Lab1

吴非 贾鑫鹏

A1架构：

Matlab实现：

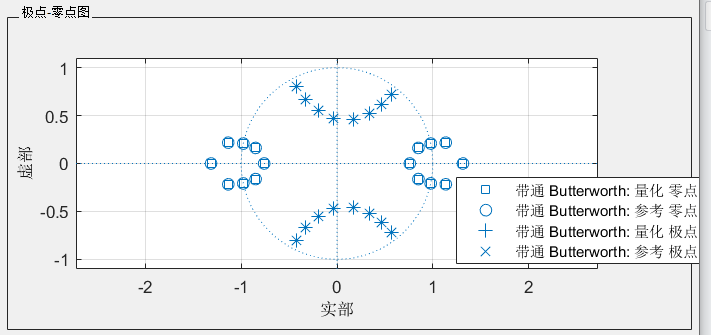


图 1 零极点分布图

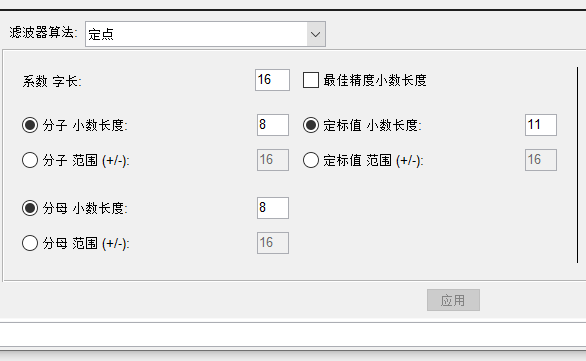


图 2 抽头系数

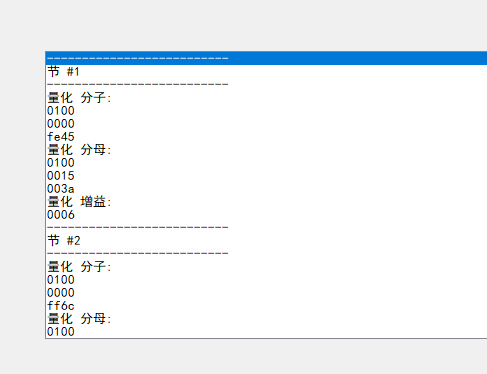


图 3 部分参数

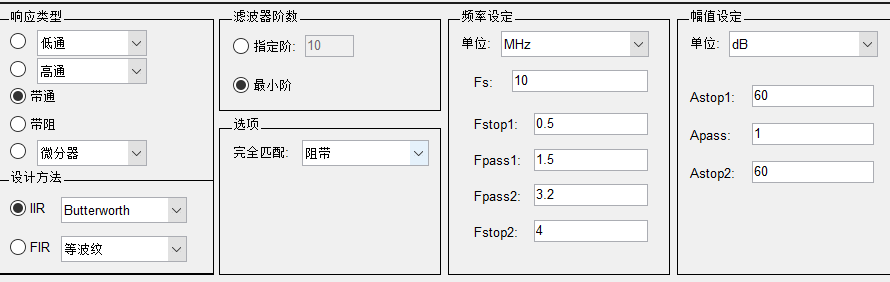


图 4 滤波器设置

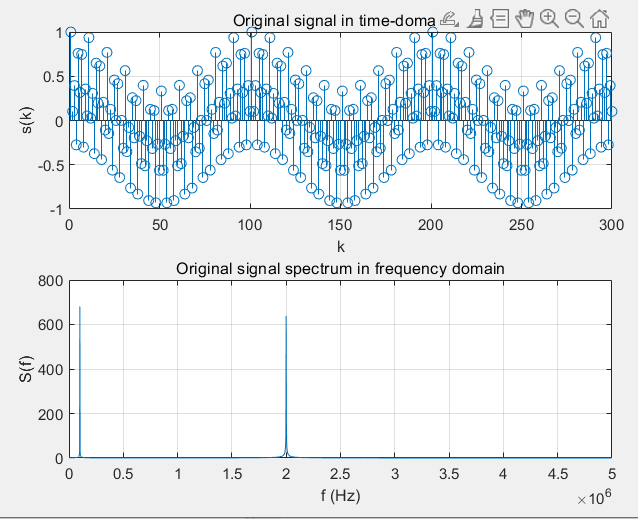


图 5 matlab仿真结果（滤波前）

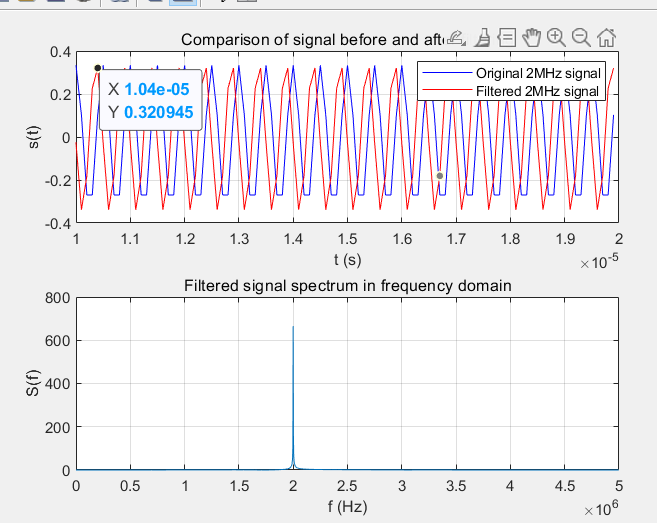
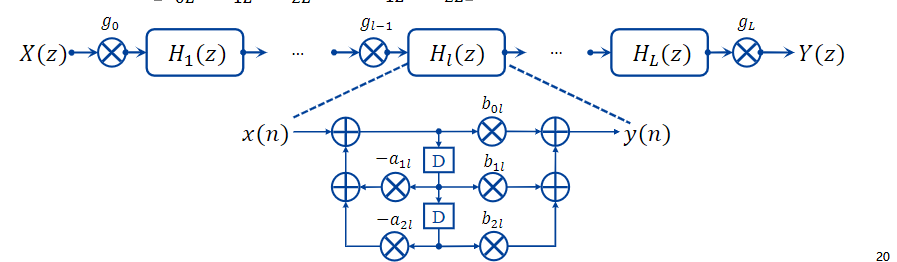
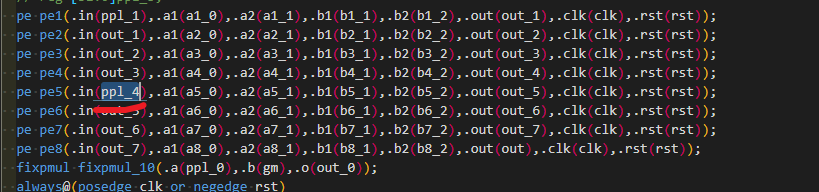


图 6 matlab 仿真（滤波后）

优化前的IIR滤波器电路图A1：（直接II型，八节二阶节）

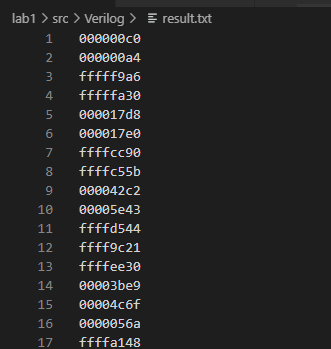


(需要注意的是，为了满足60Mhz的频率要求，在第四和第五节之间插入寄存器（流水线）：

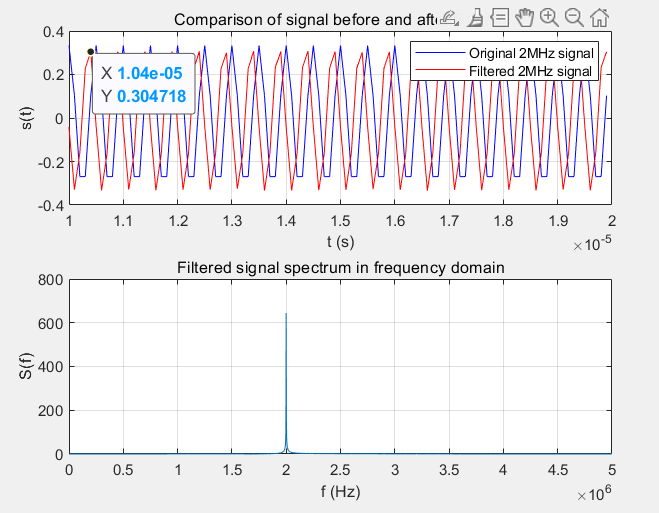


如图所示ppl\_4为寄存器类型，其他的out\_\*为wire类型，直接组合逻辑连接。

Verilog验证结果：



将结果作为输入，用matlab显示波形，结果为：

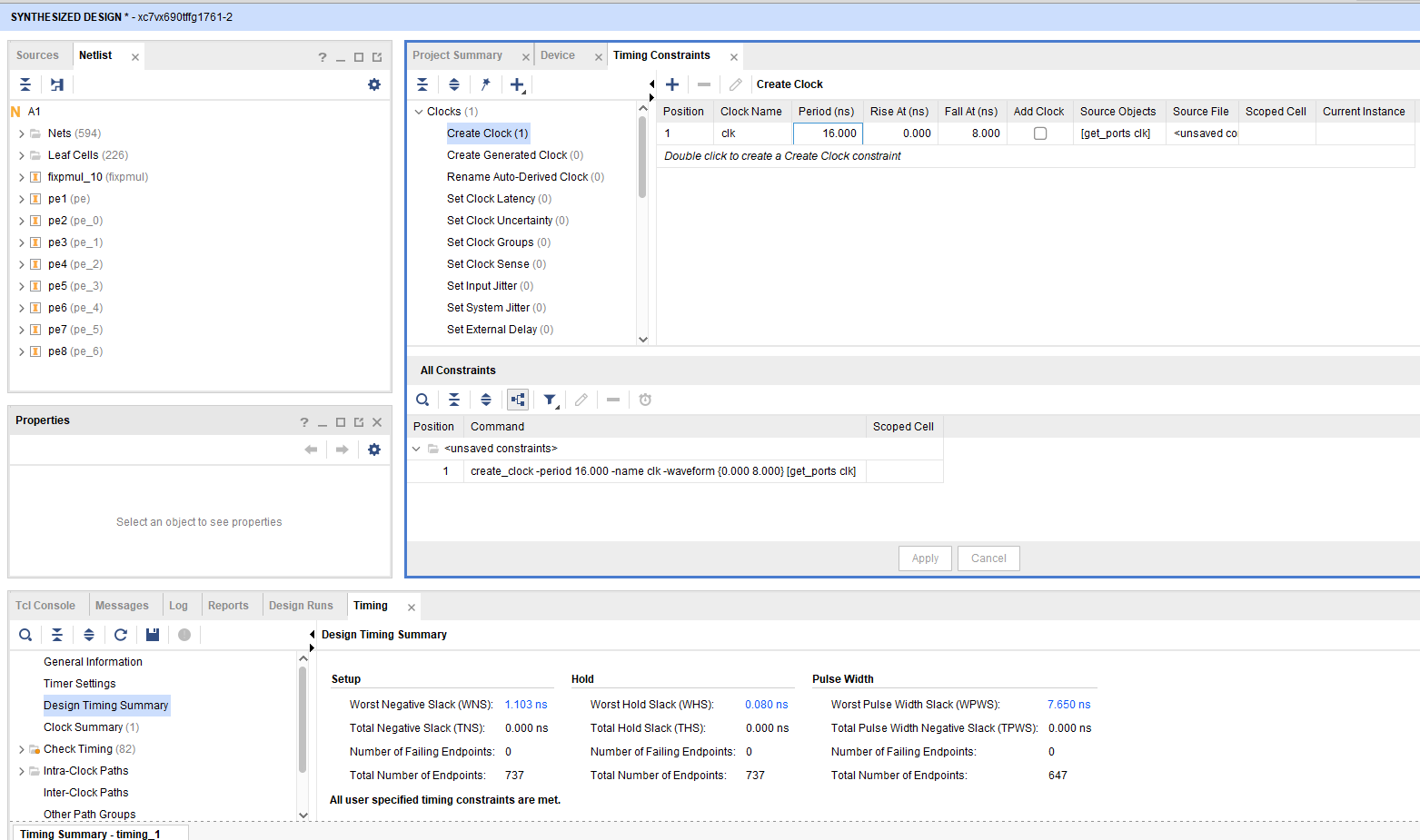


可以看到和初始用matlab模拟的结果一致，因此verilog仿真正确（之后的仿真结果只需要和这个result.txt对比，如果一致那么也一样可以说明仿真成功）

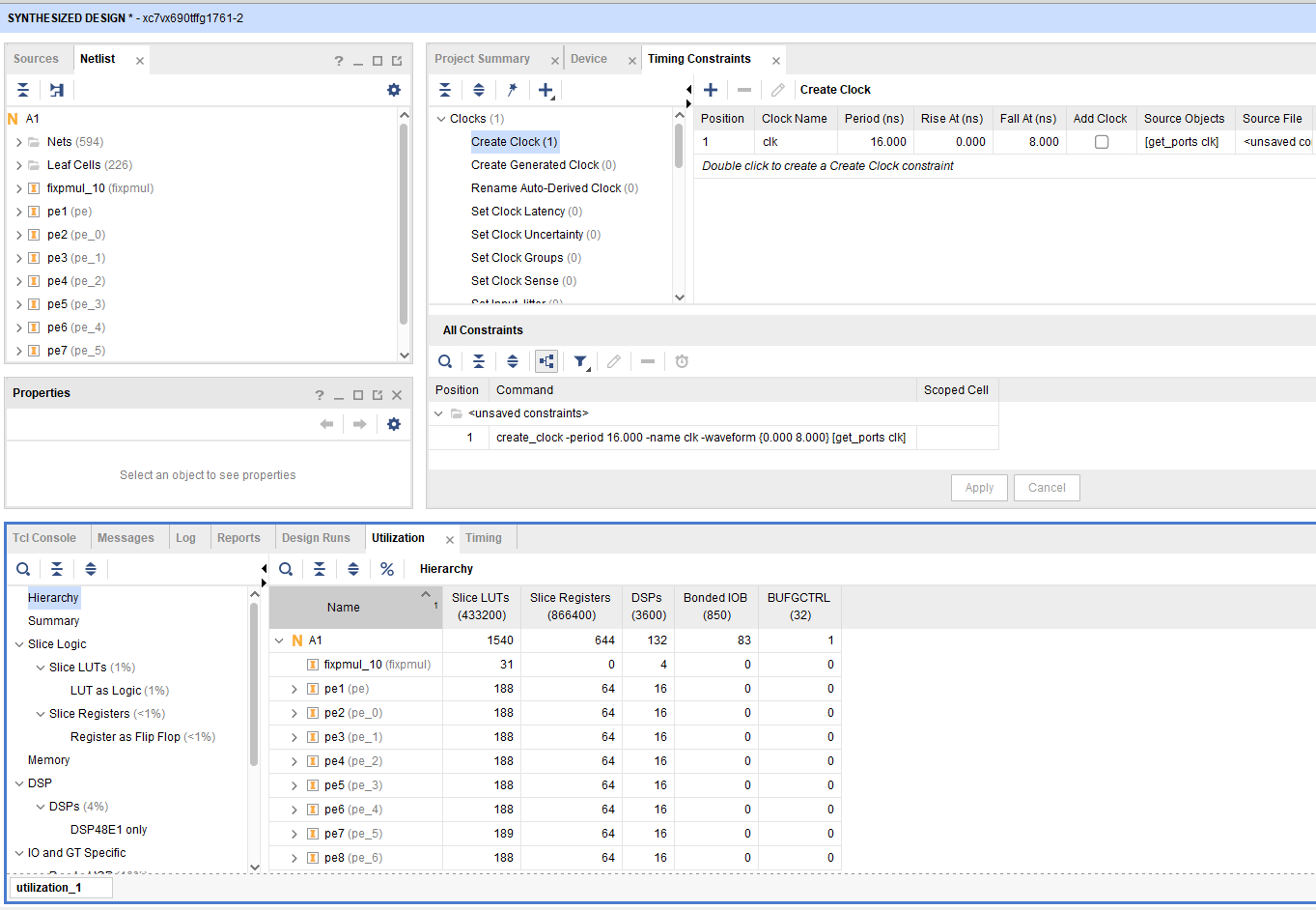
Vivado结果：

Synthesize:

频率：最低要求为60Mhz，周期为16.67ns，可见16ns满足要求。

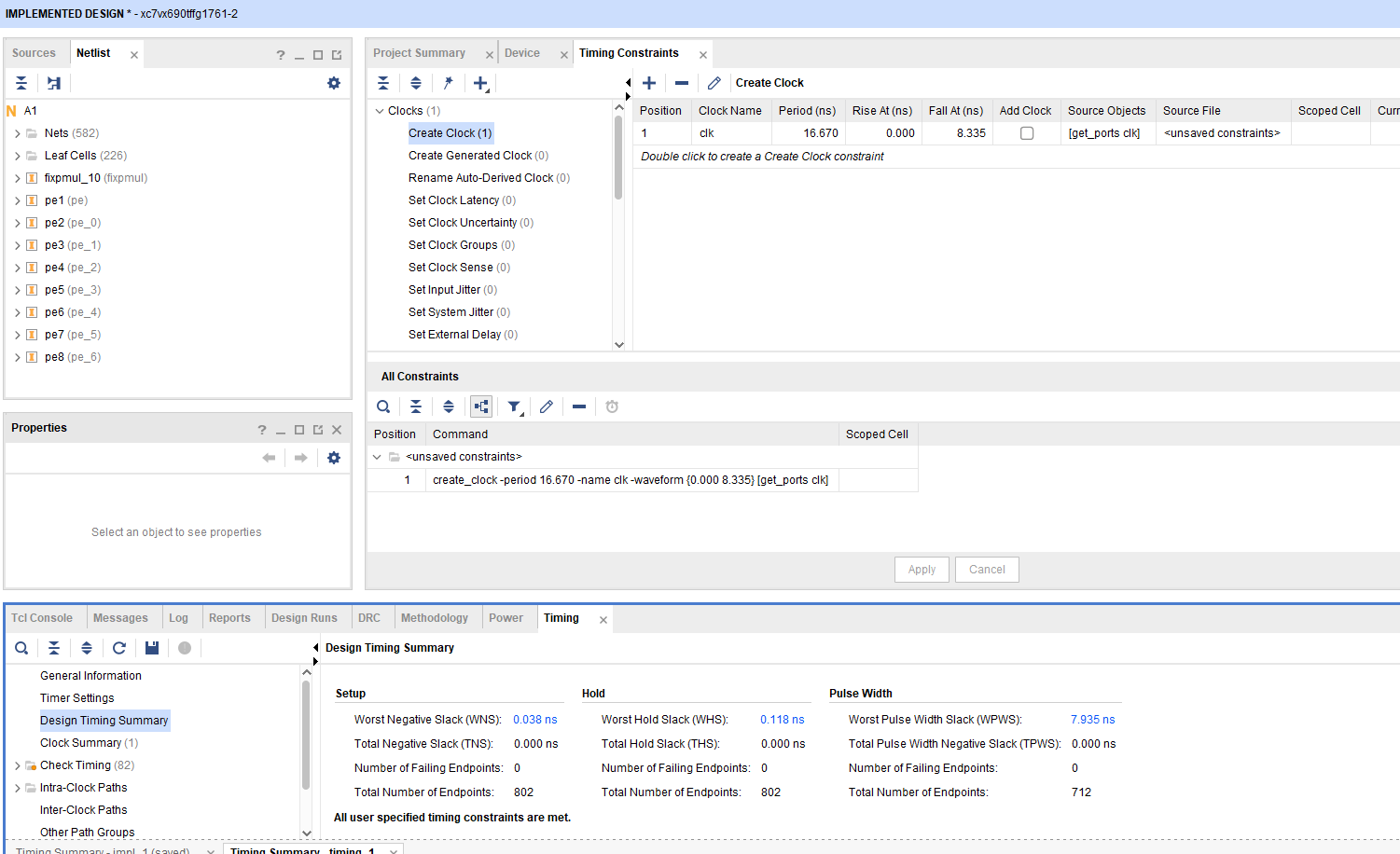


资源情况：

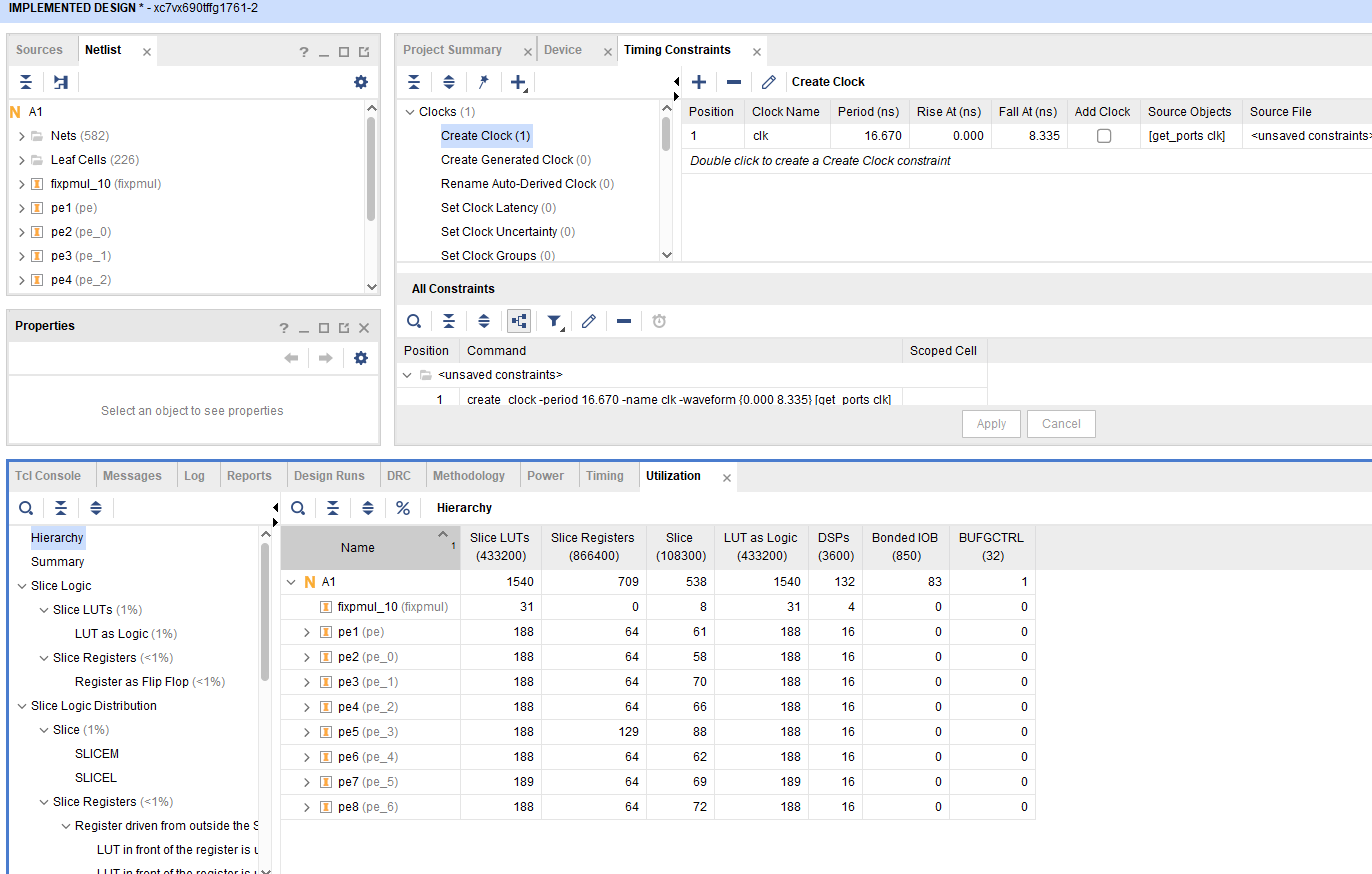


Implemente：

频率：



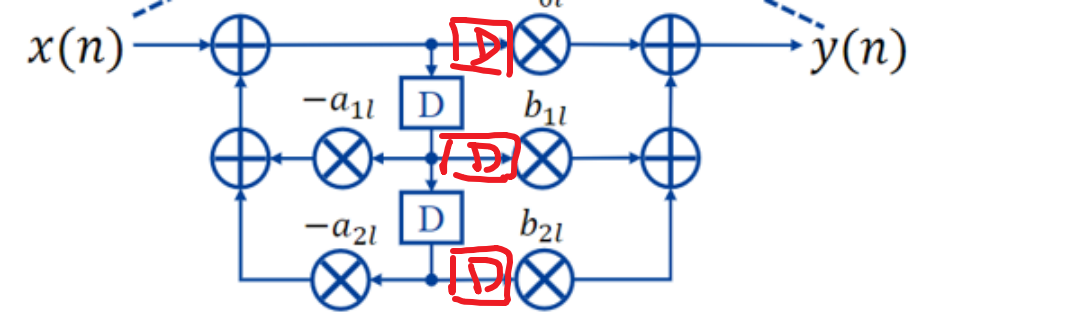
资源：



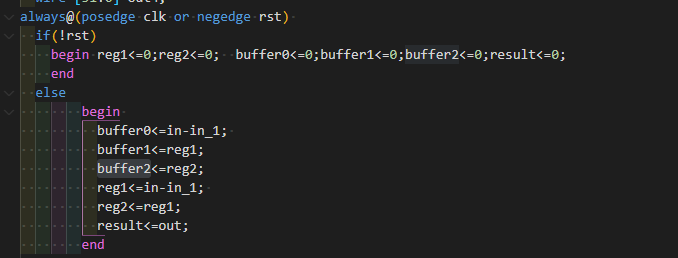
A2架构:

优化后的 （频率为1/2）

在8个处理单元间插入reg，并在每个pe中设置一级流水线（A2架构）



对应代码中的buffer0,buffer1,buffer2:

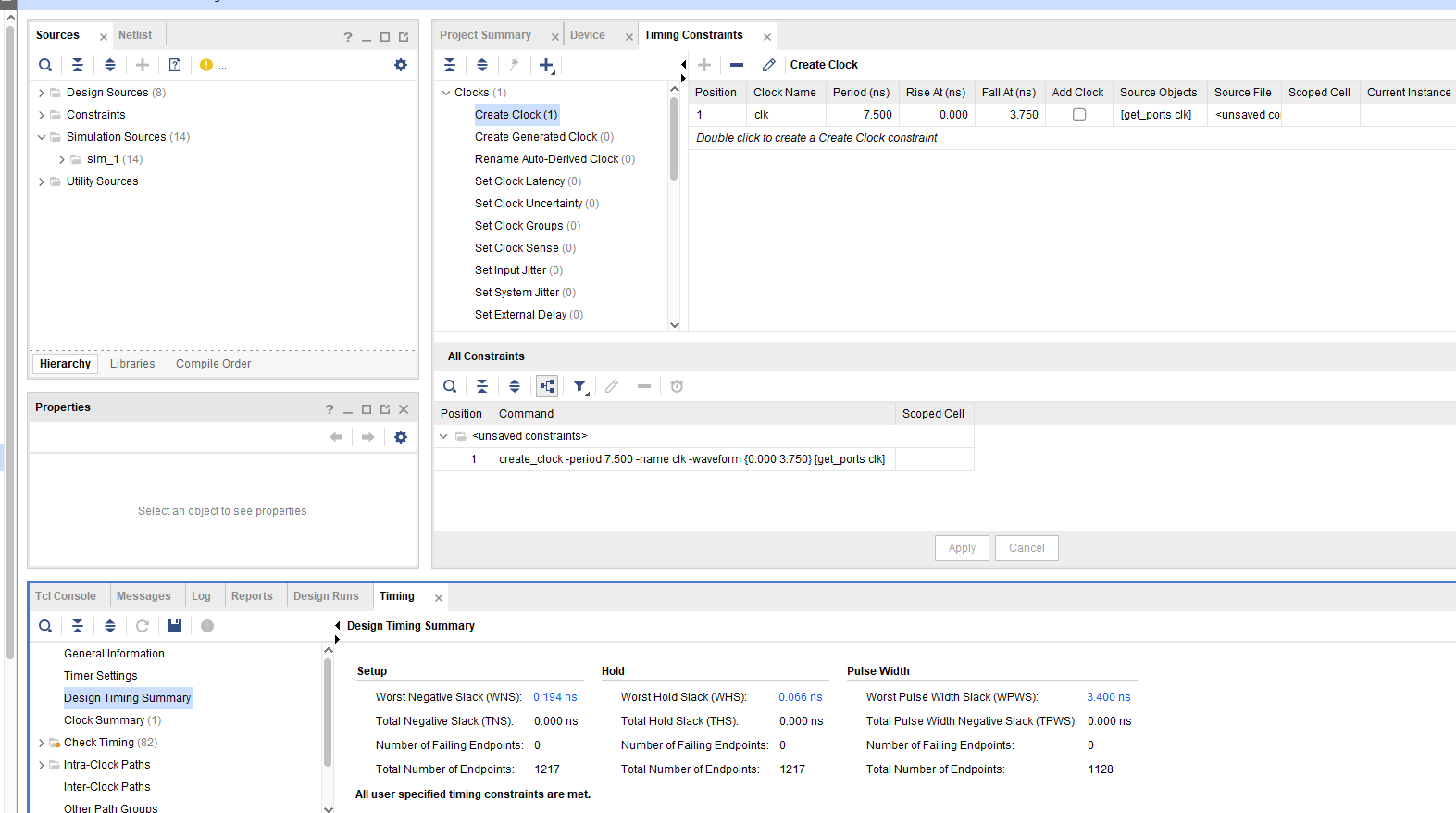


Vivado结果：

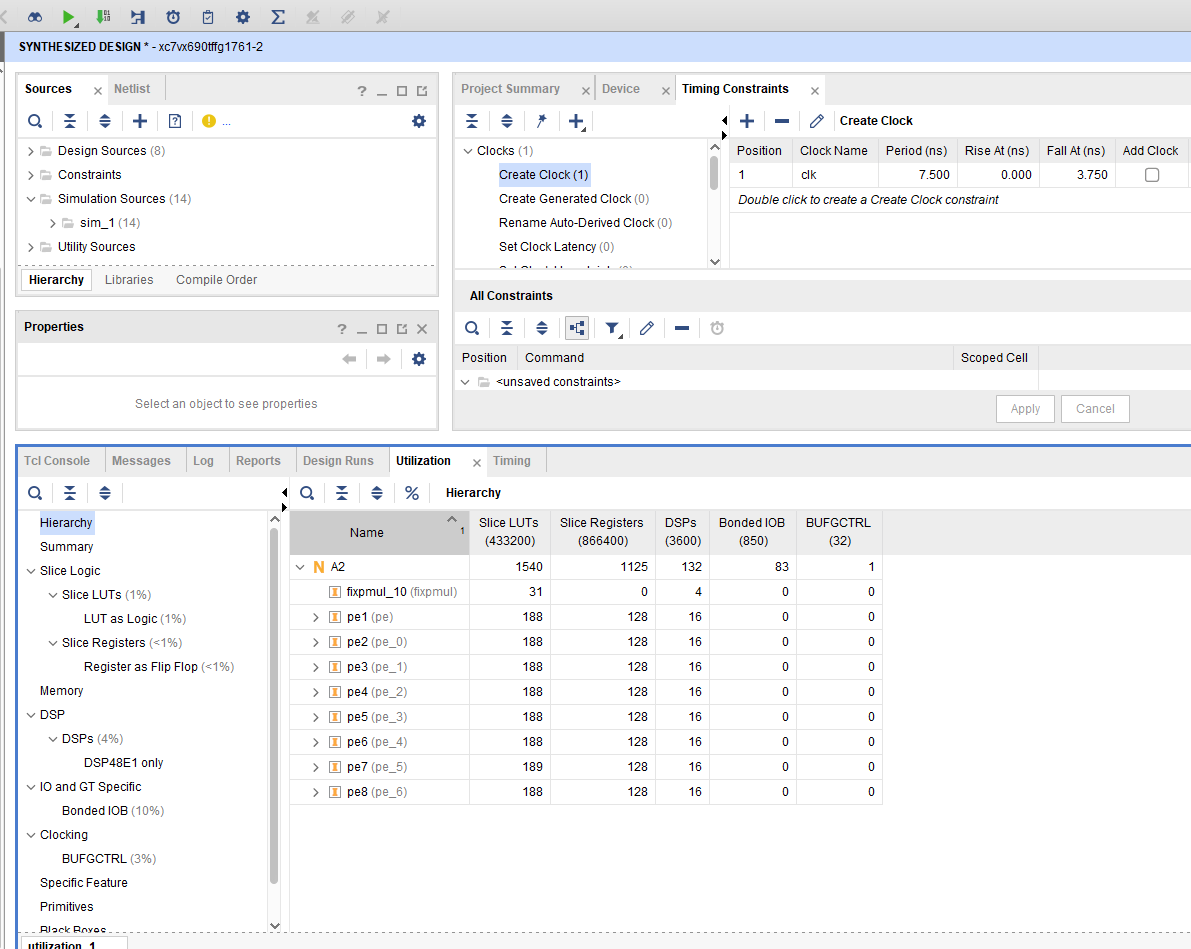
Synthesize:

频率：

可见频率为7.5ns,符合小于一半的要求。

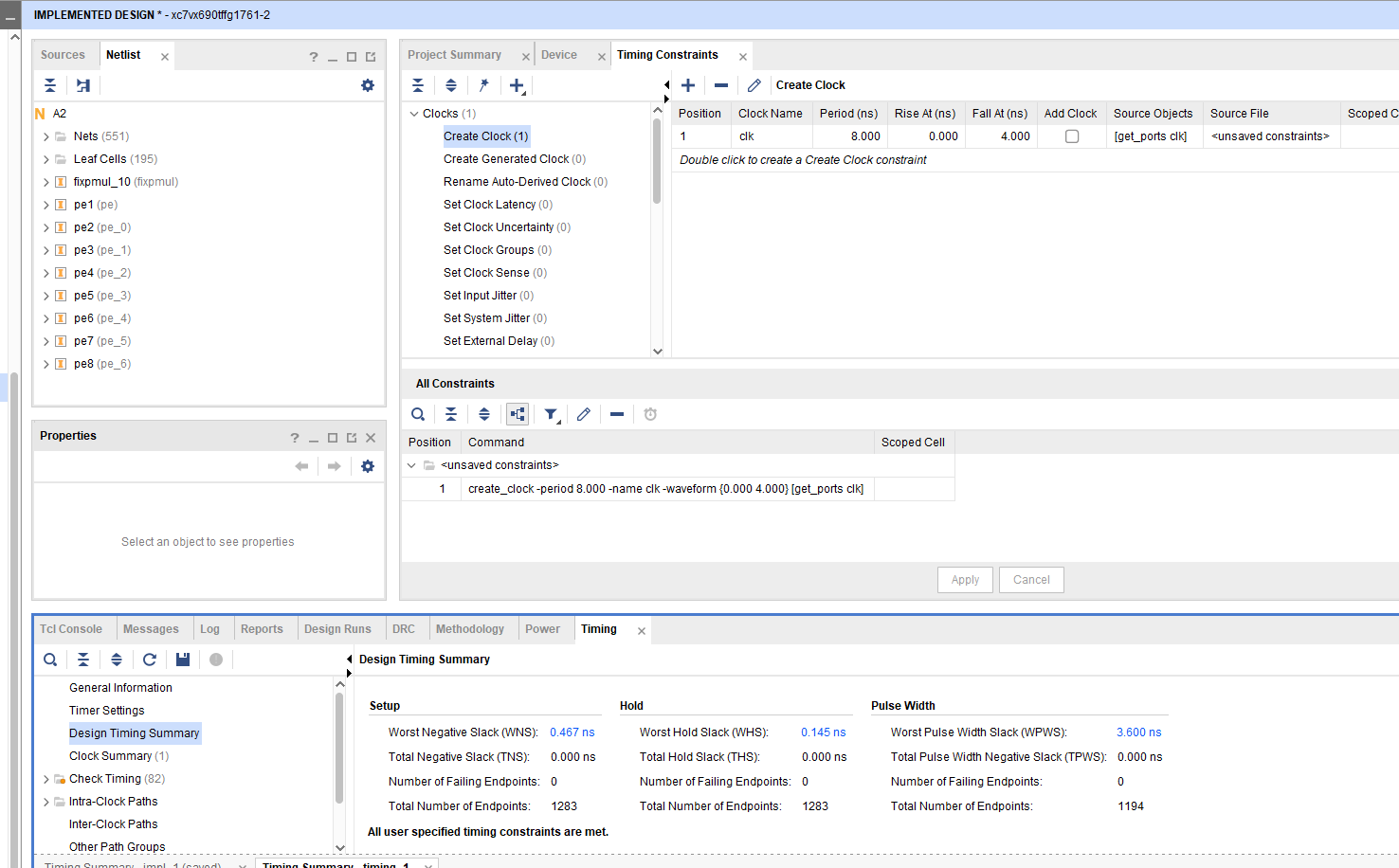


资源：

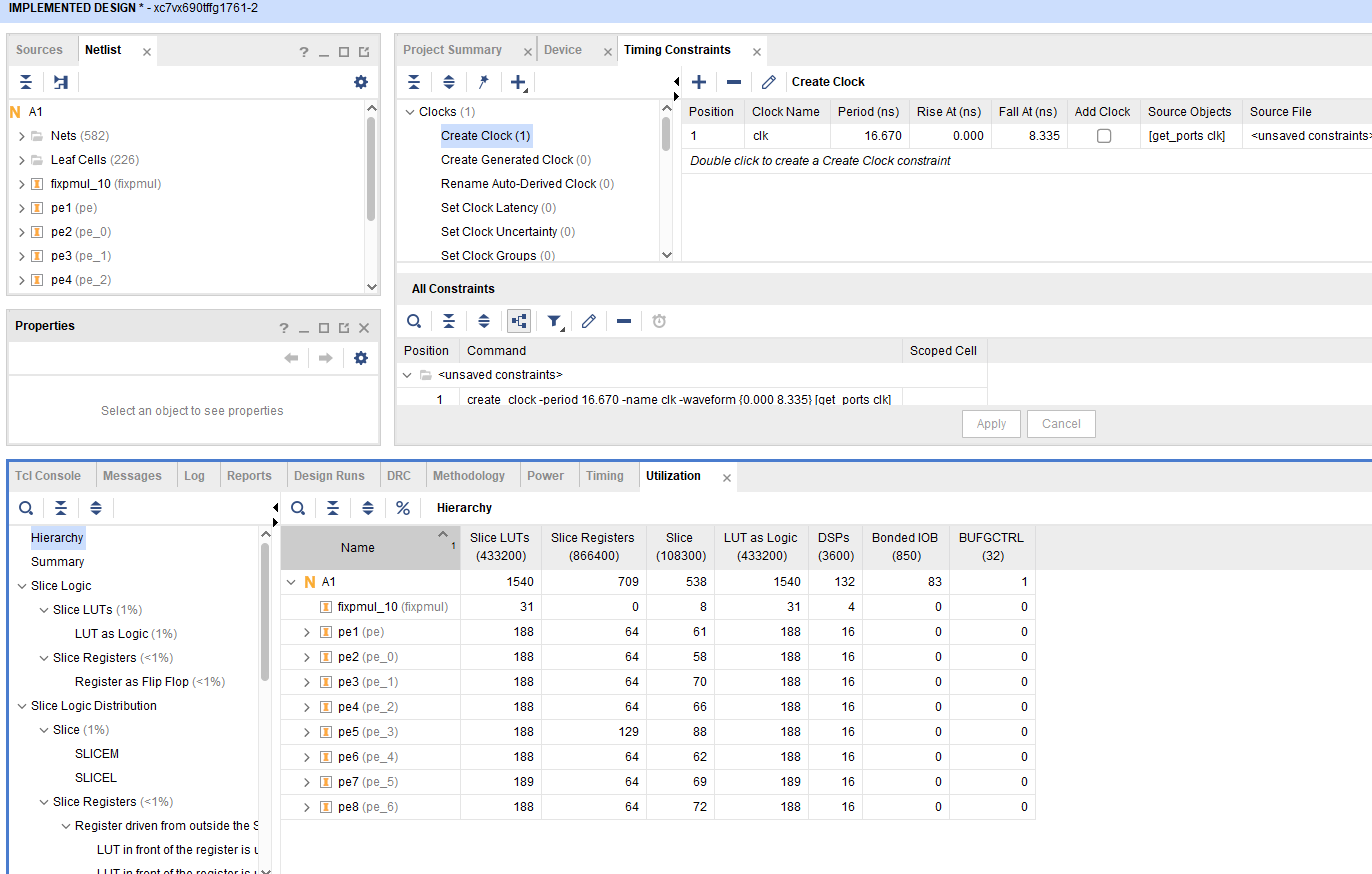


Implemente：

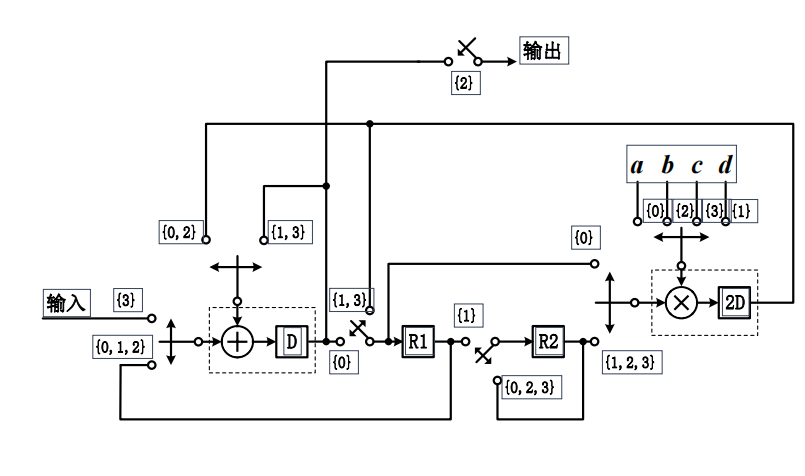
频率：可见其频率略比Synthesize的高， 猜测可能是物理实现比Synthesize多考虑了一些细节，比如连线的长度影响延迟。



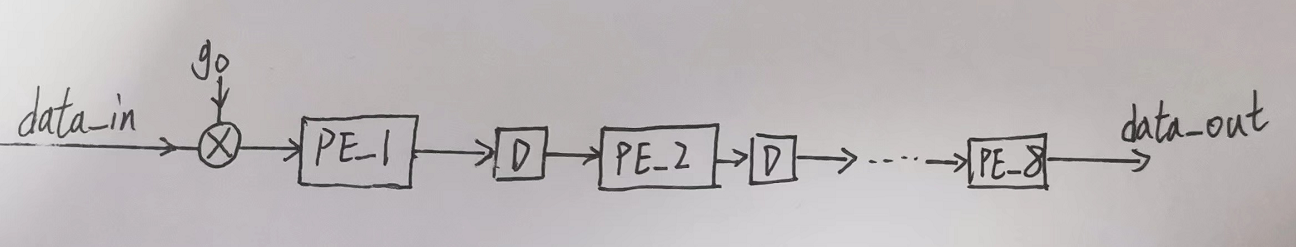
资源：



A3架构：

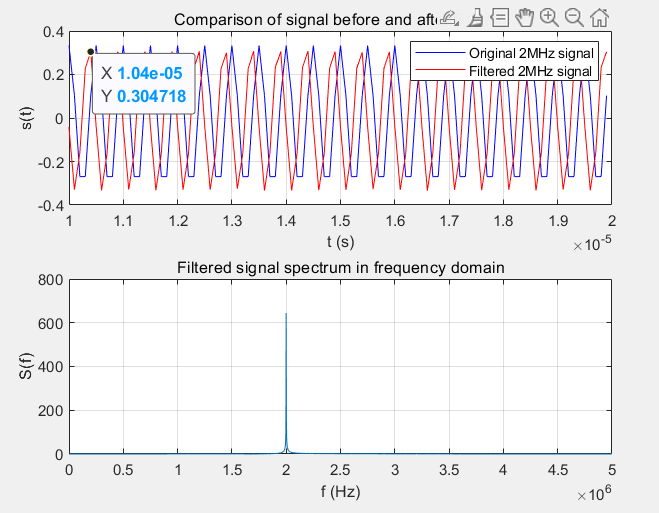


A3\_pe:



A3\_pe是对A1\_pe进行折叠后得到的结果，在相邻pe之间存在寄存器，来优化时钟周期，同时使结构满足时序。

将Rtl代码得到的结果作为输入，用matlab显示波形，结果为：



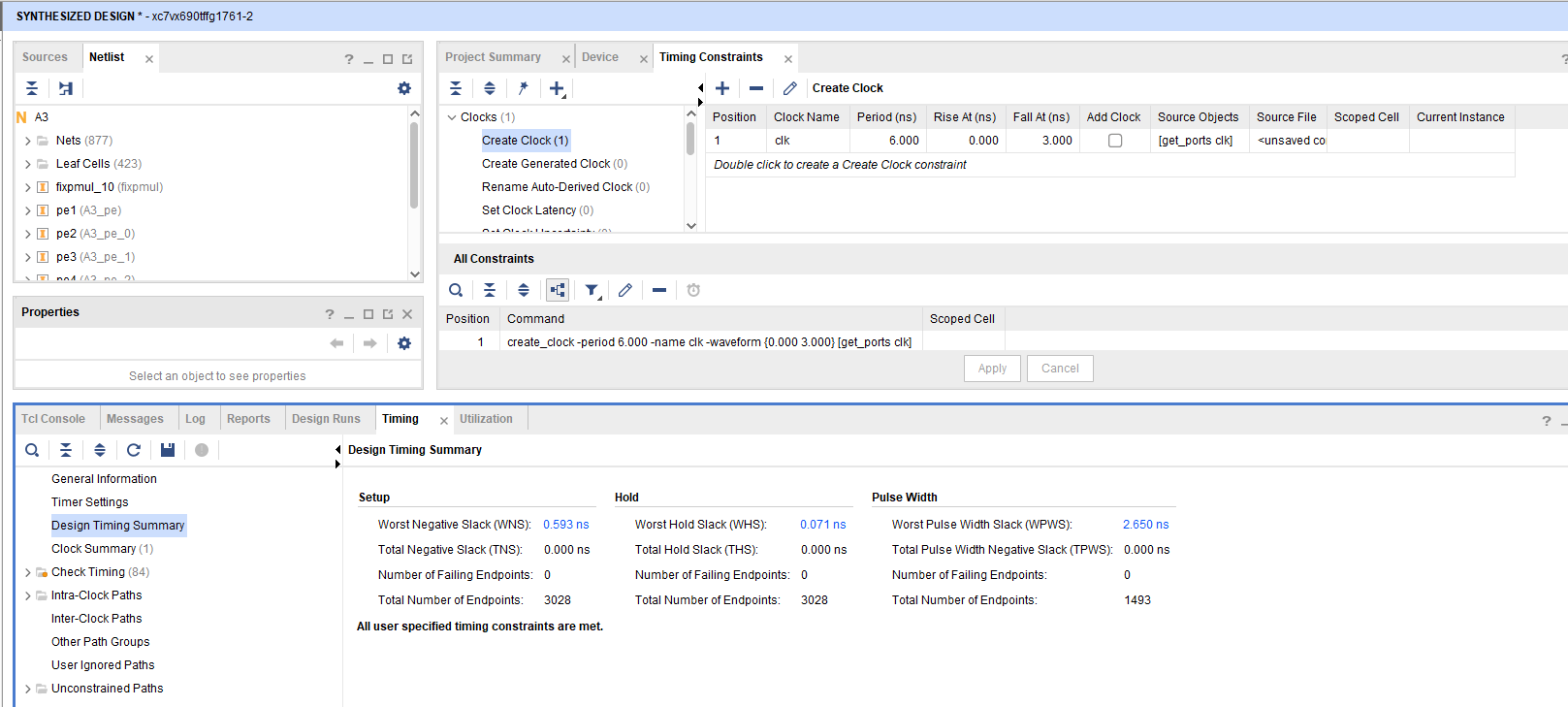
可以看到和初始用matlab模拟的结果一致，因此verilog仿真正确

Vivado结果：

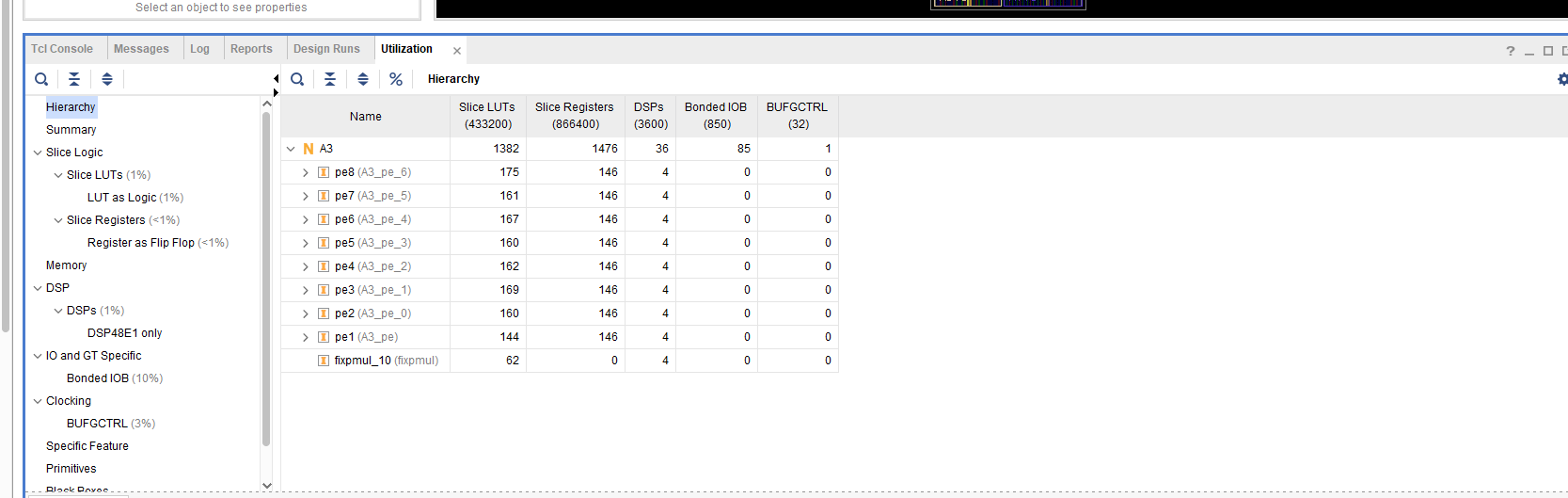
Synthesize:

频率：

可见其甚至比提高频率的A2架构更快。

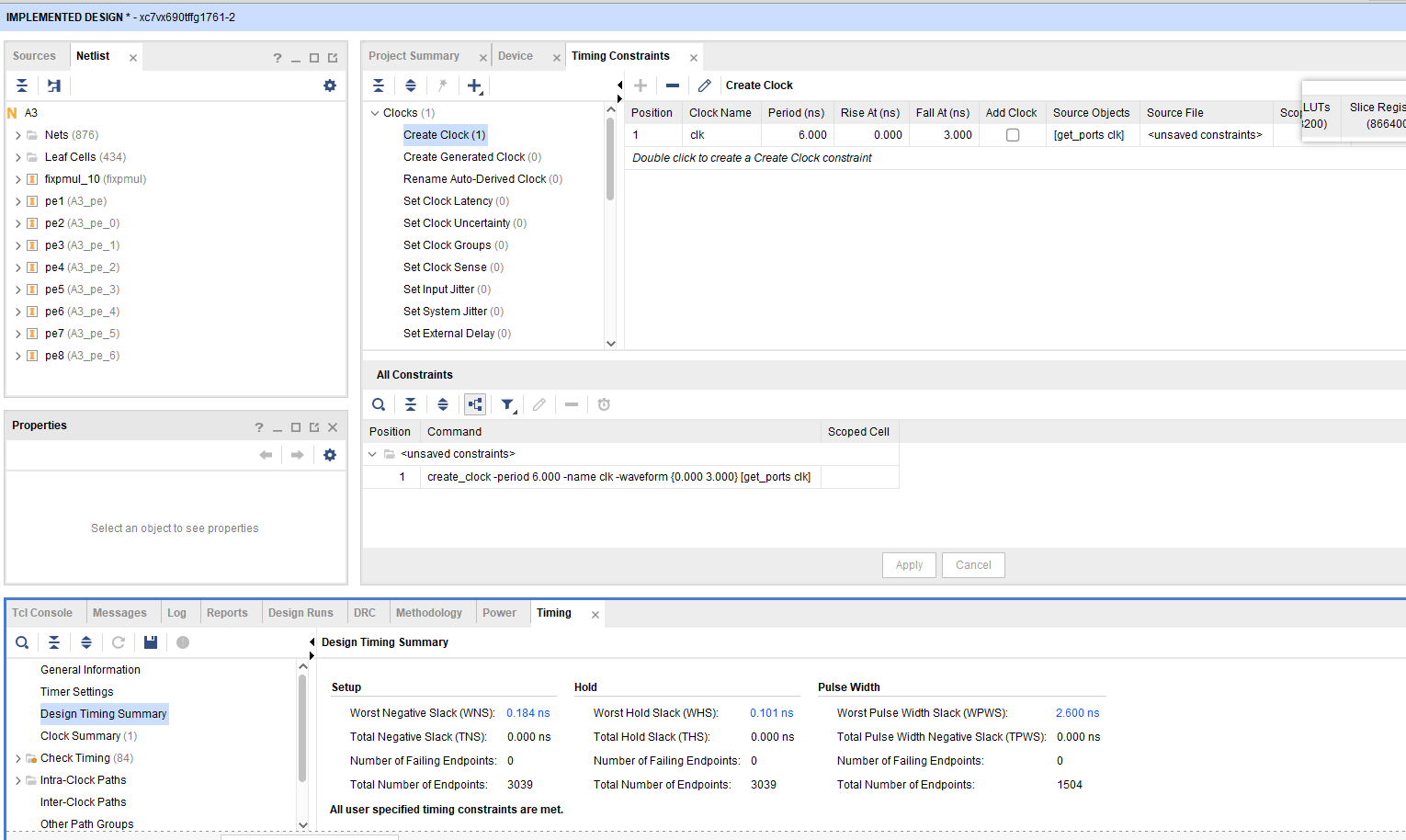


资源：可以看到其DSPs一栏pe从原始架构中的16减小到了4,这正好验证了使用折叠架构从4个加法器和四个乘法器减小到分别各一个，变为1/4，满足要求。

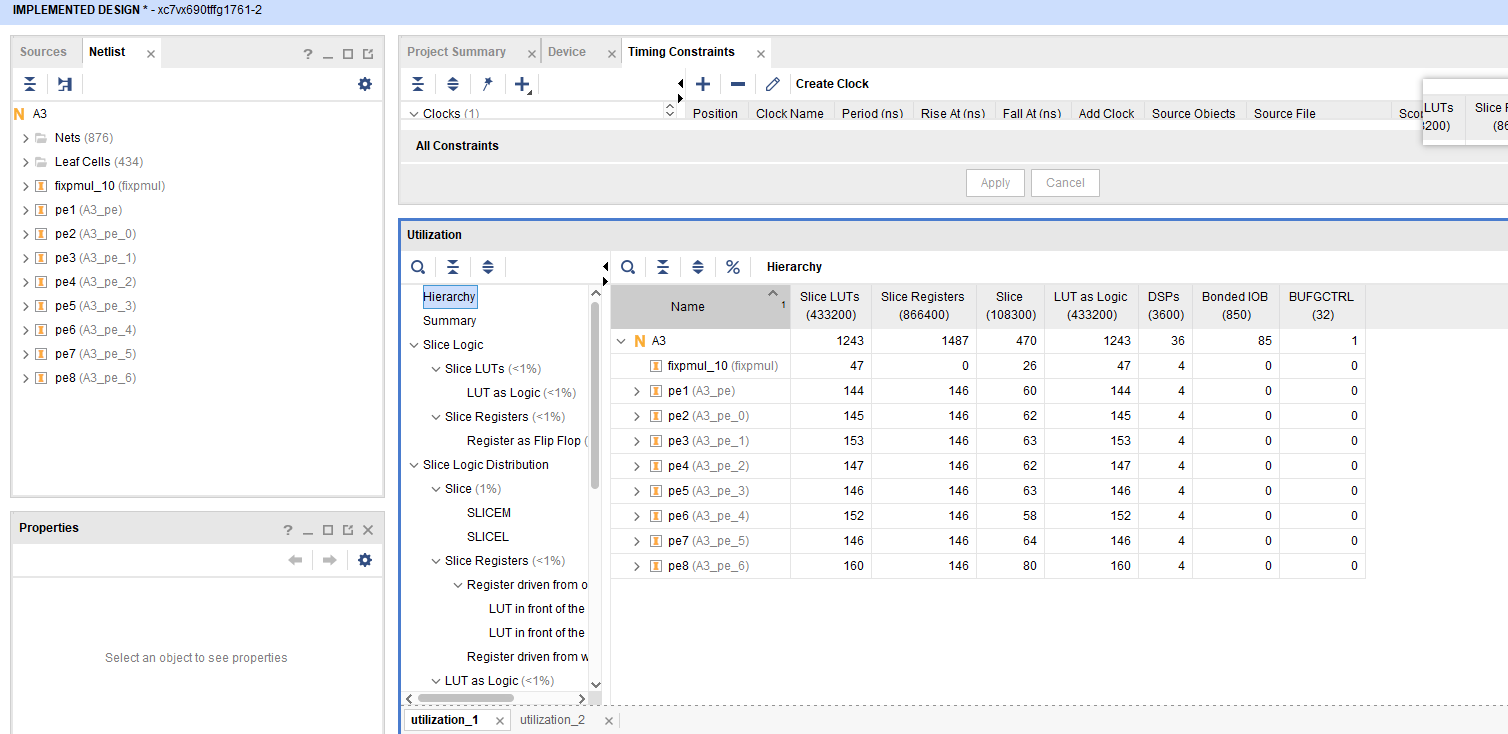


Implemente：

频率：



资源：



分工：

Matlab的验证和分析：

贾鑫鹏

数据预处理：吴非

主要为输入数据和输出数据在十进制，二进制，十六进制之间的转换（python）

A1架构的实现：

贾鑫鹏实现了主体计算逻辑，吴非进行了其中不可综合部分和使能信号控制等小细节的修改。

A2：吴非

A3：贾鑫鹏

Vivado的验证：吴非

总结：

本实验通过对IIR架构的matlab和verilog的探索实现，对IIR有了更深刻的了解，并对VLSI的并行，折叠，流水线等常用的手段进行了实践，验证了理论知识，加深了印象，同时也熟悉了matlab和vivado等常用软件的操作。

（实验过程中的小结论：Input 是reg还是wire类型不影响vivado时序报告结果。）