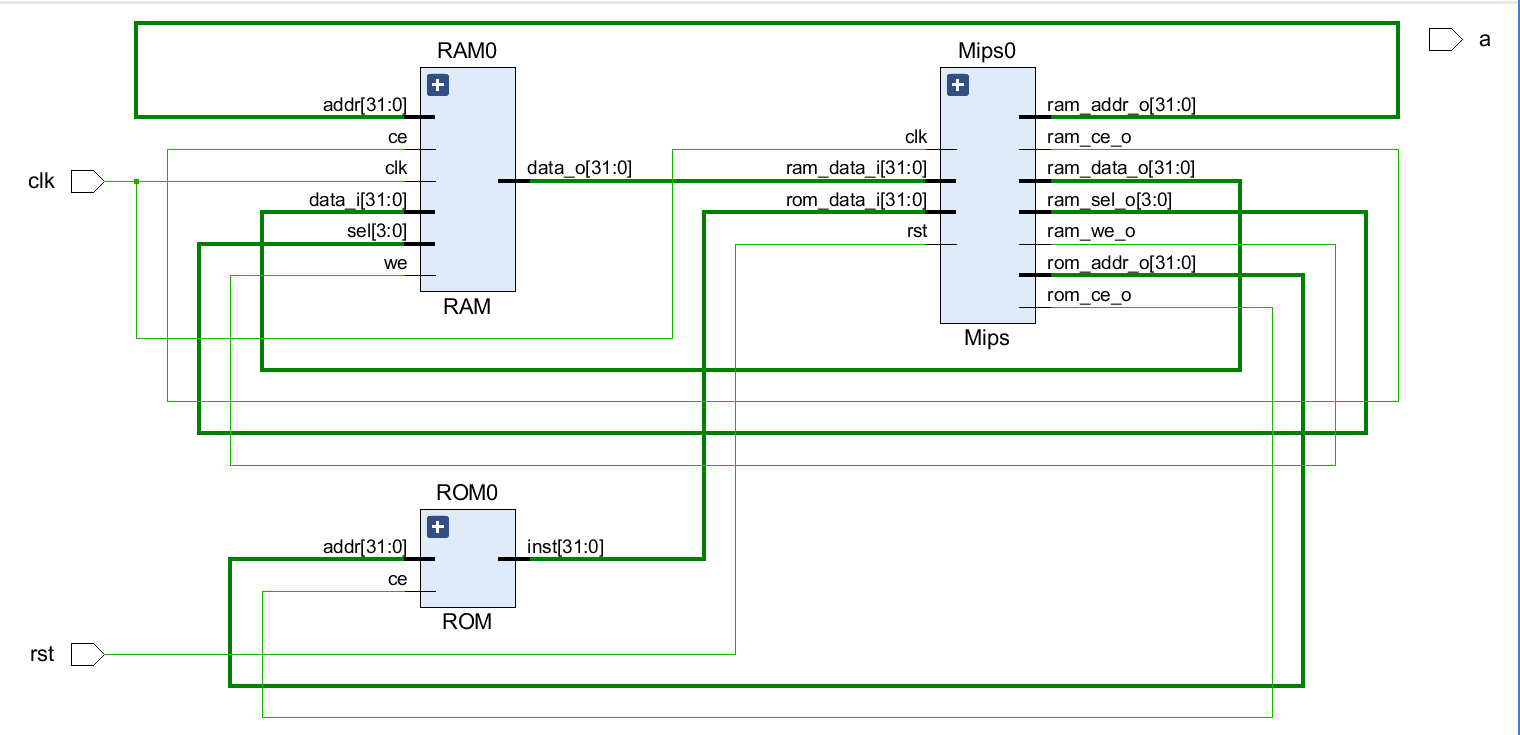
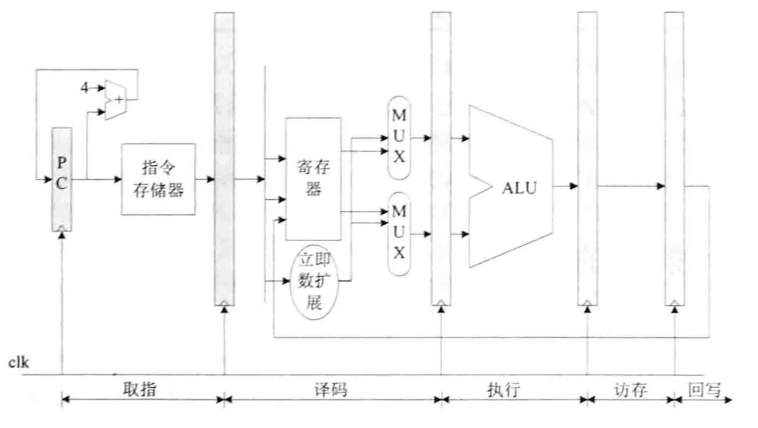
**一.处理器简介**

本处理器是基于Verilog，ISA为mips指令集，微架构为经典的取指，译码，执行，访存，回写五级流水的risc处理器。



Cpu框架图（ROM+RAM+Mips）



五级流水架构（此图为初步图，缺少一些模块）

**具体模块有：**

**（1）取指**

PC:给出指令地址。

IF/ID:取指与译码的流水分割。

**（2）译码**

ID：译码。

Regfile：通用寄存器。

ID/EX:译码与执行的流水分割。

**（3）执行**

EX:根据译码结果执行相应的操作。

DIV：除法器。

EX/MEM:执行与访存的流水分割。

**（4）访存**

MEM:对数据存储器进行访存。

MEM/WB:访存与回写的流水分割。

**（5）回写**

HILO：除法指令中使用。

**（6）流水线暂停**

CTRL：流水线暂停模块

**（7）存储**

ROM:存指令。

RAM:存数据。

**实现的指令有：**

逻辑操作指令：and，andi，or，ori，xor，xori，nor，lui；

移位操作指令：sll，sllv，sra，srav，srl，srlv；

移动操作指令：movn，movz，mthi，mflo，mtlo

算术操作指令：add，addi，addiu，addu，sub，subu，clo，clz，slt，slti，sltiu，sltu，mul，mult，mult，multu，div，divu；

转移指令：jr，jalr，j，jal，b，bal，beq，bgez，bgezal，bgtz，blez，bltz，bltzal，bne；

加载存储指令：lb，lbu，lh，lhu，ll，lw，lwl，lwr，sb，sh，sw，swl，swr；

**测试文件：**

通过学习汇编语言来完成指令的转换，我们采用在虚拟机linux系统上装上GNU交叉编译工具链来完成汇编语言的编译，得到机器码，输入rom。

**未来：**

接下来，我们计划完成mips剩下指令的加入，以及cache的加入，以增强cpu性能。

**二.设计方案**

**1.****完成一条指令的五级流水**

在本次实验中，我们选取的是相对简单的ori指令来完成五级流水的设计，之后在一条指令的五级流水架构的基础上完成其他指令的实现。

这里只展示我们的测试文件以及测试波形（在另一个word测试代码以及波形中），具体代码详情可见lab1文件夹。

**2.加入逻辑，移位以及空指令操作**

在加入这些指令之前首先要解决的一个问题便是数据冲突，这里我们经过分析，在做逻辑，移位以及空指令时应该只会存在RAW（Read After Write）这一种情况，我们拟采用数据前推的方法解决，这是因为新的寄存器的值可以在访存阶段计算出来。

这里只展示我们的测试文件以及测试波形（在另一个word测试代码以及波形中），具体代码详情可见lab2文件夹。

**3.移动操作指令：**

由于部分移动操作指令需要操作HI,LO两个寄存器，所以在这里除了实现移动操作指令外，还会实现HILO模块。

这里只展示我们的测试文件以及测试波形（在另一个word测试代码以及波形中），具体代码详情可见lab3文件夹。

**4.算术操作指令：**

算术操作指令分为两类：

第一类为简单算术操作指令，即除了除法指令外其他指令，因为这些指令只需要一个时钟周期就能完成

第二类为除法算术操作指令，这里我们采用试商法完成除法器的设计，那么对于32位除法，就需要32个时钟周期才能完成除法指令，所以还需要在这里实现CTRL模块，由于此模块较简单，所以并未在此单独仿真测试，而是选择在除法器模块一起仿真测试。

这里只展示我们的测试文件以及测试波形（在另一个word测试代码以及波形中），具体代码详情可见lab4文件夹。

**5.转移指令的实现**

转移指令需要注意延迟槽，简单来说就是转移指令的下一条依旧会实现，这就是我们为什么要在之前实现空指令的原因。

这里只展示我们的测试文件以及测试波形（在另一个word测试代码以及波形中），具体代码详情可见lab5文件夹。

**6.加载存储指令：**

由于加载存储指令涉及在RAM模块的存储以及取出，所以我们在这里会实现RAM模块，同样也不会给出单独给出RAM模块的测试，而是与加载存储指令一起测试。

这里只展示我们的测试文件以及测试波形（在另一个word测试代码以及波形中），具体代码详情可见lab6文件夹。

**7.未来可能会实现**

1.cache模块2.协处理器3.异常处理等等

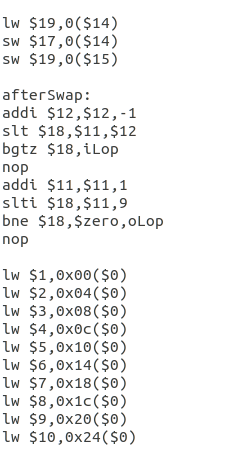
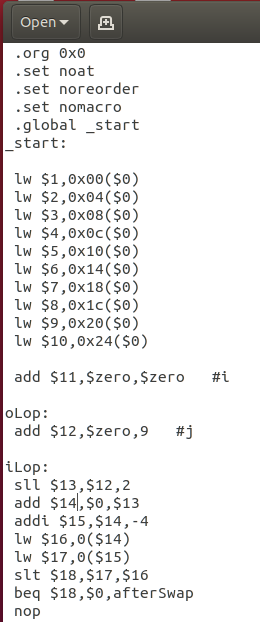
**三.程序测试**

在这里我们使用冒泡排序来验证我们的cpu。

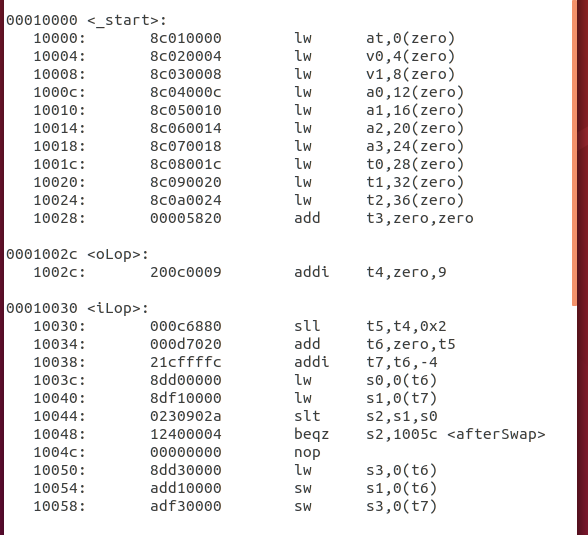
即先在data里面存10-1，然后将他们从小到大重新排列，具体实现如下：

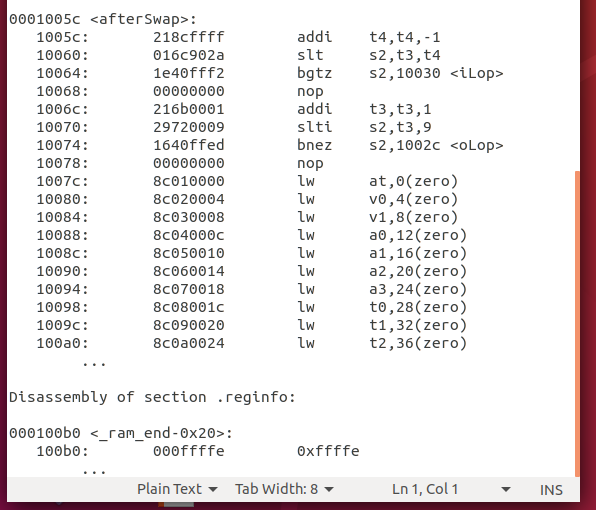
这里只展示我们的测试文件以及测试波形，具体代码详情可见mips-8.src文件夹。

**汇编语言：**

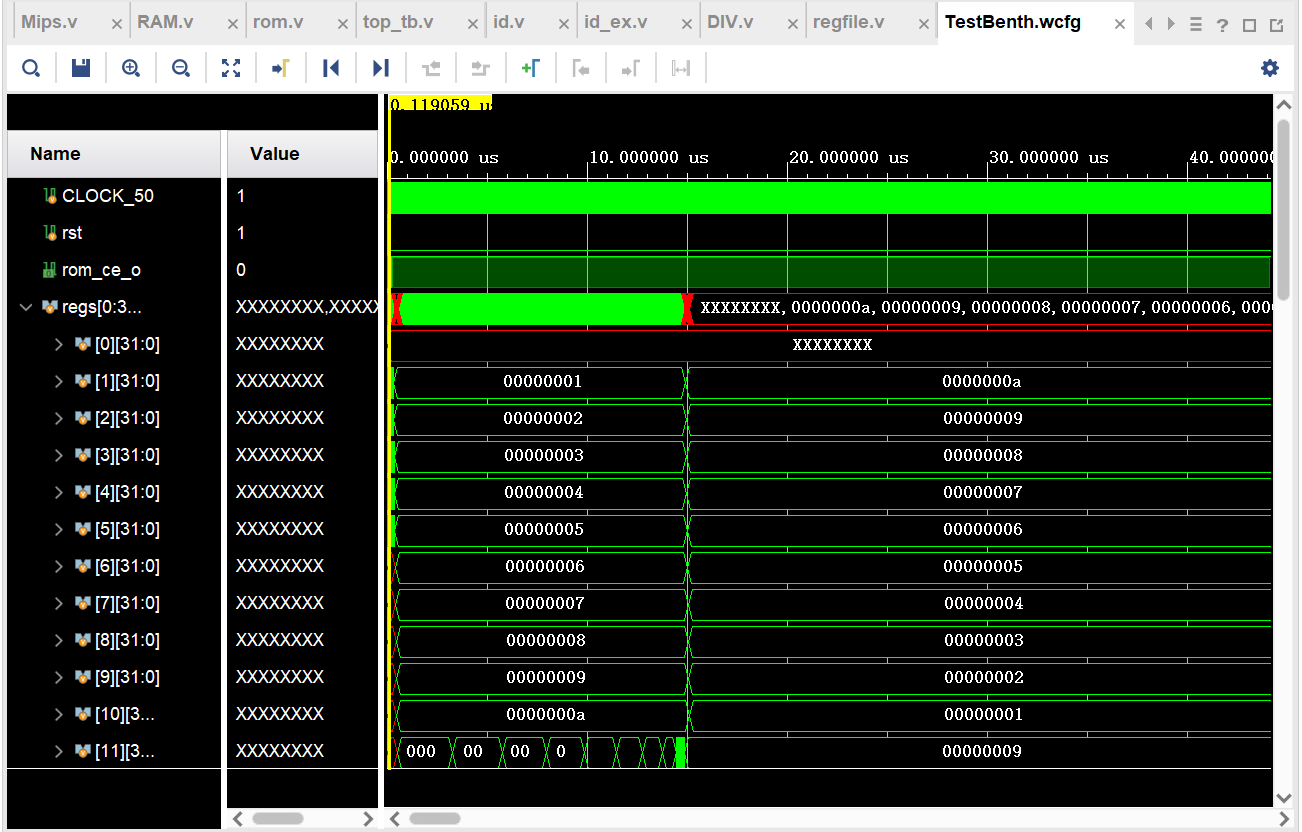
****

**机器码：**

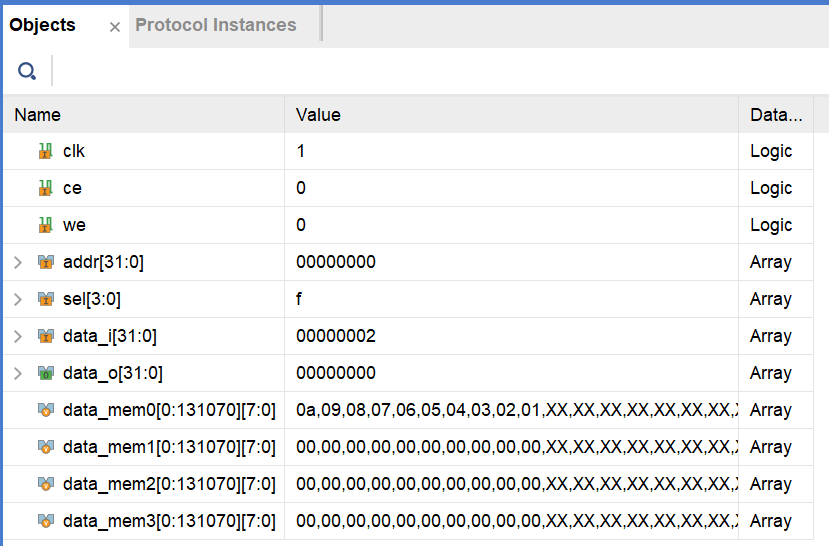
****

****

**仿真测试：**



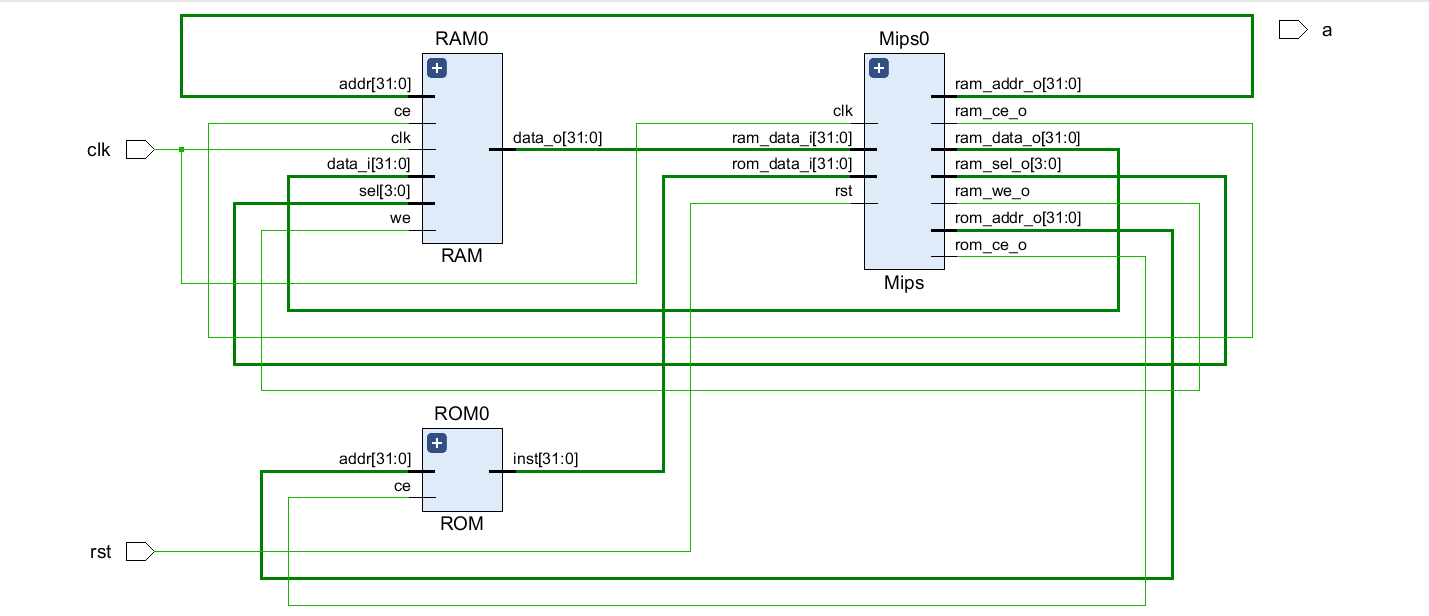
**我们可以从reg[1]-reg[10]的波形很明显地看出，在之前加载出来的是1-10，经过指令过后顺序发生变化，为10-1.**

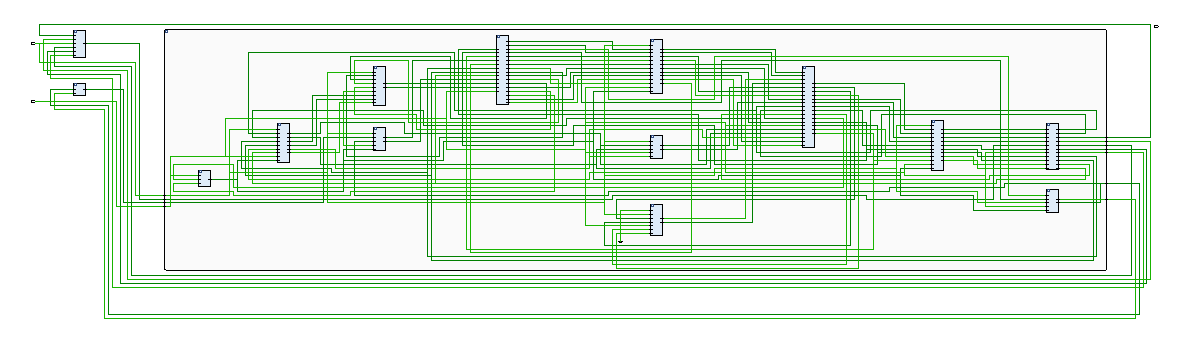


**在RAM中，数据顺序也已经改写，冒泡排序程序基本完成。**

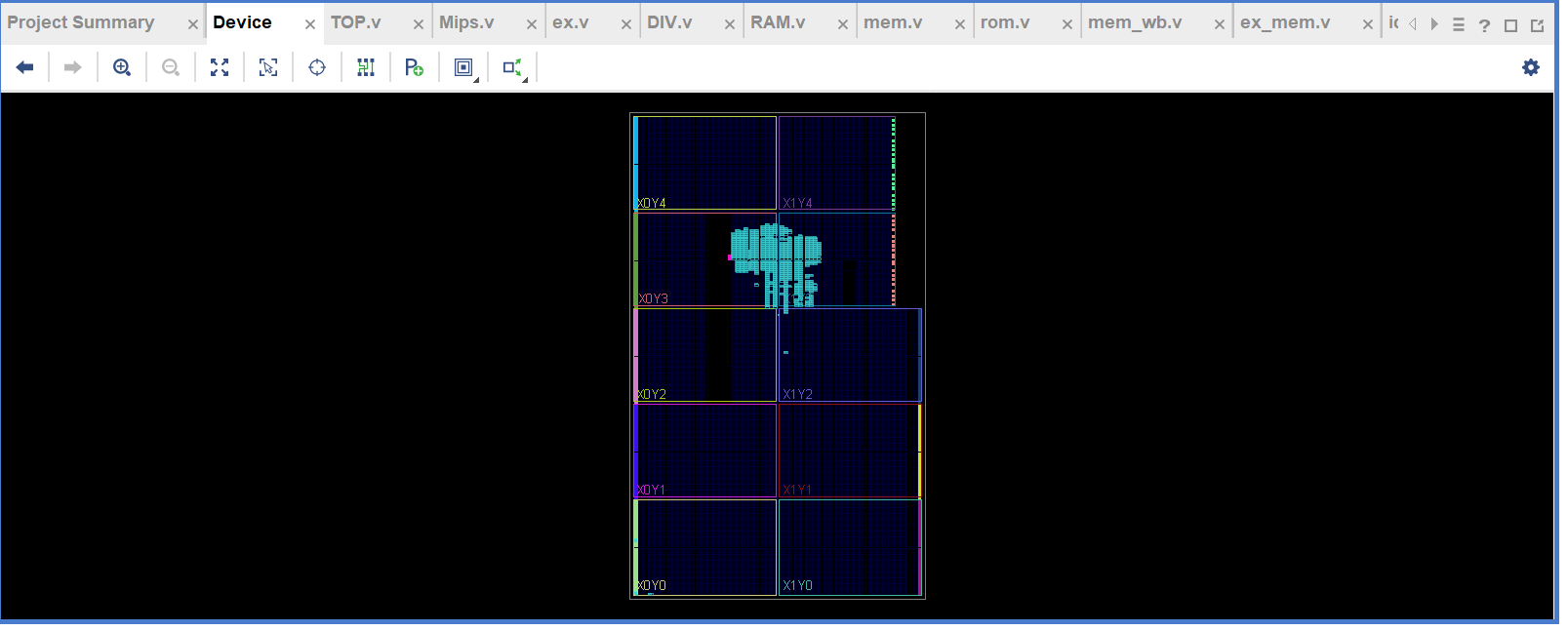
**四.RTL电路以及综合后电路（门级电路）**

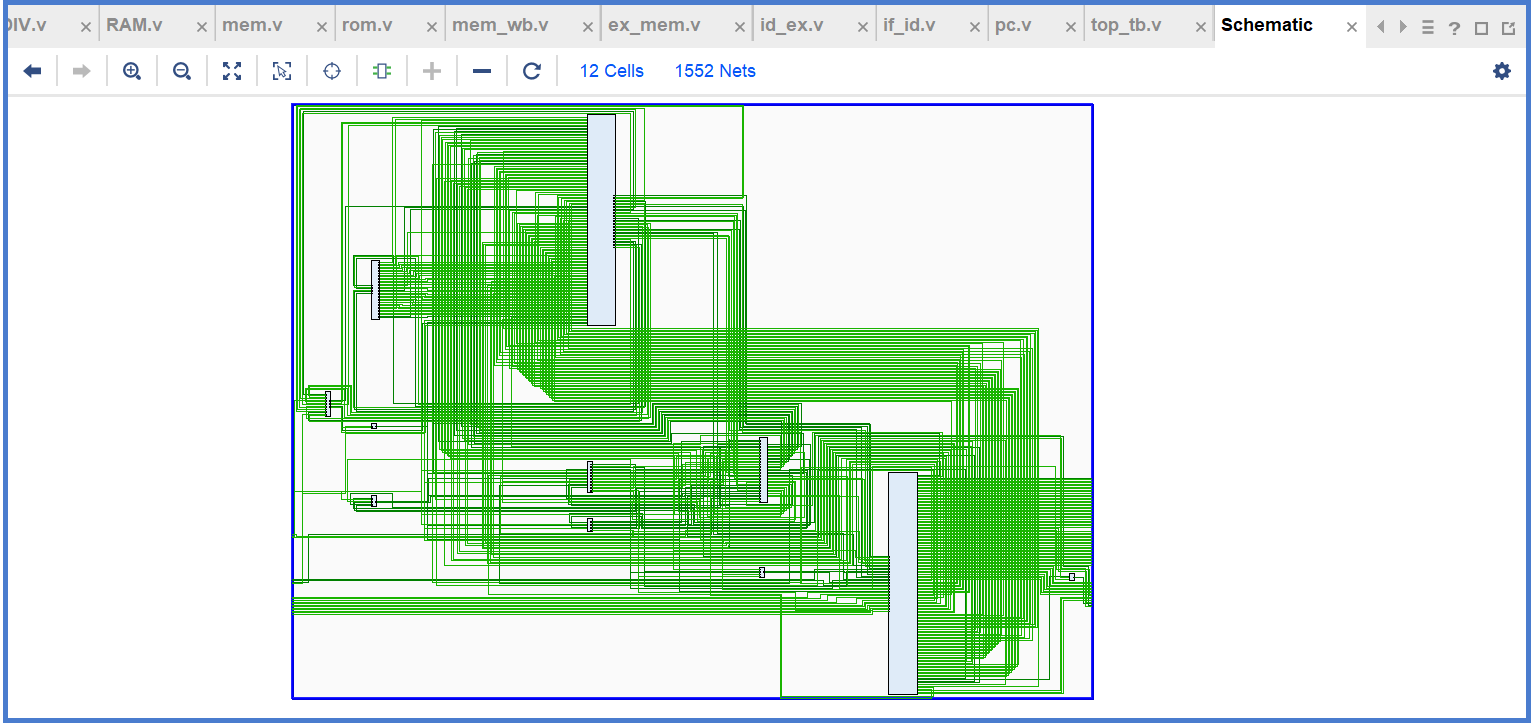
**RTL:**

****

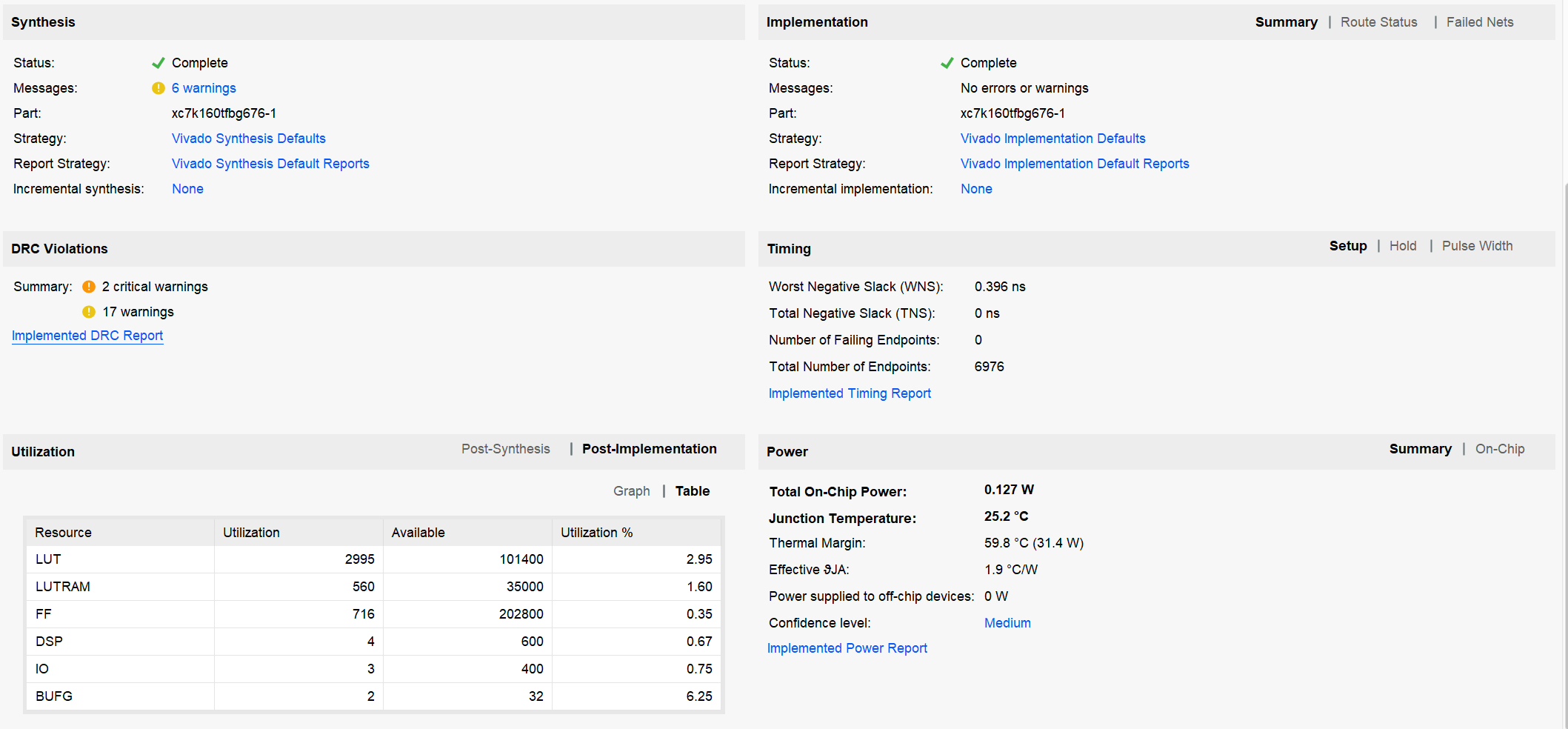
****

**综合后（门级电路）：**

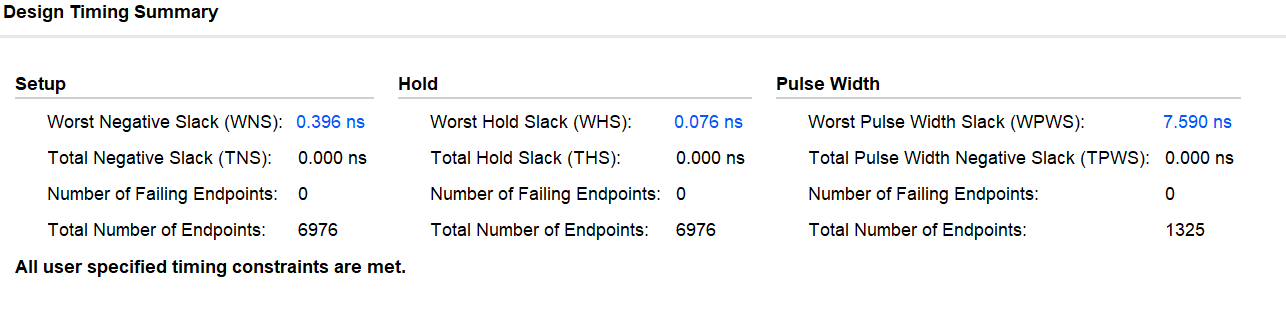
****

****

**五.统计PPA**

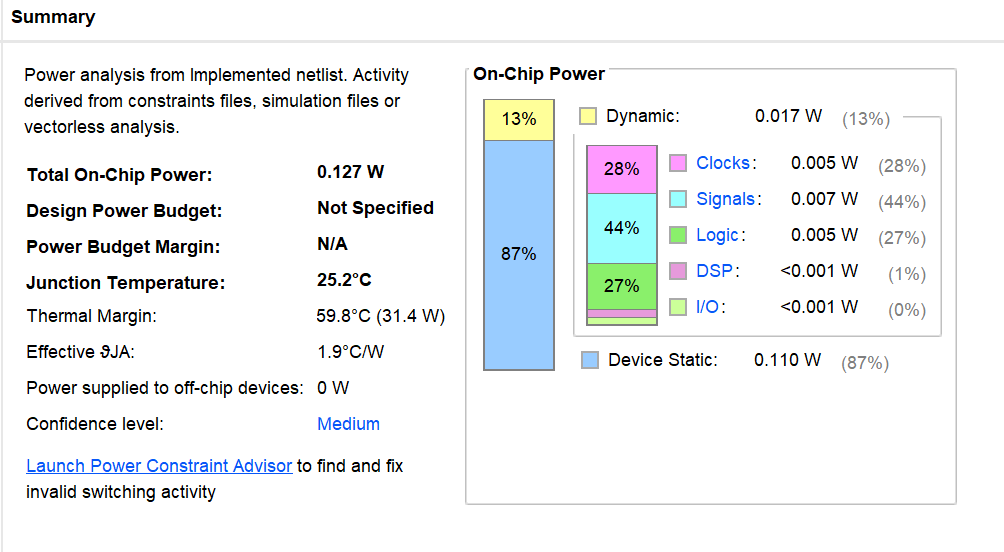
****

**1.性能：**



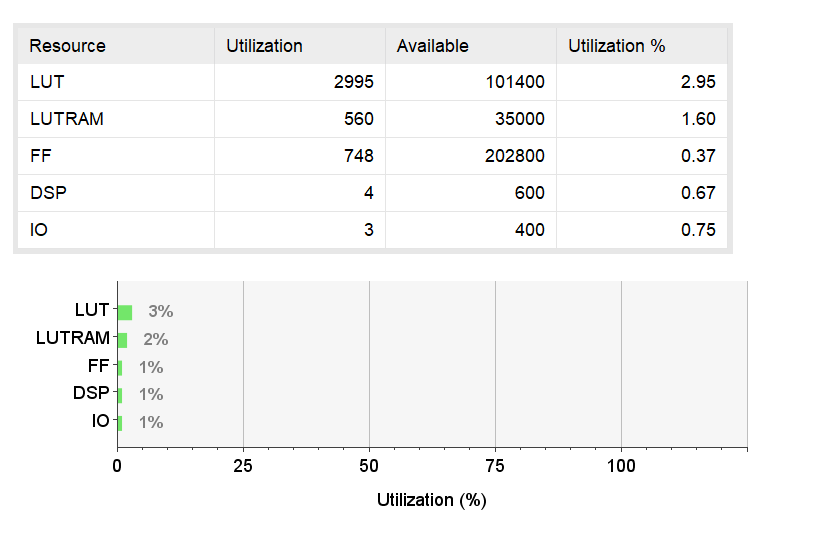
这是在始终约束clk在周期为17ns时的 Timing Summary，可以看出此时没有发现时序错误，同时还仍有一定的余量，所以频率峰值在58.82MHz左右，应该比58.82MHz还要大些

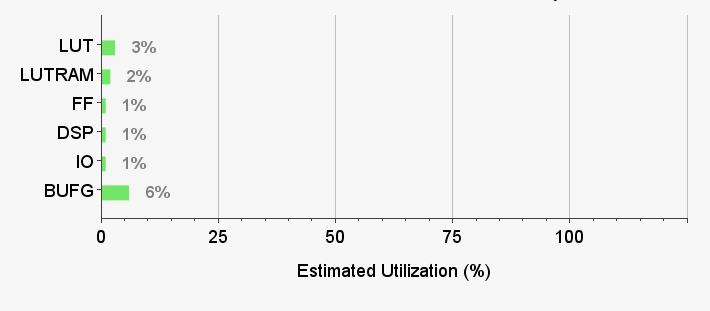
**2.功率：**

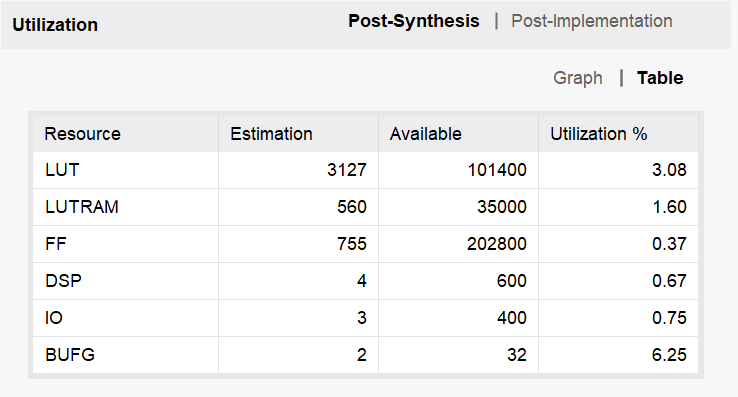
****

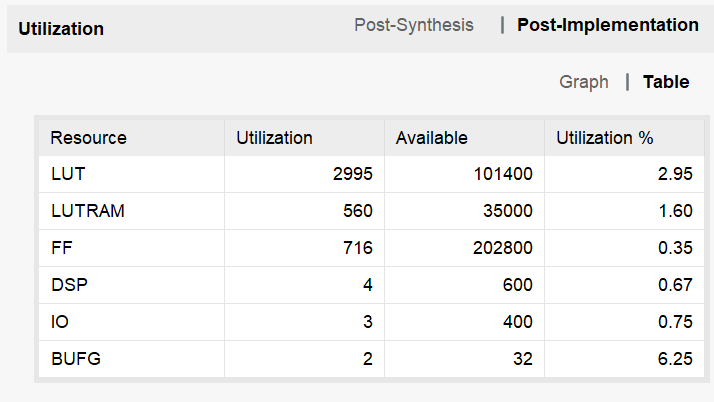
从上图中可以看出总的功率为0.127W，其中利用约0.110W，利用率为87%，从那张小图中又能看出静态功率中各个占比。

**3.硬件开销：**

****

****

****

****

上图显示各个硬件的开销，个人感觉可能是板子选的太大了，所以开销占比挺小的，可以在这个板子上跑通。

**六.心得体会**

**总结下我们犯得一些错误：**

1.TOP以及Mips接线完整导致悬空出现高阻态；

2. Mem文件放错文件夹导致编译出现问题。

3.if语句情况考虑重复，导致出现不确定态

4.汇编语言手滑敲错过，机器码手滑也敲错过。

5. 跳转指令的汇编语言出错：需要填充空指令，这是血的教训，结果在最后编写冒泡排序的时候又忘了，还是看波形看出来的。

6. 有时候不适应16进制导致出错。