# 数字集成电路课程设计报告



课程设计名称：Motion Estimation Circuit Design

姓 名：吴非

学 号：519021910924

小 组 成 员：吴非，贾鑫鹏，王玉麟，汪嘉航

提 交 日 期：

# 目录

1. 设计规范简介
2. 电路性能分析与电路结构设计
3. 电路RTL模型与仿真验证
4. 电路逻辑综合策略与综合结果
5. 电路物理实现与结果分析
6. 任务分工与设计总结

# 课程设计规范简介

对所要实现电路的设计规范进行描述，包括功能、性能等要求，可参考课程设计的规范文件。

1) 每秒 60 帧 4K 视频（3840×2160@60fps）的实时处理能力

2) 采用全搜索 ME 算法、支持 8×8 块大小的 SAD 计算、搜索区间为[-7,8]

3) 芯片设计工艺：华力 55nm 工艺

4) 评价指标：电路的实时处理能力、芯片的 PPA（Performance or frequency, Power,

Area）、输入/输出数据的带宽及其利用效率

理论压缩率：

原图片bit数：3840\*2160\*8bits

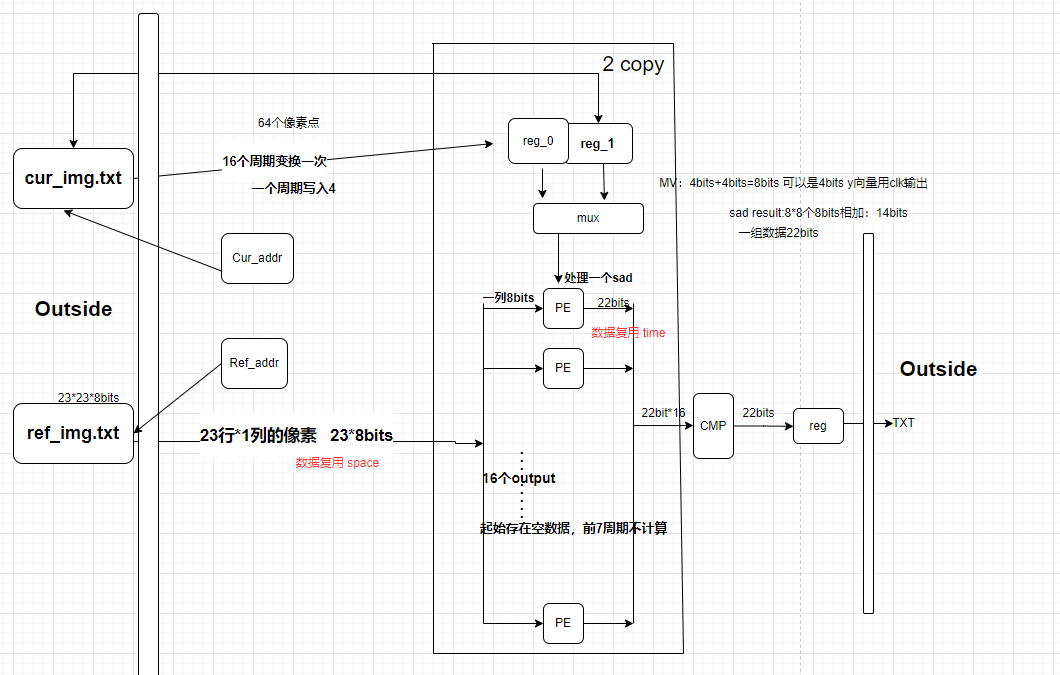
压缩后 3840\*2160/8/8\*8bits

显然压缩比率为1/64

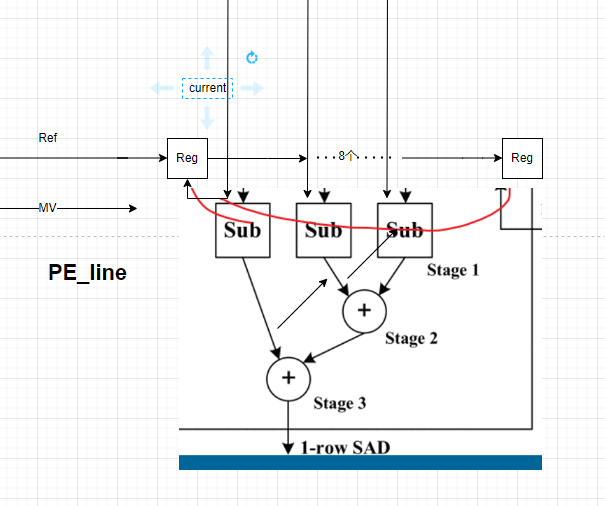
# 电路性能分析与电路结构设计

1. 对于ME、SHA256和NoC、RSIC-V等电路，根据电路设计规范，分析电路的性能、吞吐率、（峰值）带宽、平均带宽和（实时）工作频率等指标，给出必要的计算公式和估算依据。在此基础上，给出基于模块化划分的电路硬件结构图，并进行说明。

整体架构：



如图，本架构的主要思想是每个周期计算一列16个sad块，然后更新参考帧数据，具体在pe\_line图中的8个reg。



虽然是参考帧的数据在不断流动，但是我们可以想像，这就好比当前块在搜索区域内滑动，这和卷积操作很相似，当前块为卷积核，不过不同之处在于我们一次计算一列内的所有值。

由于使用一套计算模块，当我们计算下一个当前块的时候会出现参考帧的数据接不上的情况，但是隔一个当前块刚好接上，但是由于需要，不能间隔输出，因此我们通过并行技术复制一份一样的模块，但是这样又会让性能过剩，为了减小发热，以及提高寿命，在一个周期停顿不计算，一个周期计算，这样性能不会改变。由于上述分析它和初始模块差了16个周期，因此也需要另一个cur\_reg存储下一个当前块。

具体每个小模块:

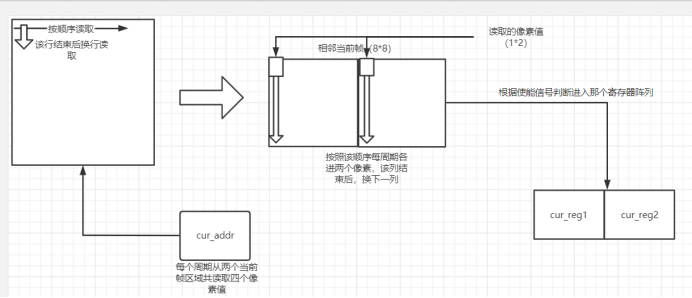
cur\_addr和ref\_addr产生需要的数据地址。

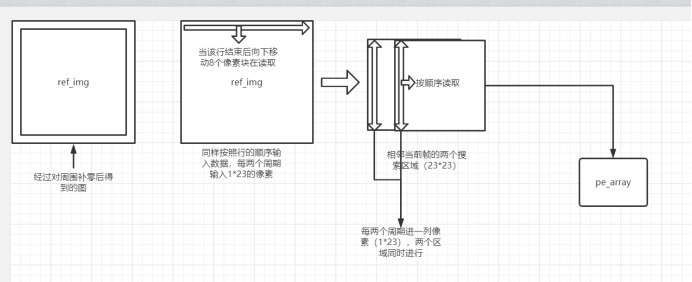
cur\_reg: 存储当前块数据，有相对于当前sad二倍大小的空间（为了切换当前块的时候不停顿）。

pe,pe\_line: 核心计算模块（同时也存储了参考帧数据）。

cmp: 比较一个周期来的16个sad值，选出最小的，并和全局的最小值比较，16个周期后输出当前sad的最小值。

主要数据流动示意：



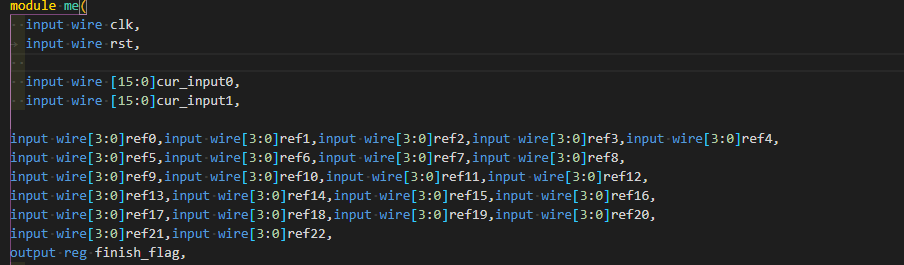


电路指标：

性能：

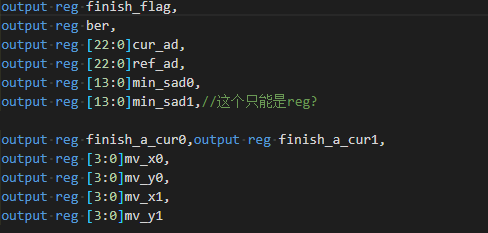
1/60秒内需要计算3840\*2160/8/8=129600个当前块的数据，一个当前块需要16个周期，200Mhz带入计算出需要0.010368s计算一张图片，计算60张需要0.622s小于1s，1s内能计算约96张图片。

带宽：



由图可见，输入带宽：2+32+4\*23=126bits

输出带宽：



2+46+28+2+16=94bits

利用效率：

输入：参考帧数据一直输入，当前帧的数据也只是在换行的时候停顿32个周期，而一行内一共有3840个周期，这很微小，因此可以近似为100%

输出：

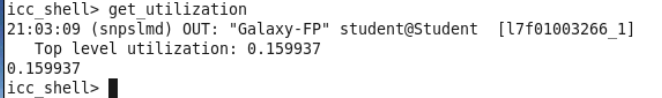
每16个周期输出一次，因此效率为1/16,当然也可以进行改进为近100%，每个周期输出所有数据的16份中的一份，让后续的模块整合16个周期的数据为完整的数据即可。

PPA:

功耗：51.2346mW

面积：芯片面积：1646578um^3

面积比例：



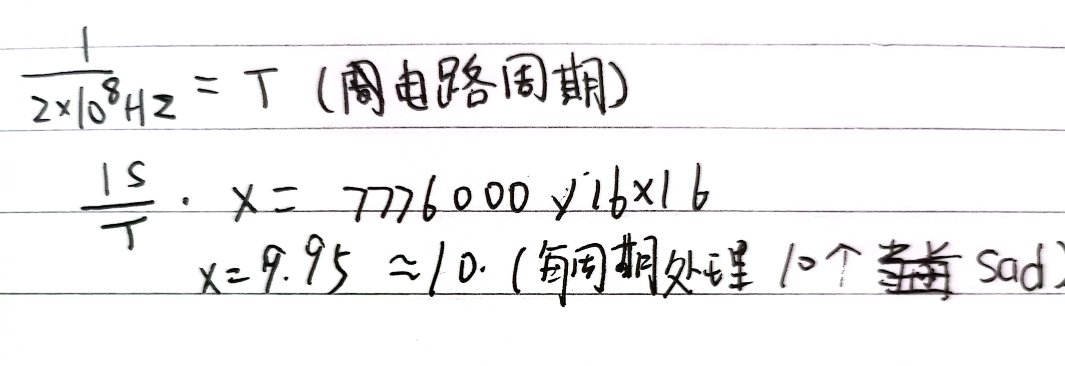
面积利用率低，是使用pad数量多的缘故。

1. 对于加法器电路，根据电路设计规范，分析电路的性能瓶颈，给出必要的估算依据，给出提升性能的方法和途径。在此基础上，给出基于模块化划分的电路硬件结构图，并进行说明。

理论分析：

1s内一共要处理的数据量：1s\* 60 \*3840\*2160\*8bit

=3981312000bit，一次sad计算处理8\*8\*8bit=512bit，因此1s内一共要处理7776000个块



由此得到最少一个周期需要计算10个sad值，为了避免电路过于复杂，我们直接采取一个周期计算一列的sad值，也就是16个sad值，这也就意味着性能会超出大约60%.

同时根据全加器的延迟为0.17ns，频率为200mhz

1\*10^9/(2\*10^8\*0.17)= 29.5级，因此全加器一个周期可以有29.5级

而算一个8bit的加法，需要8,4,2,1共4级，一行8个点，加起来需要3级，

一个sad8行，需要三级，一共16个sad相互比较，需要4级，一共需要14级远小于29.5，满足要求。同时为了能够让频率更快，以及更好的区分计算sad值和比较sad值的功能，将其分成pe和cmp两个功能模块，两模块之间用reg分隔做流水线。

性能瓶颈：

根据上述的全加器级数，显然应该是数据从pe进到出pe的路径为关键路径，这是电路的性能瓶颈。

提升性能方法：

-增加流水线级数

-并行

-采用更先进的架构（改进电路一个周期停顿一个周期计算的缺点）。

1. 介绍自己在上述部分中的主要工作；

查找论文等相关资料，设计了架构并画出架构图。

# 电路RTL模型与仿真验证

1. 简要描述电路的RTL模型设计，重点介绍电路的验证方法、验证平台的搭建、和验证结果；

数据预处理：

电路整体以当前帧和搜索区域的一列为整体进行输入，所以在选择输入之前，需要对图像进行预处理，对于先前帧而言，需要对其周围[-7,8]范围内进行补零操作，在补零完成后以列的顺序存进寄存器中，而对于当前帧，为了方便对所需块的读取，按照1\*8的大小从左向右进行读入，即每次存储8个像素点，向右读取，直至本行读完，然后进行换行操作。（该部分操作由脚本完成）

代码：

1. cur\_addr.v和ref\_addr.v：

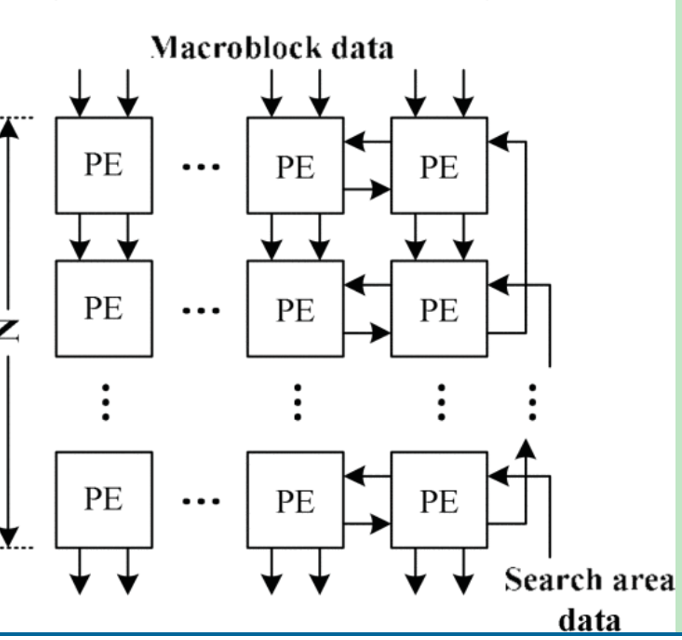
根据电路架构功能输出地址给me\_tb.v，me\_tb.v返回数据给芯片。

1. cur\_reg.v：

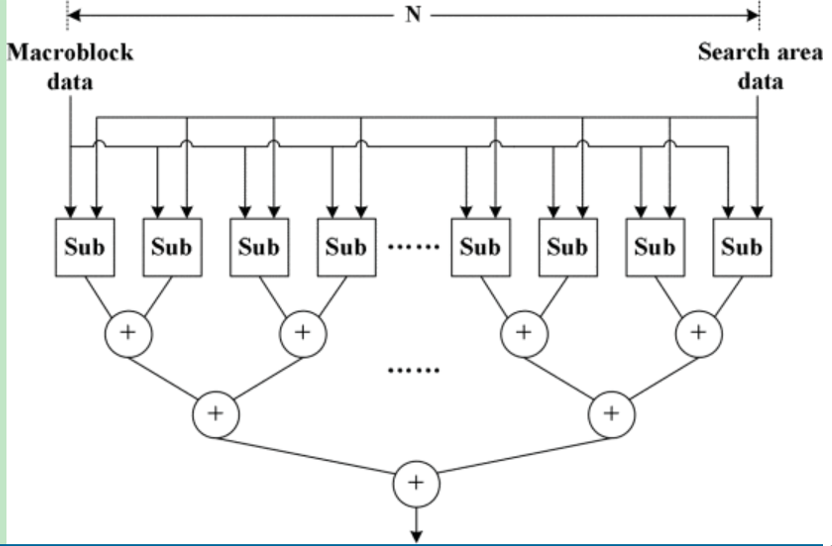
由于当前块数据需要保存32个周期不变，为了减小io数量，提高性能，使用了两个cur\_reg模块来存储当前块数据，之所以用两个是因为一个供给计算模块计算，另一个同时更新下一个当前块的数据，避免停顿问题。

1. pe\_array,v,pe.v,pe\_line.v如二板块所描述，电路核心计算模块，采用混合架构（收缩阵列架构和树结构混合）。

收缩阵列架构：



1. 树架构：



1. cmp.v：

在计算16个sad值的同时产生移动向量mv\_x和mv\_y

1. delay.v:

当前块数据延迟，使能信号延迟

1. output\_tb.v:

输出结果到文件

验证结果：

结果和参考结果对比，正确。

仿真性能：

由之前的性能分析，计算60张需要0.622s小于1s，符合要求。

1. 验证结果包括：功能点的验证、测试向量的多少、仿真性能的统计；

在进行每个小模块的编写时设计了各个模块单独的testbench。

1. 介绍自己在上述部分中的主要工作；

pe,pe\_line基本计算模块，cmp比较器，输出到文件以及延迟模块(包括相关模块的testbench)

和贾鑫鹏同学合作，他主要负责数据预处理，一同完成RTL代码的编写。

最后的整合和测试，以及后续逻辑，物理综合不符合后的修改。

# 电路逻辑综合策略与综合结果

1. 给出电路简要的逻辑综合流程、参数设置以及综合结果；
2. 综合结果包括：时序结果、面积结果、功耗结果、资源使用结果和设计违例，附上相关的结果；
3. 介绍自己在上述部分中的主要工作；

1.逻辑综合流程

第一步：设置lib以及search path

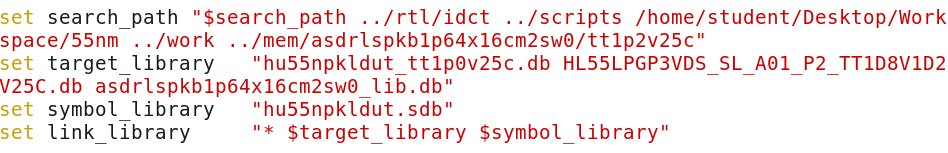


图1

第二步：顶层文件的书写与例化，读取RTL



图2

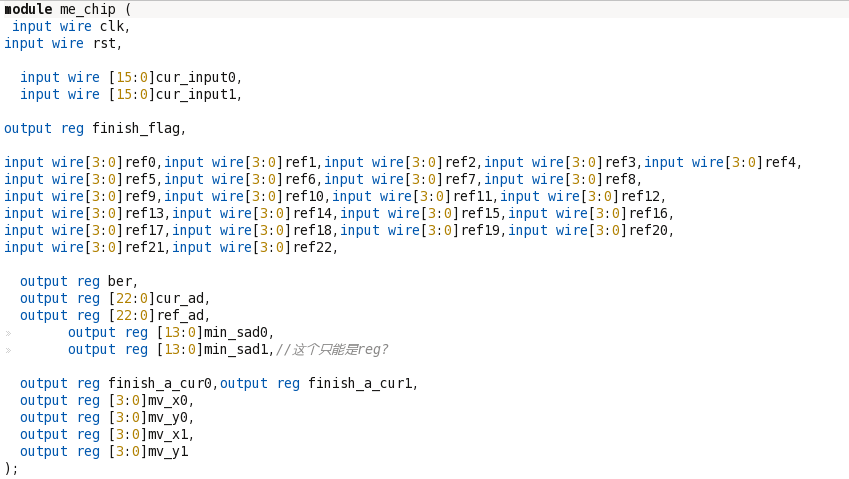
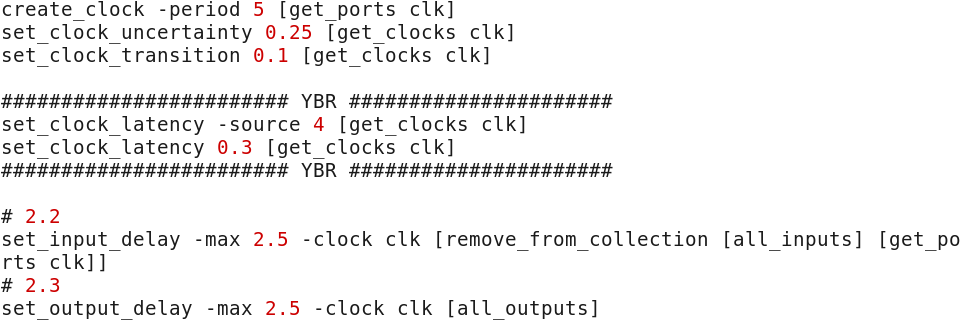


图3.me\_chip顶层文件

共包括230个pad，其中136个为输入pad，94个为输出pad。

第三步：设置约束

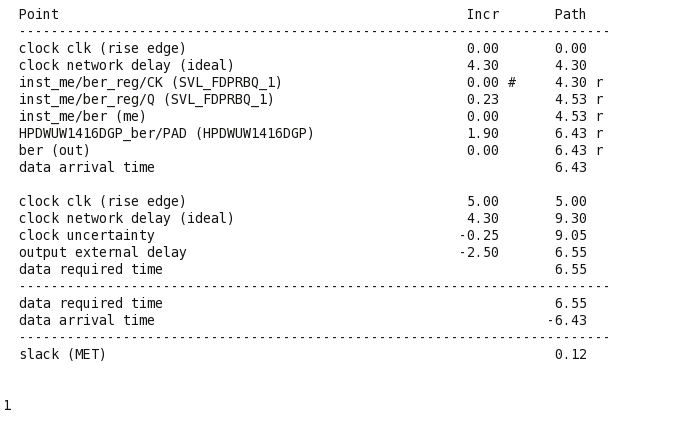
时钟周期为5ns，其他参数按照周期等比例缩放。



第四步：compile

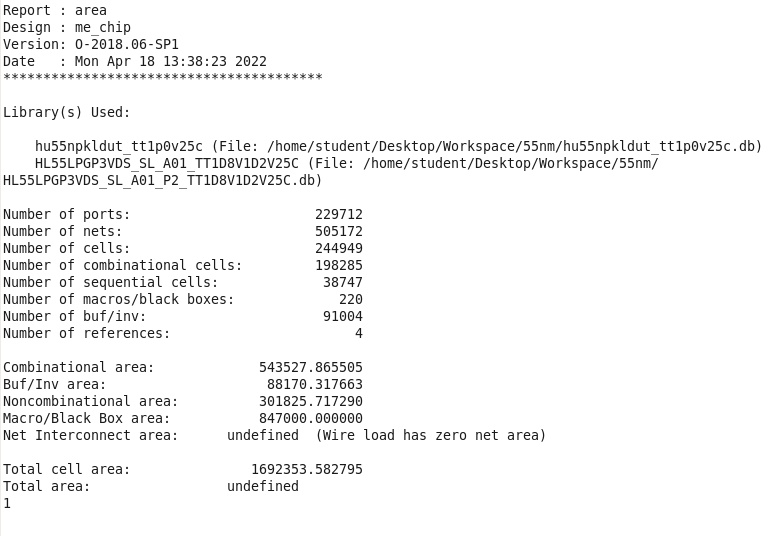
1. 逻辑综合结果

1)时序结果



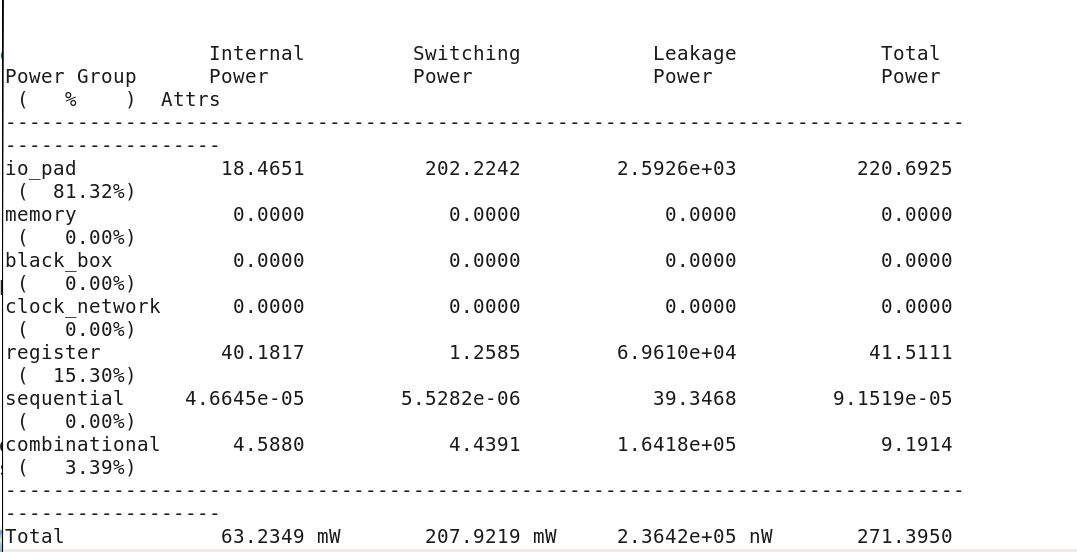
1. 面积及资源使用结果

总面积为1692353平方微米，详细资源使用结果见下图。



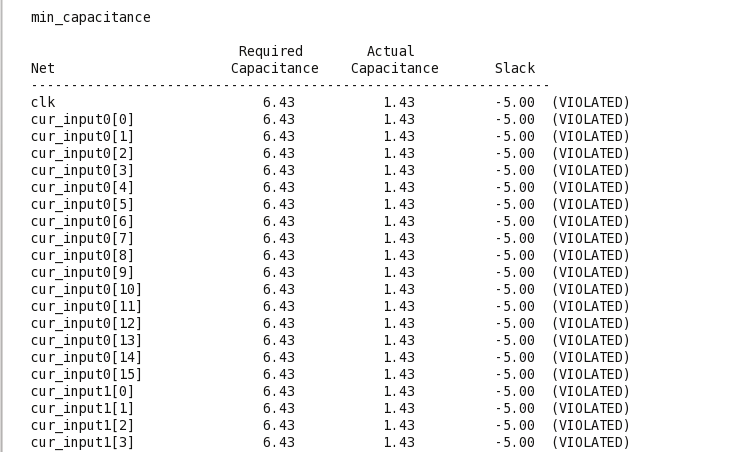
1. 功耗结果

除去io\_pad的功耗，芯片的功耗为50.7025mW



1. 设计违例

最小电容违例



（3）

这部分由汪嘉航同学负责，我涉及的主要是不符合要求后修改RTL代码以及督促汪同学完成任务。

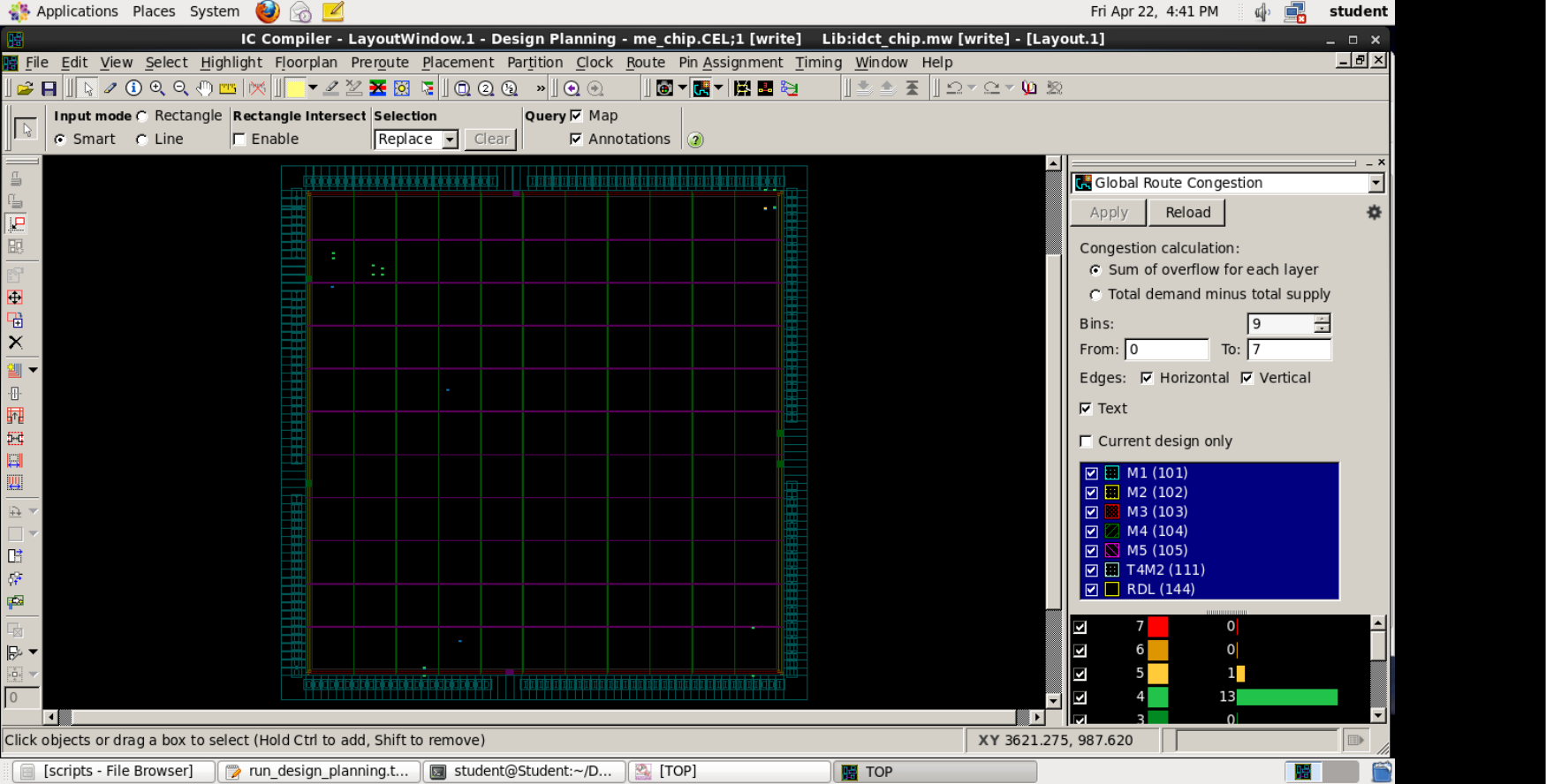
# 电路物理实现与结果分析

1. 给出利用ICC设计电路的各步骤结果；
2. 步骤包括：局部规划、布局、时钟树综合和布线；
3. 结果包括：完成布局规划的芯片和完成布局布线的芯片概貌图、时序结果、面积结果、功耗结果、芯片的压降、芯片的拥塞分析、资源使用结果和设计违例；
4. 介绍自己在上述部分中的主要工作；

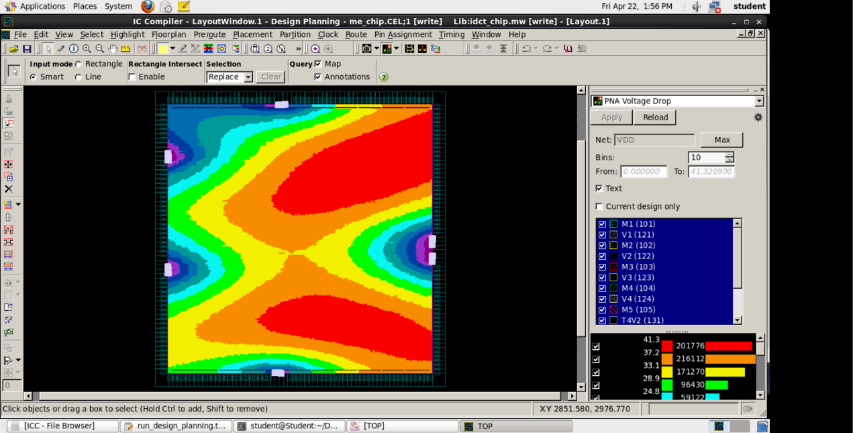
ICC设计电路包括，版图规划，布局，时钟树综合，布线

一下是各步骤结果；

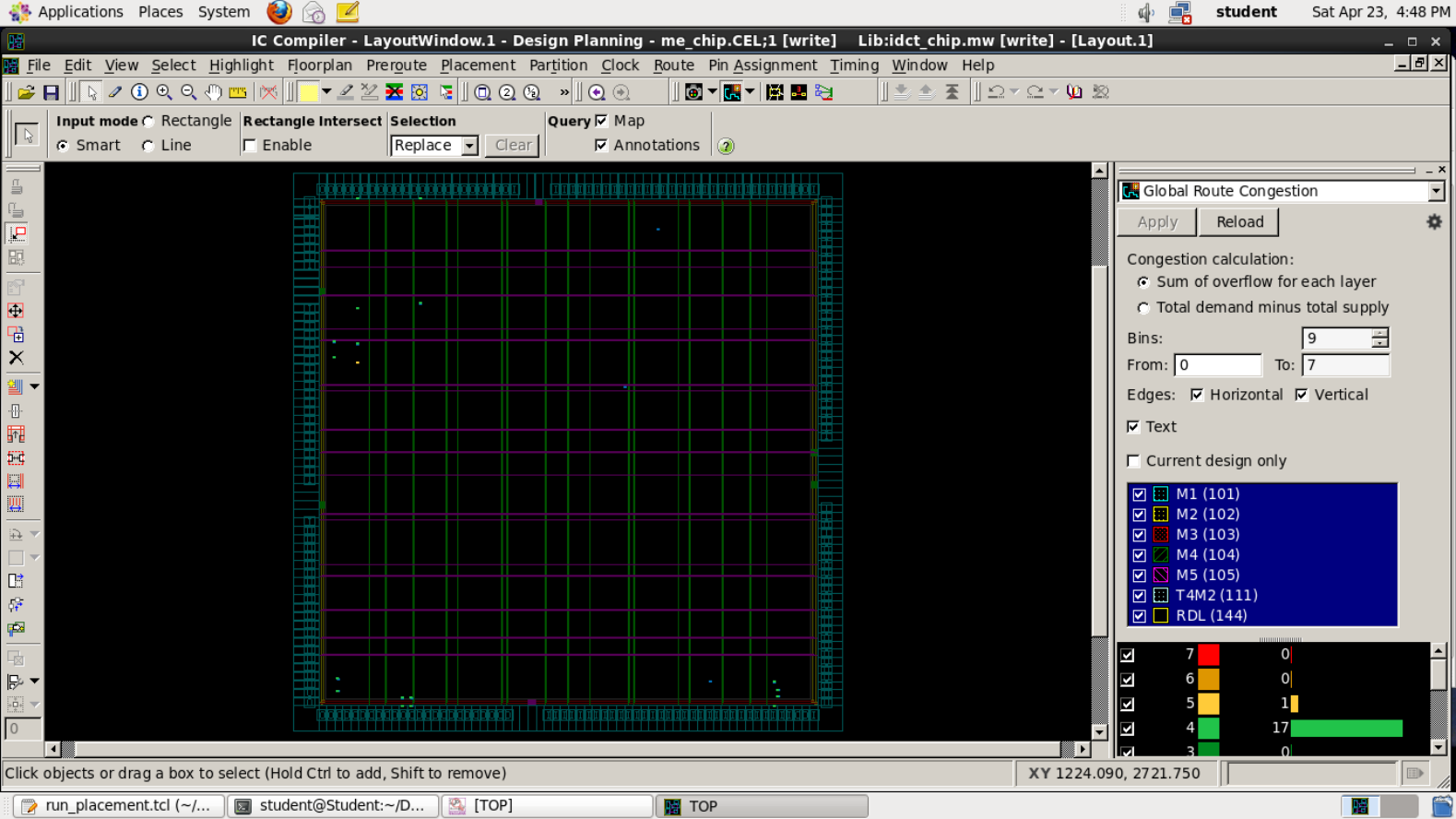
版图规划：芯片概貌图



压降

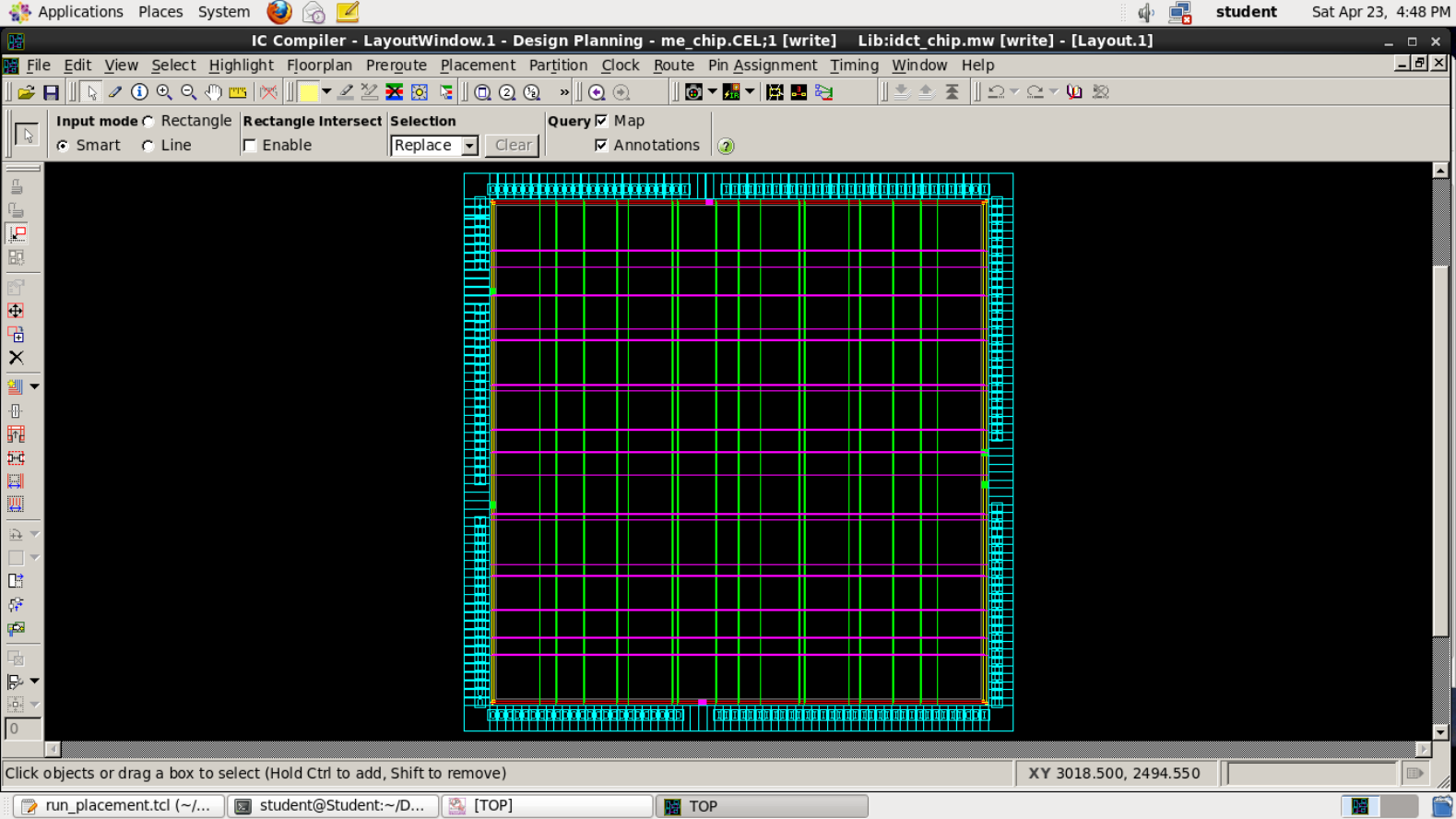


拥塞

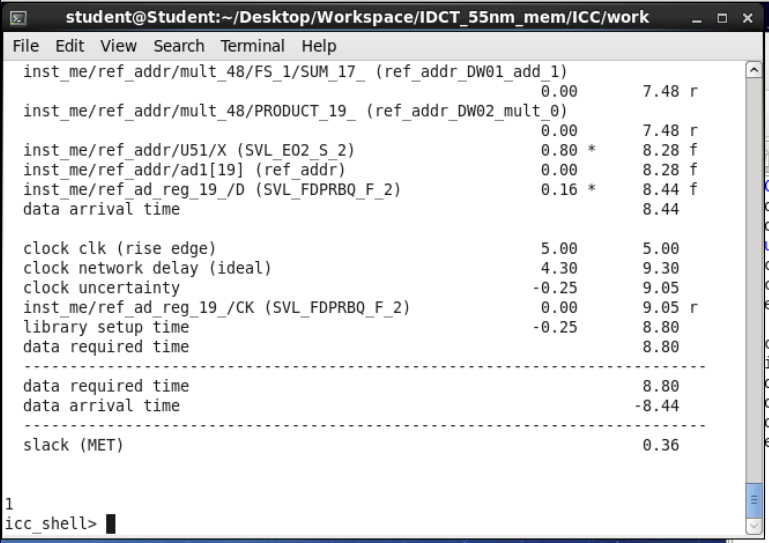


Placement

芯片概貌图

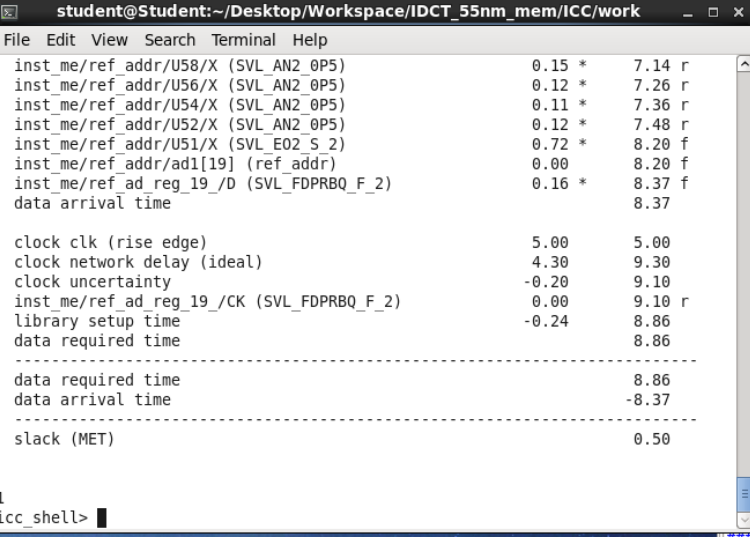


时序结果



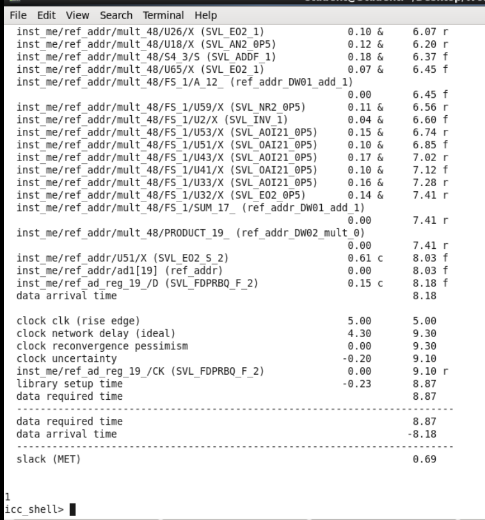
时钟树综合

时序结果

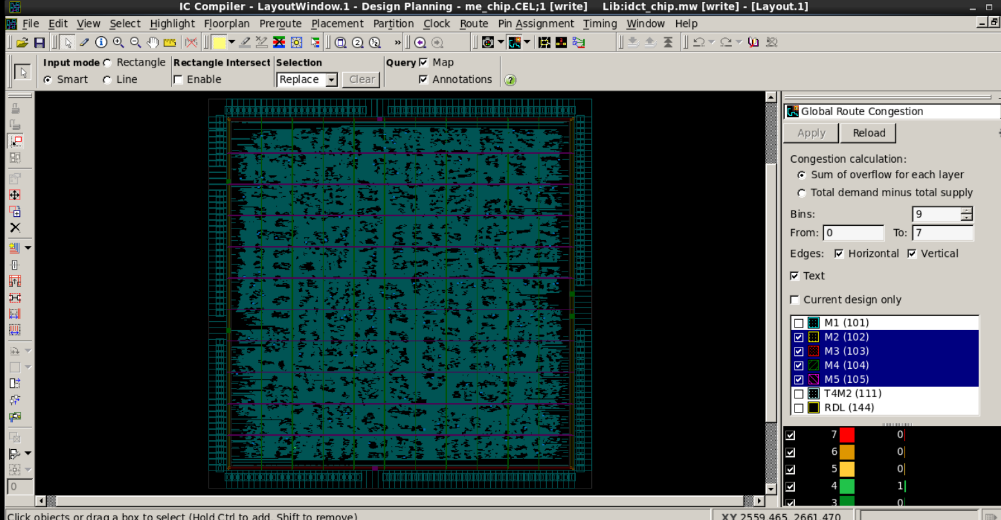


布线

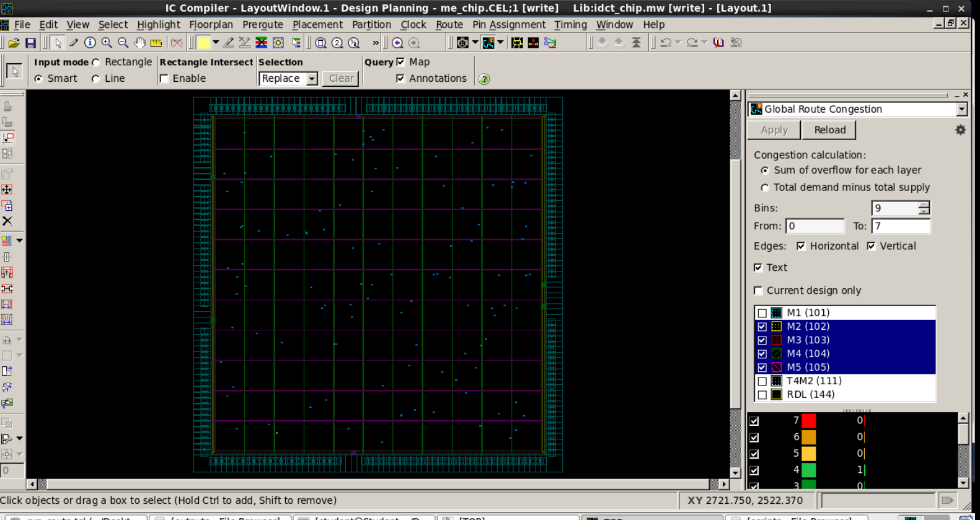
时序结果



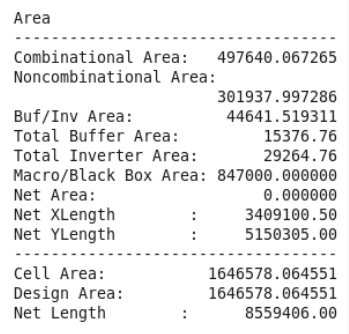
Finish



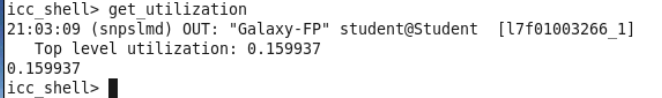
拥塞



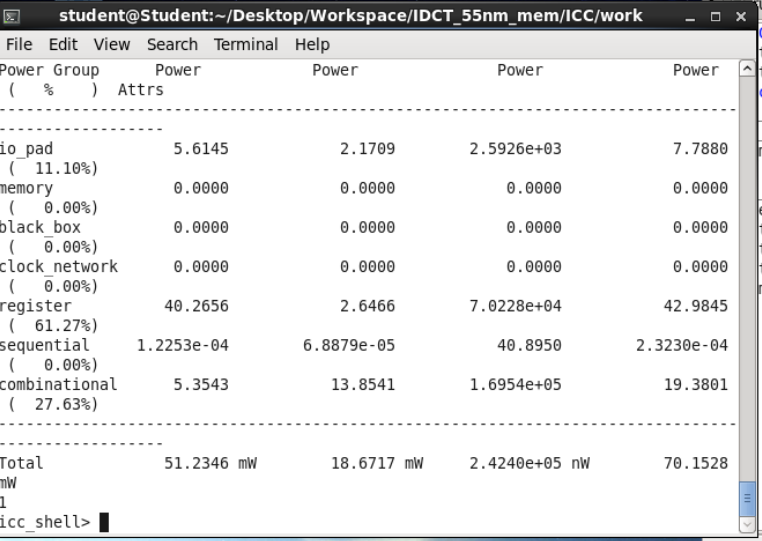
面积结果



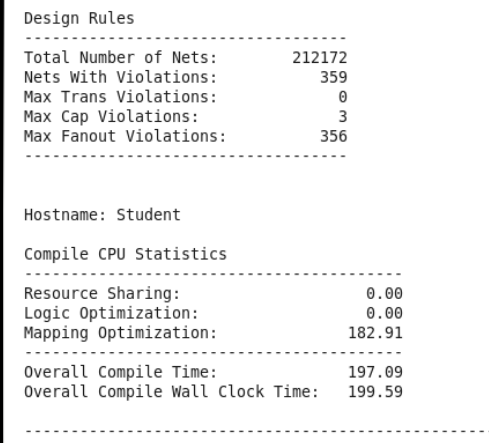
面积比例：



功耗结果



资源使用结果



从上述结果可以看到，我们的ICC设计，总体来看效率很高，使用的IO，面积，资源使用结果较多，在最初阶段存在一定的拥塞，但在经过优化后，拥塞得到了完美解决。压降最大为40mV<50mV，时序也均满足，符合要求。

（4）

这部分由王玉麟同学负责，我涉及的主要是不符合要求后修改RTL代码以及督促王同学完成任务。

# 任务分工与设计总结

简要说明课程设计中各小组成员的任务分工，课程设计过程中遇到的主要问题，解决思路及其它问题。

吴非：

电路架构图绘制和架构的设计

RTL代码中pe,pe\_line基本计算模块，cmp比较器，输出到文件以及延迟模块的编写(包括相关模块的testbench)

代码最后的整合和测试，以及后续逻辑，物理综合不符合后的修改。

问题：

Verilog代码基本语法的熟悉:  
 问助教，上网查阅，同学讨论，编写testbench验证想法是否正确。

电路架构有功能bug:

查阅资料，小组讨论修改架构和RTL代码。

代码调试：

换行后32个周期的停顿需要很准确的控制各个使能信号，通过查看波形调试，这部分所占用的时间甚至超过代码编写的时间。

前期讨论和进展不顺利：

定期督促和开会讨论。

贾鑫鹏：

参与架构设计的讨论，对数据的预处理。

RTL代码中，地址产生模块cur\_addr,ref\_addr, 与当前帧 存储模块cur\_reg的编写与调试。

后期代码运行中结果存在bug，参与调试与解决。

问题：

Verilog代码熟悉程度不高。

查资料熟悉代码内容，并对代码进行修改。

数据预处理

通过cpp脚本进行处理，在过程中出现了多次错误的处理 最终通过与原数据的多次对比，得到正确结果。

小模块中时序问题。

通过观察输出波形进行调试，改变使能信号与判断信号达到要求。

汪嘉杭：

参与电路架构的讨论

前期RTL代码中的mux和decoder，以及testbench

pe模块例化文件pe\_chip和顶层文件的例化me\_chip的编写

逻辑综合

问题及解决思路

时序不满足要求

通过分析时序报告来发现优化策略

王玉麟：

参与电路架构的讨论

RTL代码中，cmp比较器初版，输出到文件及testbench

物理综合

问题及解决思路

电路架构有功能bug:

查阅资料，小组讨论修改架构和RTL代码

输入违例

输入时先经过reg再输入

电压压降，

先定义全部pad位置，调整电源pad位置

时序

调整pad位置，使布线距离更短

电路skew较大

优化时，设置调整参数

绕线偏差

实际placement存在误差，每次布局优化后，都需route\_zrt\_global重新评估