1 实验过程

运行环境：ubuntu 20.04 LTS 腾讯云轻量云服务器2核4G

工艺库：sky130

运行设计：gcd　uart　aes_cipher_top　picorv32

运行的中间log文件以及teminal记录都在打包文件中。

在实验开始时，我在自己的本地电脑的ubuntu 20.04系统上运行后端iFlow，遇到了各种问题（后端脚本的问题、下载网络的问题、ubuntu系统自身存在的不兼容等问题），带来了很大的困扰，花费了很多时间。出现了如以下问题：

错误信息：CMake Error at /usr/share/cmake-3.18/ Modules/ FindPackageHandleStandardArgs. cmake:165 (message): Could NOT find SWIG (missing: 3.0) (found version "4.0.2")

解决措施：遵从班级群同学发现的bug解决方案，将`src` `OpenDB` `OpenSTA`里的三个Cmake文件中的`findpackage(SWIG REQUIRED 3.0)`语句改成`findpackage(SWIG 3.0 REQUIRED)`后，顺利运行。

错误信息： fatal error: aux.h: No such file or directory 2 | #include "aux.h" | compilation terminated.

在我自己安装的Ubuntu 20.04和Windows双系统的联想拯救者R9000p机器上又出现了一系列的报错，让人崩溃，并且所安装的Ubuntu系统自身也存在一些bug，在花费到尝试到深夜之后，感觉到棘手，会花费很多时间处理一些无关痛痒的问题，然后就下决心购买了腾讯云的服务器继续进行试验，第一次使用腾讯云的服务器，没有进行GitHub的网络配置，下载速度非常慢，达到“KB/s”，最终经过一段时间的等待，顺利的完成了build任务，build过程总体上还是比较顺利的。

后来在跑iFlow流程时，在gcd的后端流程、uart的后端流程没有出现问题，但是在aes_cipher_top的后端流程、picorv32的后端流程中出现了问题，这两个设计的规模较大，运行时间过长，往往是运行一两个小时之后，突然报错，在实验了若干次之后。主要是出现下面两个问题，这是Floorplan的`CORE_AREA`、`DIE_AREA`设置不合理出现的问题。

```
[INFO GRT-0128] Max H Overflow: 2
[INFO GRT-0129] Max V Overflow: 6
[INFO GRT-0130] Max Overflow  : 6
[INFO GRT-0131] Num Overflow e: 495
[INFO GRT-0132] H   Overflow  : 332
[INFO GRT-0133] V   Overflow  : 689
[INFO GRT-0134] Final Overflow: 1021

[INFO GRT-0124] Update congestion history type 4.
[INFO GRT-0102] iteration 200, enlarge 52, costheight 1379, threshold 0 via cost
 0.
log_coef 0.76644856, healingTrigger 85 cost_step 2 L 1 cost_type 1 updatetype 4.
[INFO GRT-0126] Overflow Report:
[INFO GRT-0127] total Usage   : 414706
[INFO GRT-0128] Max H Overflow: 2
[INFO GRT-0129] Max V Overflow: 6
[INFO GRT-0130] Max Overflow  : 6
[INFO GRT-0131] Num Overflow e: 496
[INFO GRT-0132] H   Overflow  : 324
[INFO GRT-0133] V   Overflow  : 691
[INFO GRT-0134] Final Overflow: 1015

[ERROR GRT-0118] Routing congestion too high.
```

```
    elapsed time = 02:25:30, memory = 2693.82 (MB)
    completing 100% with 486907 violations
    elapsed time = 02:37:20, memory = 2674.37 (MB)
Killed
klayoutInsertLef.py : Insert lefs into lyt file of klayout
klayoutInsertLef.py : Finished
ERROR: /home/ubuntu/iFlow/scripts/common/def2gds.py:102: Unable to open file: /h
ome/ubuntu/iFlow/result/picorv32.droute.openroad_1.2.0.sky130.HS.TYP.V1/picorv32
.def (errno=2) in Layout.read
  /home/ubuntu/iFlow/scripts/common/def2gds.py:102 (class RuntimeError)
```

看到群里老师的建议，跑route之前，前面floorplan和place的结果之前，要关注这两步的利用率，最好要把利用率保持在30%左右，我在后续aes_cipher_top设计的floorplan设置时遵从了这个建议，但是也出现了overflow和kill内存的问题，后来进行了多轮控制变量法的调整，最后利用率保持在26%左右时，然后也没有出现overflow和kill内存的报错，顺利地得到了aes_cipher_top的gds文件，达到了实验文档中的要求。

在运行picorv32的后端流程中，出现了较大的问题，在droute时，优化迭代的时间一轮花费两个多小时，然后出现上述的内存被killed的红色错误，在微调floorplan的`CORE_AREA`、`DIE_AREA`后，仍然无法解决问题，把`CORE_AREA`、`DIE_AREA`面积调小了之后，就出现Overflow的问题，把面积`CORE_AREA`、`DIE_AREA`调大了就出现内存被killed的问题，进行了若干轮之后，我认为这是电脑处理器性能过低、内存不足的原因，如果后续有更加优秀的平台，或许这个问题将得到有效解决，当前或许再花费更多的时间也并不会对结果有所改变。

在本次实验中，通过实验课老师的授课以及阅读文档资料用户手册，尤其是自己在后端工具链实际上运行具体设计的过程中，我对整个后端流程有了初步的认识和了解，增加了知识面，有益于自己的学习工作。在运行iFlow的实际过程中，也发现了该开源工具链存在的一些问题，在百门级设计gcd、千门级设计uart的运行中比较顺利，但是到了万门级的aes_cipher_top以及更大规模的picorv32的运行上，其优化算法可能效率存在一些问题，最终在我的机器上运行比较吃力，可见iFlow的后端工具链在优化效率、使用操作、用户手册建设、用户体验、兼容性等方面依然存在一些问题，需要持续不懈的继续努力。

下面进行iFlow的流程：

1 综合

运行以下命令：

```
./run_flow.py -d gcd -s synth | tee ./temp/syn.txt
```

首先检查综合相关的tcl脚本。

2 布局

2.1 floorplan

选择合适的floorplan的`CORE_AREA`、`DIE_AREA`参数；

运行以下命令：

```
./run_flow.py -d gcd -s floorplan -p synth | tee ./temp/floorplan.txt
```

2.2 tapcell

运行以下命令：

```
./run_flow.py -d gcd -s tapcell -p floorplan | tee ./temp/tapcell.txt
```

2.3 PDN

运行以下命令：

```
./run_flow.py -d gcd -s pdn -p tapcell | tee ./temp/pdn.txt
```

运行过程中出现报错：

```
    invoked from within
"pdngen::apply_pdn $config_file $verbose "} -errorline 9
[ERROR PDN-9999] Unexpected error: can't read "stripe_locs(met4,POWER)": no such element in array
%
```

回到前步floorplan

```
vim floorplan.openroad_1.2.0.tcl
```

调节floorplan的`DIE_AREA`和`CORE_AREA`

2.4 gplace

运行以下命令：

```
./run_flow.py -d gcd -s gplace -p pdn | tee ./temp/gplace.txt
```

2.5 resize

运行以下命令：

```
./run_flow.py -d gcd -s resize -p gplace | tee ./temp/resize.txt
```

2.6 dplace

运行以下命令：

```
./run_flow.py -d gcd -s dplace -p resize | tee ./temp/dplace.txt
```

2.7 CTS

运行以下命令：

```
./run_flow.py -d gcd -s cts -p dplace | tee ./temp/CTS.txt
```

2.8 filler

运行以下命令：

```
./run_flow.py -d gcd -s filler -p cts | tee ./temp/filler.txt
```

3 布线

3.1 groute

运行以下命令：

```
./run_flow.py -d gcd -s groute -p filler | tee ./temp/groute.txt
```

3.2 droute

运行以下命令：

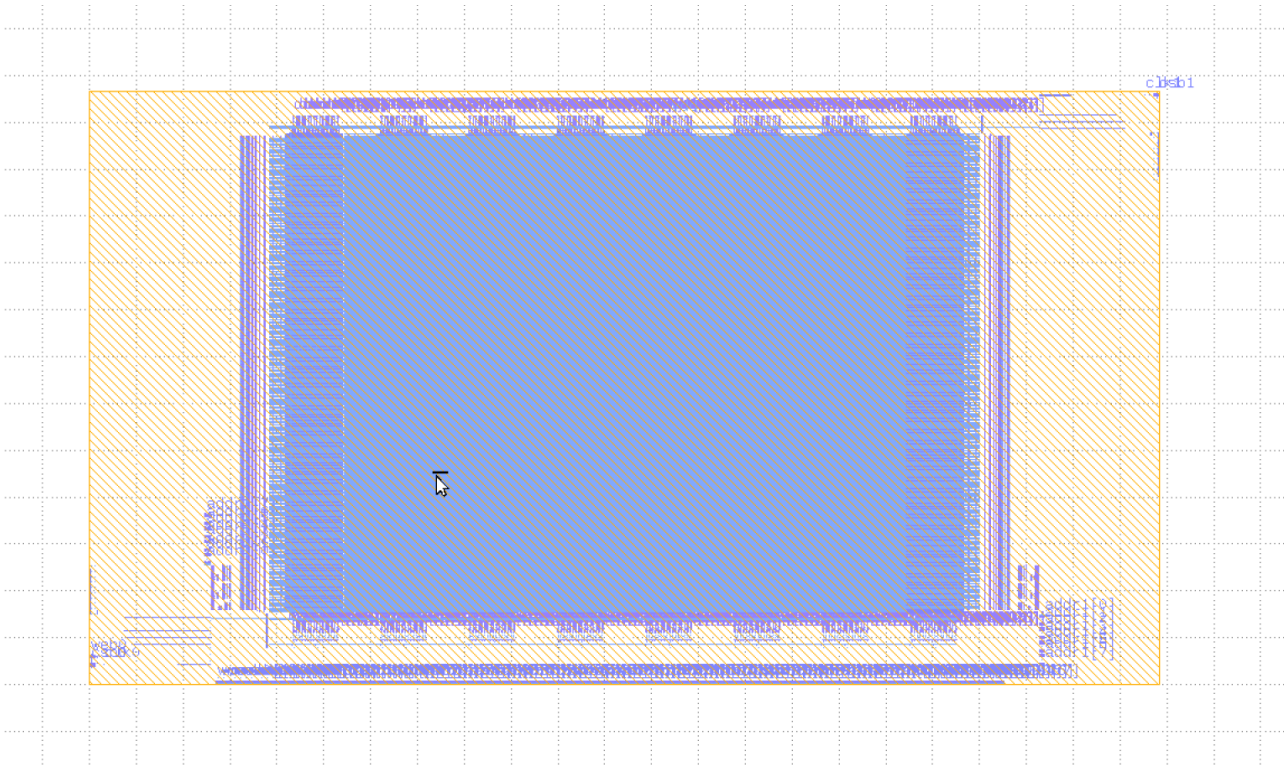
```
./run_flow.py -d gcd -s droute -p groute | tee ./temp/droute.txt
```

4 版图

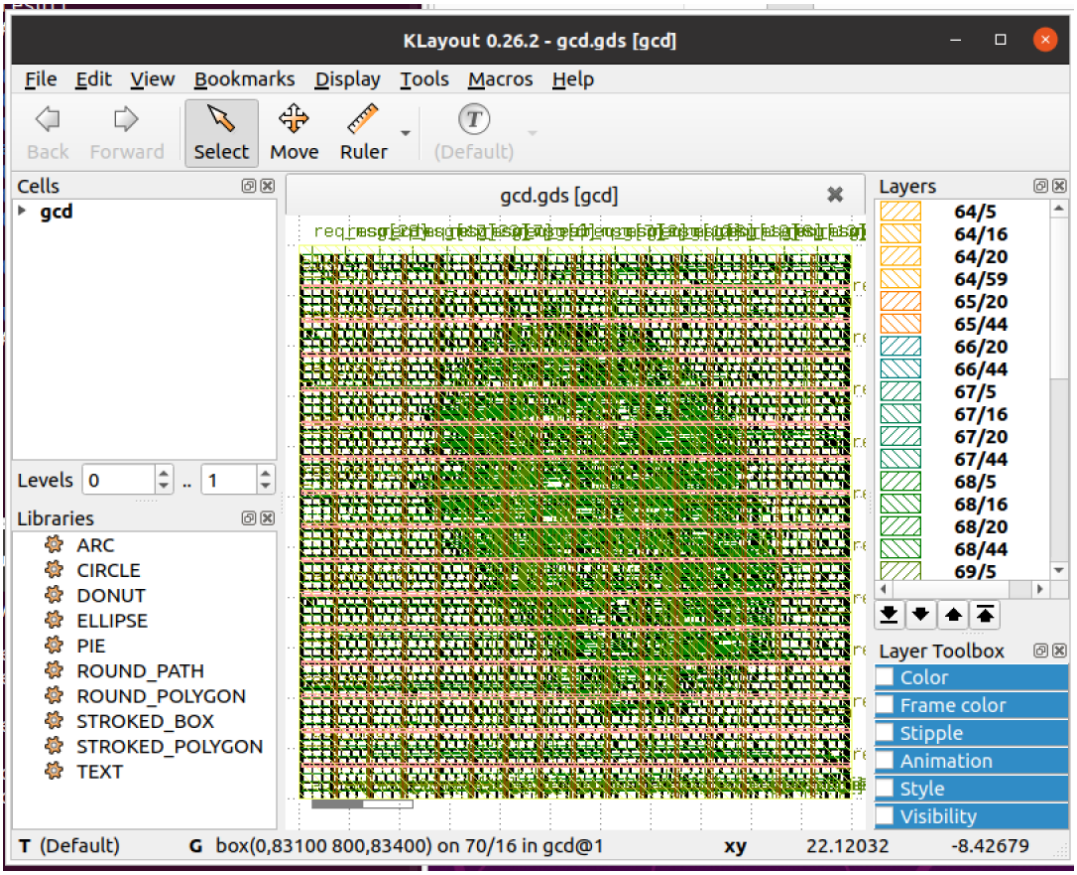
运行以下命令：

```
./run_flow.py -d gcd -s layout -p droute | tee ./temp/layout.txt
```

用klayout工具加在droute后输出的def文件，并同时导入design中的标准单元、IO cell以及marco的lef文件，def的结果如下图所示：



用klayout gcd.gds命令打开gcd绕线后的GDS如下图所示：



5 更换设计

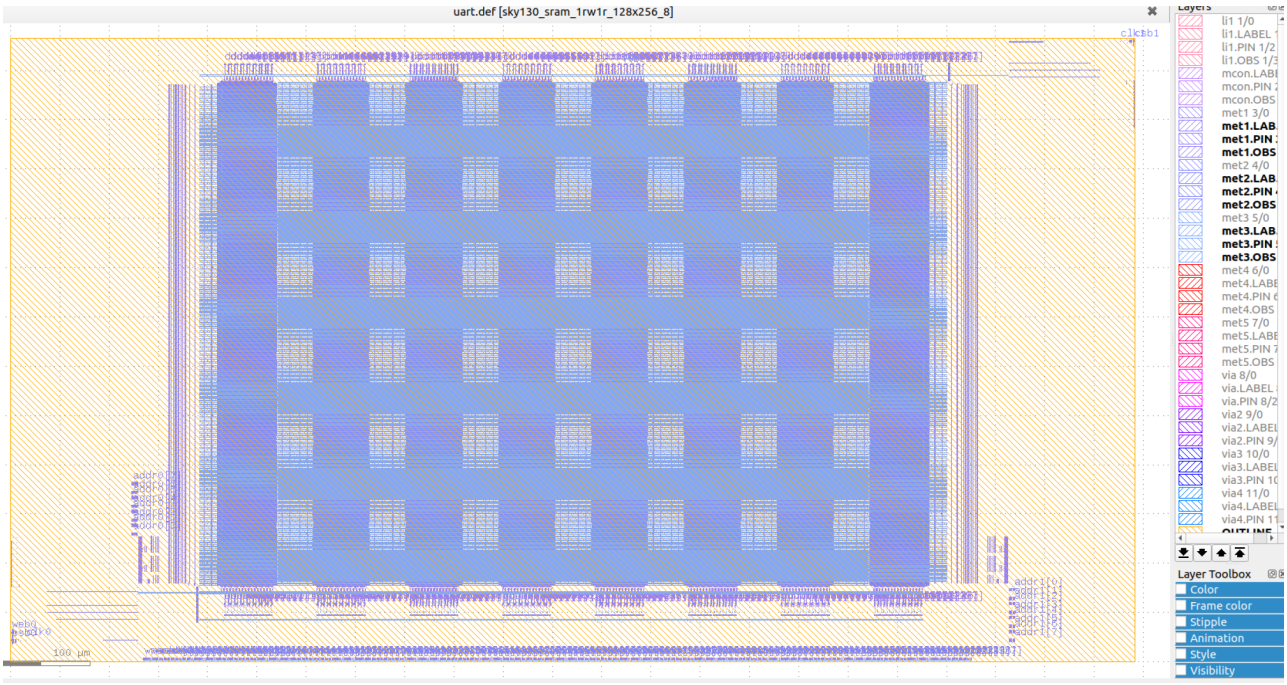
5.1 uart设计

由百门级设计过渡到千门级设计。

运行以下命令：

```
./run_flow.py -d uart -s synth,floorplan,tapcell,pdn,gplace,resize,dplace,cts,filler,groute,droute,layout -f sky130 -t HS -c TYP -v V1 -l V1 | tee ./temp/uart.txt
```

用klayout工具加在droute后输出的def文件，并同时导入design中的标准单元、IO cell以及marco的lef文件，def的结果如下图所示：



用klayout uart.gds命令打开uart绕线后的GDS如下图所示：

5.2 aes_cipher_top 设计

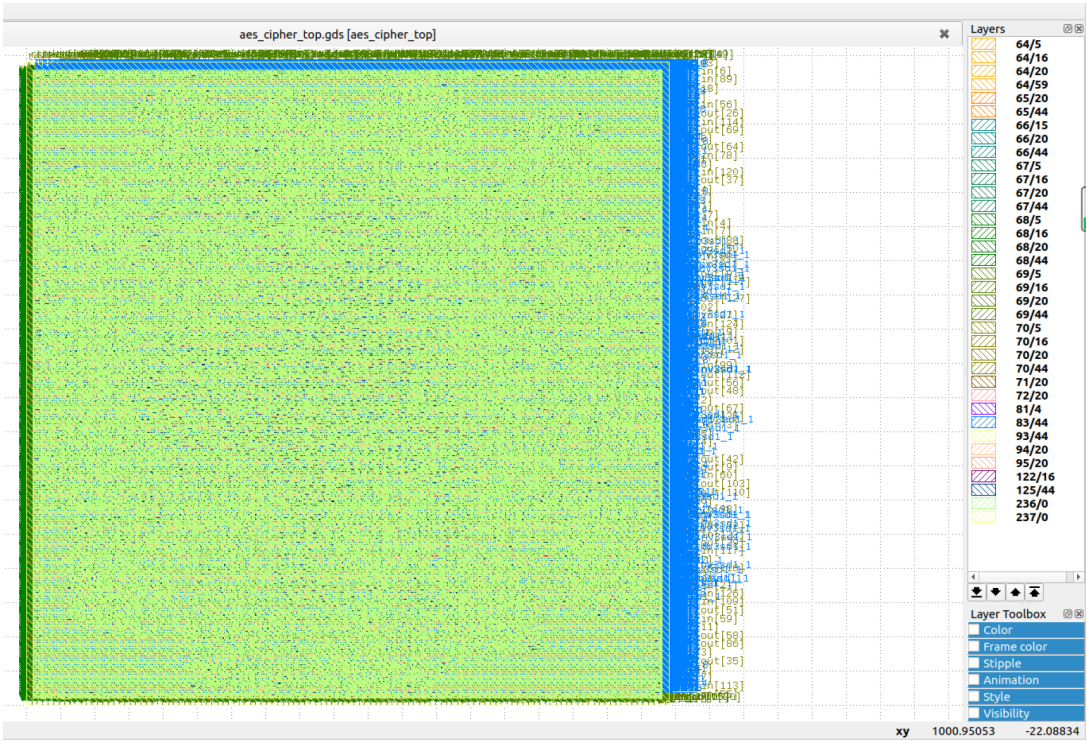
由千门级设计过渡到万门级设计。

运行以下命令：

```
./run_flow.py -d aes_cipher_top -s
synth,floorplan,tapcell,pdn,gplace,resize,dplace,cts,filler,groute,droute,layout -f sky130 -t HS -
c TYP -v V1 -l V1| tee ./temp/aes_cipher_top.txt
```

用klayout工具加在droute后输出的def文件，并同时导入design中的标准单元、IO cell以及marco的lef文件，def的结果如下图所示：

用klayout aes_cipher_top .gds命令打开aes_cipher_top绕线后的GDS如下图所示：



5.3 picorv32 设计

picorv32 是一个实现 RISC-V RV32IMC指令集的CPU内核，尝试 picorv32设计跑后端流程。

运行以下命令：

```
./run_flow.py -d picorv32 -s
synth,floorplan,tapcell,pdn,gplace,resize,dplace,cts,filler,groute,droute,layout -f sky130 -t HS -
c TYP -v V1 -l V1| tee ./temp/picorv32.txt
```

对这个设计进行了百般尝试，运行了若干时间，结果在我的机器上并没有最终得到gds结果，很遗憾。failed！😭我相信这是我机器的配置过低的问题，然后写实验报告记录这个过程，希望以后可以得到更加优异的平台，来运行iFlow的后端流程。

6 实验总结

这次实验总体上是比较成功的，经历了一个出现问题，解决问题的过程，成功地解决了一些问题，但是最终没有跑通picorv32依然是很遗憾的。这次实验，我学习到了很多，以前对linux下的相关工具使用并不是很熟，这次通过大作业熟悉了很多命令，是对工作学习很有益的一个过程。之前对后端流程也不熟悉，这次走了整个过程，对这样一个过程也有了初步了解，感谢老师提供的这样一个平台。在做作业的过程中，对开源工具链也进行了一些测试，发现了一些问题，希望iFlow的开发维护者在以后的工作中持续优化这些工具，加快建设用户手册、社区建设，提升优化效率、用户体验等，希望能够得到更多的公开资料供使用者研究使用，相信iFlow未来的表现。