# 目录

[目录 1](#_Toc408063766)

[原理说明 1](#_Toc408063767)

[硬件原理图与PCB图 2](#_Toc408063768)

[FPGA芯片 3](#_Toc408063769)

[复位 3](#_Toc408063770)

[时钟 4](#_Toc408063771)

[PCB图 4](#_Toc408063772)

[综合、仿真结果 4](#_Toc408063773)

[仿真步骤 6](#_Toc408063774)

[文件说明 10](#_Toc408063775)

[原理图 10](#_Toc408063776)

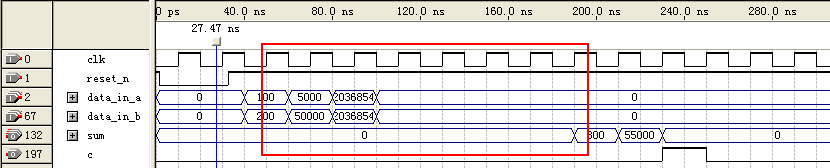
[工程 10](#_Toc408063777)

[源代码文件 10](#_Toc408063778)

[仿真文件 10](#_Toc408063779)

# 原理说明

64位8级流水线加法器，即是将64位拆成8个8位进行运算，最后将8个8位运算的结果相加得出最后的和和进位位。采用8级流水线进行加法运算，则从第一次输入两个加数的第一个时钟起，需要第8个时钟周期对应的和才输出来，之后源源不断的输入加数，则和也不断的输出，如下图所示：



8级流水线需要将加法运算拆分成8个时钟周期来完成，每个时钟周期需要将前面计算得到的和、还未进行计算的加数进行缓存，由此，比如第1个8位计算得到的和就需要缓存7次，第2个8位计算得到的和就需要缓存6次，以此类推。同时，还要将未进行计算的加数进行缓存，比如[63:56]这个8位就需要缓存7次，[55:48] 这个8位就需要缓存6次。

第1个时钟周期：计算第1个8位的和，并加上前一个的进位位。缓存前面得到的和、未进行计算的加数

第2个时钟周期：计算第2个8位的和，并加上前一个的进位位。缓存前面得到的和、未进行计算的加数

第3个时钟周期：计算第3个8位的和，并加上前一个的进位位。缓存前面得到的和、未进行计算的加数

第4个时钟周期：计算第4个8位的和，并加上前一个的进位位。缓存前面得到的和、未进行计算的加数

第5个时钟周期：计算第5个8位的和，并加上前一个的进位位。缓存前面得到的和、未进行计算的加数

第6个时钟周期：计算第6个8位的和，并加上前一个的进位位。缓存前面得到的和、未进行计算的加数

第7个时钟周期：计算第7个8位的和，并加上前一个的进位位。缓存前面得到的和、未进行计算的加数

第8个时钟周期：计算第8个8位的和，并加上前一个的进位位。缓存前面得到的和、未进行计算的加数

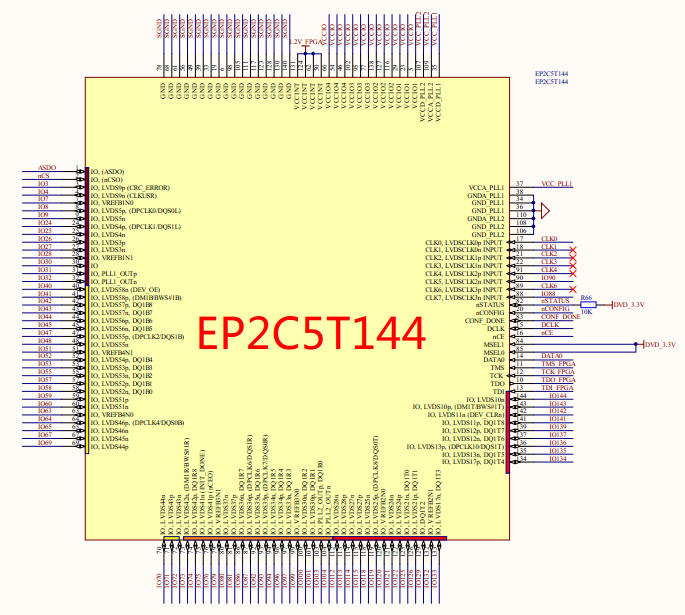
并计算出最后的结果。

8级流水线对应8个时钟周期，对应代码的8个always模块。

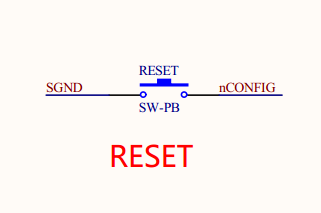
# 硬件原理图与PCB图

64位8级流水线加法器只需FPGA最小系统：

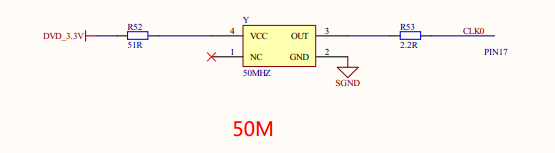
## FPGA芯片



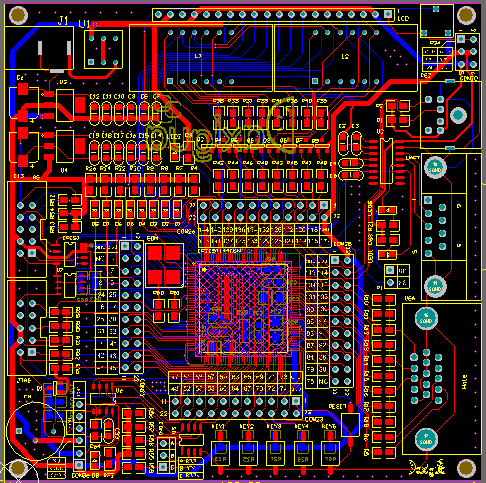
## 复位



## 时钟



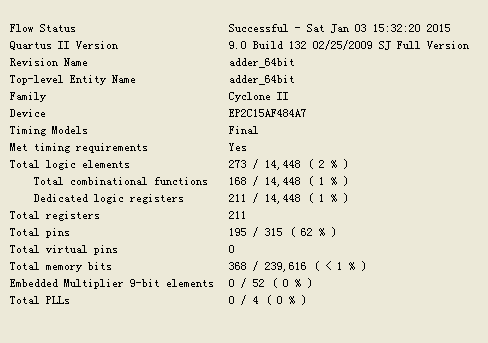
## PCB图



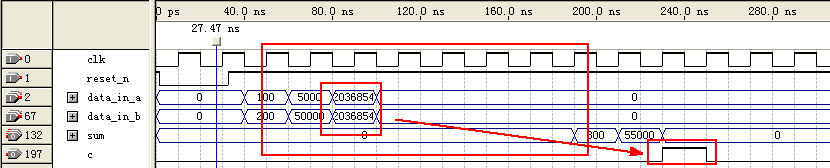
# 综合、仿真结果

综合结果如下，全编译成功，用到的资源如下：

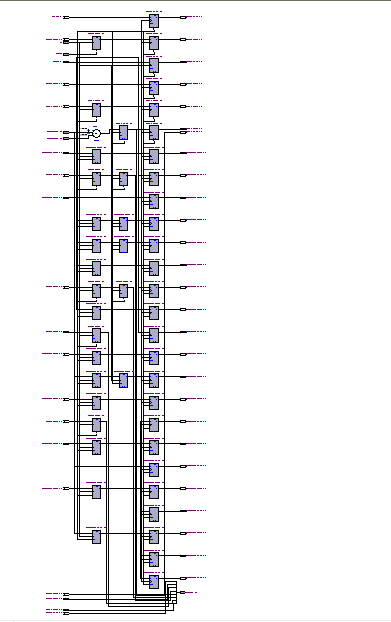




仿真结果如下，图中箭头所指的结果为2^63+2^63=2^64，即和为0，进位位为1

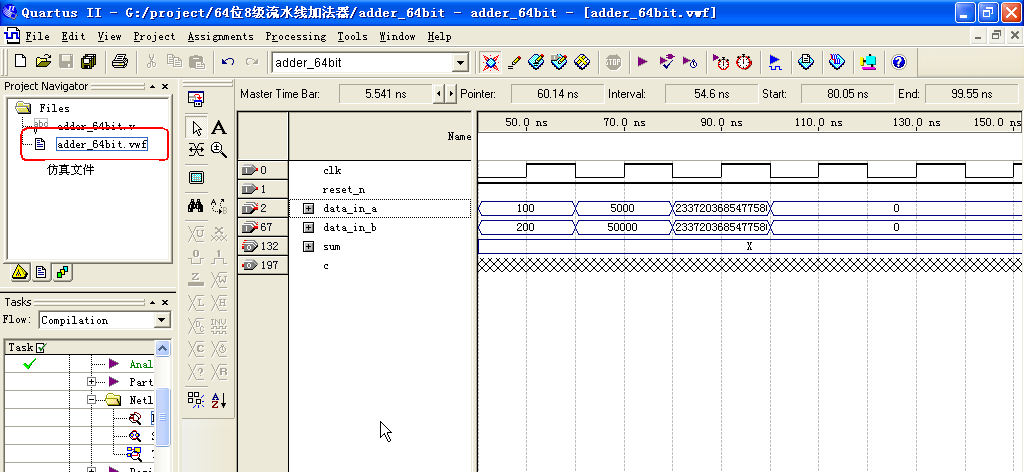


RTL视图如下

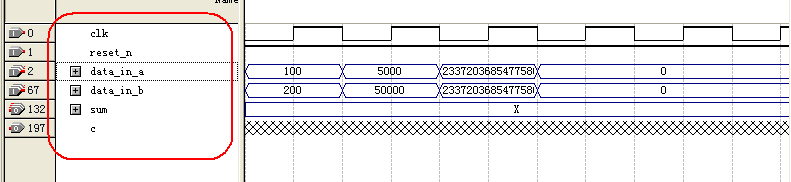


# 仿真步骤

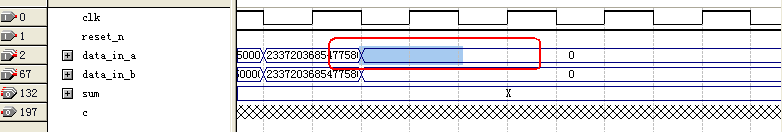
先打开工程，双击仿真文件。

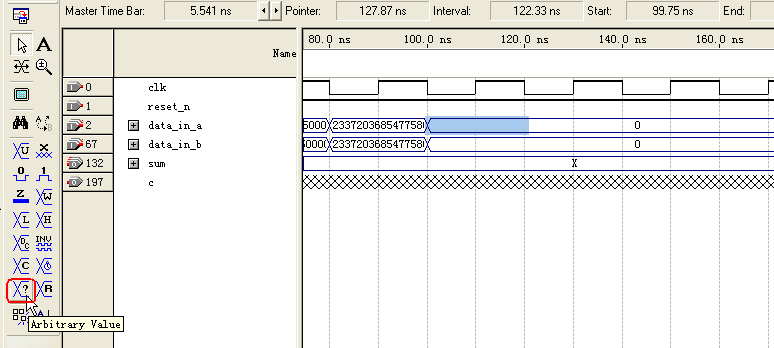


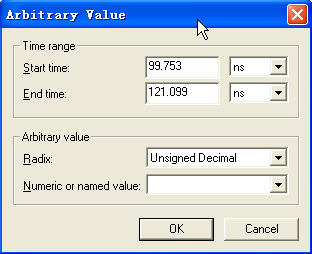
Clk、reset\_n、data\_in\_a、data\_in\_b已经设置好了，可以在data\_in\_a、data\_in\_b上再添加一些数。



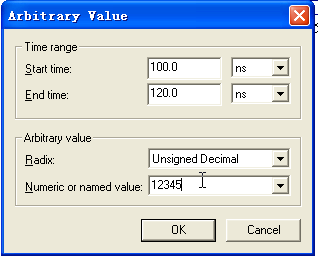
用鼠标选中摸个区域，出现蓝色阴影。



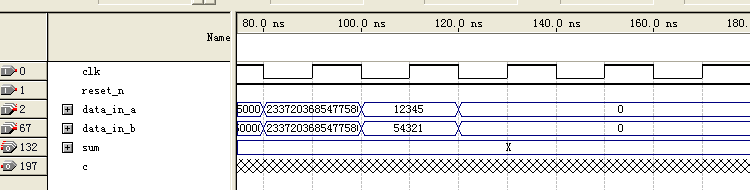




将参数改为如下：时钟周期为20ns，所以开始和结束必须间隔20ns

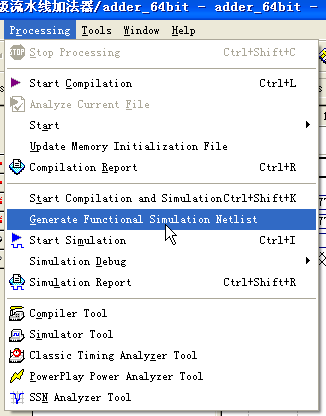


同样地，按照这样设置data\_in\_b



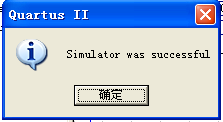
没有编译过的需要按下，

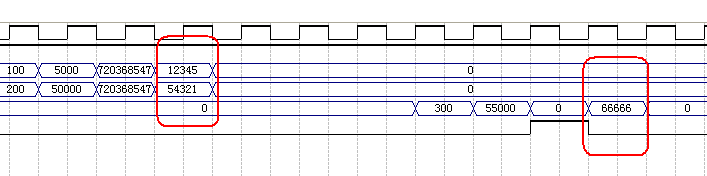
接下来生成网表





最后点击进行仿真





结果正确

# 文件说明

## 原理图



## 工程



## 源代码文件



## 仿真文件

