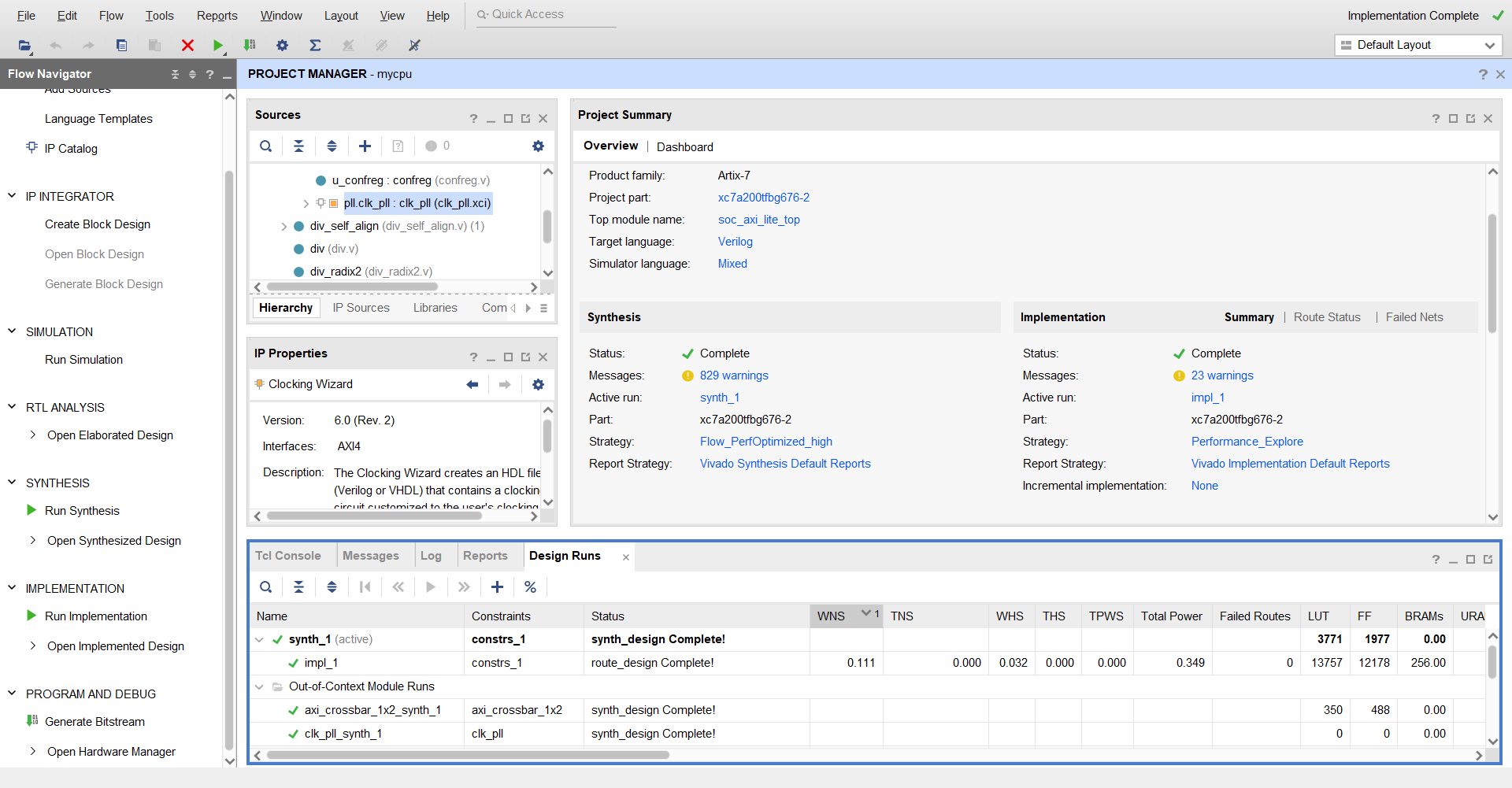
            mul\_out = 0;

        end

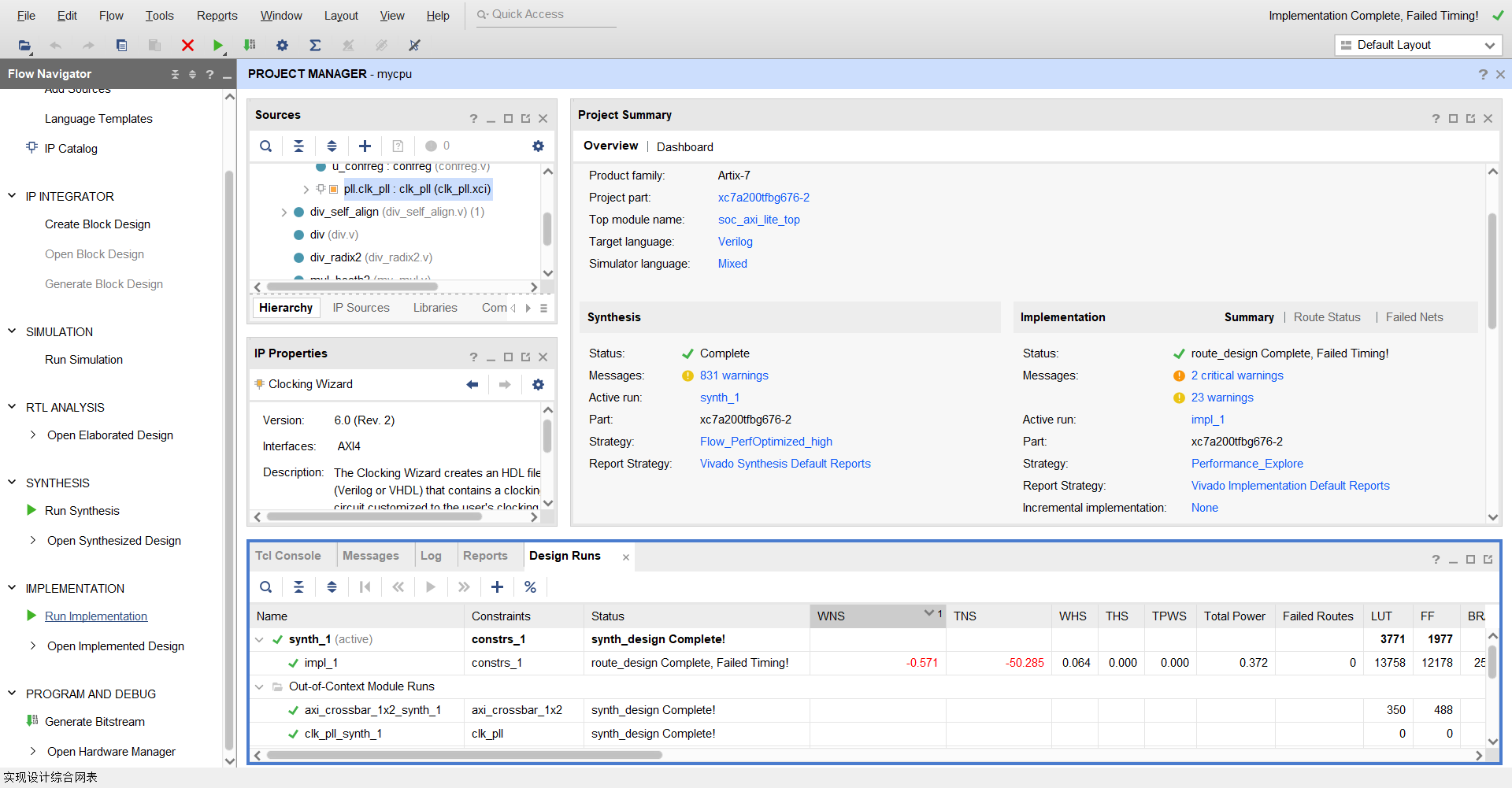
    end

# 测试效果：

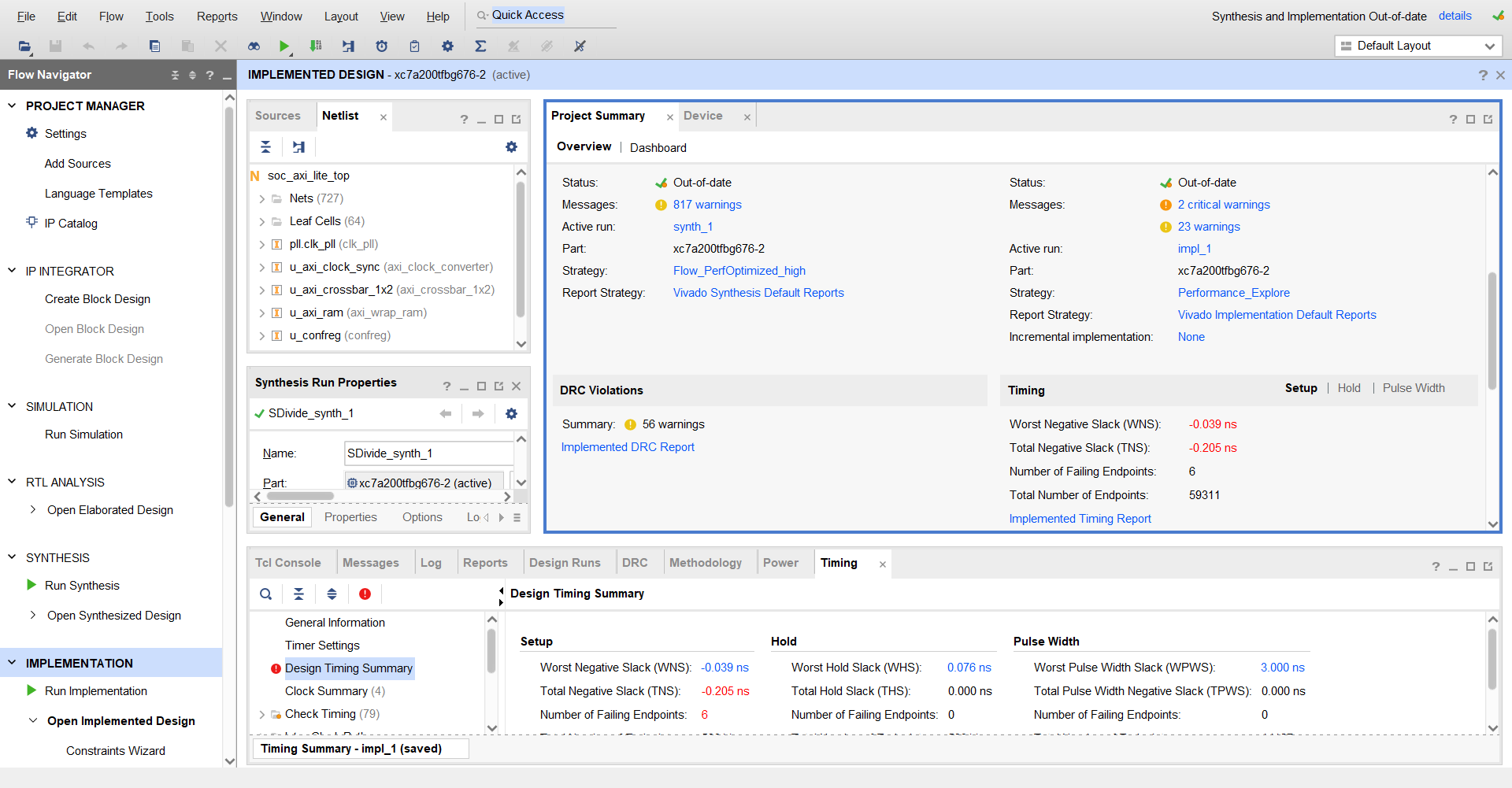
除法ip + 5级乘法流水/50HZ: 裕量：0.111



除法ip + 5级乘法流水/55HZ: 裕量：-0.571



除法ip + 单周期booth乘法/50HZ：裕量：-0.039



其中，无符号除法运算运算到输出有效需要34个周期，加上一个周期用于判断除法开始，以及一个周期用于判断结束，共36个周期。同理有符号除法需要38个周期。五级流水乘法ip共需要6个周期，其中1个周期用于接受数据，4个周期用于计算，1个周期用于输出数据。

综上所述，乘法ip与除法ip在减少时延方面上占优势，若对时延的要求比较高，可以牺牲周期数来提高频率；另外，除法ip功能仿真成功，除法ip与乘法ip结合卡在异常指令，由于时间关系，放在以后重构代码的时候处理…………