
电子科技大学示范性微电子学院

实 验 报 告

(2024.11 -2025.01)

课程名称 IC 综合实验 2

实验名称 IC 综合实验 2

指导老师 王忆文

学生姓名 李卓霖，况明远，邓锦琪，徐子涵

学生学号 2022340101012、2022340102004、
2022020915006、2022340104032

组 号 11

电子科技大学示范性微电子学院

电子科技大学

实验报告

实验地点：清水河校区国际创新中心 B523

实验时间：2024.11-2025.01

报告目录

- 一、实验室名称：清水河校区国际菁蓉创新中心 B523
- 二、实验项目名称：IC 综合实验 2
- 三、实验学时：100
- 四、实验原理：请附页
- 五、实验目的：请附页
- 六、实验内容：请附页
- 七、实验器材（设备、元器件）：请附页
- 八、实验步骤：请附页
- 九、实验数据及结果分析：请附页
- 十、实验结论：请附页
- 十一、总结及心得体会：请附页
- 十二、对本实验过程及方法、手段的改进建议：请附页

实验报告成绩：_____

附页：

四、实验原理：

4.1. 自动布局（Placement）

自动布局的任务是将设计中逻辑网表中的标准单元和宏块放置在芯片布局中合理的位置。其主要目标是优化面积、减少布线长度、降低功耗，并满足时序要求。步骤和原理如下：

1.逻辑块初始摆放：

在 ICC 中，使用 `place_opt` 命令进行初始摆放。该命令会根据设计约束和网表信息，将标准单元初步放置在布局区域中。初始放置的目的是为后续的优化提供一个合理的起点。

2.布局优化：

在 ICC 中，`place_opt -post_place_opt` 命令会通过**模拟退火或二次规划**等算法，进一步优化标准单元的位置，减少连接线的总长度，从而降低布线复杂度和功耗。

3.拥挤分析：

分析布局区域的拥挤程度，避免过度拥挤导致后续布线困难。ICC 会动态调整标准单元位置，缓解高拥挤区域的压力。在 ICC 中，`check_place` 命令会检查布局区域的拥挤程度，并动态调整标准单元的位置，以缓解高拥挤区域的压力，确保后续布线的可行性。

4.时序优化：

`place_opt -post_place_opt -timing` 命令进行**时序优化**。它会根据时钟路径延迟和逻辑路径的建立保持时间，调整标准单元的位置，确保设计满足时序要求。

4.2. 自动布线（Routing）

自动布线负责在布局完成的基础上，将所有逻辑单元和引脚之间的连接按照设计规则完成。具体步骤如下所示：

1.全局布线：

将芯片区域细分为网格，并为每个信号网络规划大致的路径走向。他可以降低信号传输延迟并缓解布线拥挤，优化整体布线策略。。

2.详细布线:

依据全局布线所确定的路径，进行精确的**金属层布线及过孔分配**。严格遵循设计规则，如确保满足最小间距和最大电流密度等要求，以保证设计规则检查（DRC）的通过性。

3.时序与功耗优化:

通过调整**布线路径的长度**，减少由于寄生电阻和电容（RC）引起的延迟，同时解决信号完整性问题，如串扰和天线效应。

优化时钟树的布线，确保时钟信号的偏斜控制在可接受的范围内。

4.3. 时钟树综合（Clock Tree Synthesis, CTS）

简单概括下，时钟树综合就是指从某个 clock 的 root 点长到各个 sink 点的 clock buffer/inverter tree。工具试图将某个 clock 所属的所有 sinks 做到相同长度。从概念上，我们可以得到几个要点。

①clock 的 root 点需要定义清楚。这个可以通过 `create_clock` 来定义。如果是 `create_generated_clock`，它的 master clock 也是很清晰的，即知道 generate clock 的 source latency;

②clock 的 sinks 要知晓。

对于时钟树综合的目的，首先是 clock skew 尽量小，特别是对时钟质量要求比较高或者高频时钟，还有一个点是 clock latency 尽量短。为了达到以上两大目标，数字后端工程师任重道远。首先，我们拿到一个 design，需要先花点时间，理清楚 clock 结构，各种 mode 如何切换。同时，需要不断向前端设计人员请教 design 相关（后端需要知道的信息），比如哪些 clock 是需要同步，哪些是异步。不过我们本次实际仅采用同步信号，可简化问题分析。

时钟树综合的步骤列举如下:

1. 时钟源定义与时钟树准备

从逻辑设计中确定时钟源（如主时钟输入或 PLL 输出），明确需要驱动的触发器集合。分析时钟负载的分布，包括时钟引脚的数量、位置及负载大小，为后续的时钟树设计提供关键基础数据。

2.时钟树生成

根据特定算法生成时钟树拓扑，常用的方法包括：

- ①平衡二叉树：结构简单，便于实现低偏斜。
- ②H 树：对称性强，适用于时钟负载分布均匀的场景。
- ③网格时钟：常用于高性能设计，能实现极低的时钟偏斜，但功耗较高。

3.时钟缓冲器插入

在时钟树的不同分支插入缓冲器或门控单元，以提供足够的驱动能力并控制延迟。通过缓冲器的合理插入，可以平衡不同路径的延迟，减少偏斜。

4.时钟延迟平衡

针对不同路径的延迟，通过调整线长、插入额外缓冲器或修改线宽等方法，尽量实现所有触发器的时钟信号同步到达，确保延迟一致性。

5.时钟门控优化

在时钟树中插入门控单元，用于动态关闭未使用的时钟分支，从而降低动态功耗。这一环节中，门控单元的位置选择及控制信号的生成是优化的关键。

6.时钟树的布局布线

在生成时钟树拓扑后，需要对时钟线进行实际布局布线，需综合考虑布局约束和工艺规则。确保布线满足设计规则，同时尽量减少信号干扰和功耗损耗。

五、实验目的：

后端设计是数字芯片设计流程中的重要环节，其核心任务是将前端设计生成的 RTL 代码转化为实际的物理版图文件，以便芯片能够在晶圆厂进行制造。具体来说，后端设计的实验目的包括以下几个方面：

1. 实现逻辑到物理的转换

后端设计将经过验证的逻辑网表映射到具体的物理层，实现从功能设计到物理设计的转化，生成可制造的物理布局并完成芯片布线

2. 满足时序、面积和功耗目标

后端设计的主要技术目标是优化芯片的性能、功耗和面积。如调整布局、时钟树设计和路径布线，满足时序要求，合理分配标准单元和宏单元的位置，减少芯片占用的总面积，减少时钟树功耗、动态功耗和静态功耗（如泄漏电流）。

3. 确保制造可行性

后端设计需要确保生成的物理设计可在晶圆厂制造，符合工艺设计规则。后段设计需要确保布线间距、过孔尺寸、线宽等是否符合制造工艺要求，并通过 LVS、DRC、ERC 等检查，确保物理实现与逻辑设计一致，电流密度、功率分布等符合要求。

4. 集成外部接口

后端设计需要结合芯片封装和外部接口的要求完成芯片设计，分配和设计 IO 端口位置，确保符合封装工艺和 PCB 设计需求。

5. 生成最终的制造文件

后端设计的最终输出是用于晶圆厂流片的制造文件（如 GDSII），同时输出标准延迟文件（SDF 文件）用于后仿真

6. 后仿真

后仿真旨在验证物理实现是否与前端设计功能一致，确保逻辑正确性，确保芯片在加入了寄生参数，延迟条件下能够在目标频率正常工作。

六、实验内容：

我将本次 ICC 后端的设计部分以及后仿列为一个表格，并简单介绍其内容，如表 1 所示。

表 1 实验内容

步骤	分析
ICC 环境建立与数据导入	搭建后端设计的 ICC 实验环境，并导入前端设计生成的网表和约束文件；
布局规划	进行芯片布局规划，包括标准单元和宏单元的区域划分及 IO 端口位置设计；
预布线	完成关键信号路径的预布线，为后续优化提供参考；
布局	优化单元位置分布，减少面积和拥塞，确保物理设计符合时序目标；
时钟树综合	设计并优化时钟树结构，减少时钟偏斜和延迟，满足全局时序需求；
布线	对所有信号完成详细布线，并优化线长和寄生参

	数，确保布线规则的完整性；
收尾与检查	执行 DRC、LVS 等设计检查，确保物理设计符合工艺和制造规则；
后仿真	验证物理版图的功能与时序性能是否与前端设计一致，确保设计可靠性；

七、实验器材（设备、元器件）：Synopsys IC Compiler

八、实验步骤：

8.1. ICC 环境建立与数据导入

使用 ICC 第一步是进行环境建立与数据导入，准备好 ICC 流程中用到的工艺库、标准单元库、RC 延迟库、网表文件等。

打开隐藏的.Synopsys_DC.setup 文件，修改标准单元库的搜索路径，定义各变量名字，比如将我们的设计名字设定为”ic_chip”。

也可以借助脚本文件完成 ICC 环境建立与数据导入，比图形化界面更快捷。

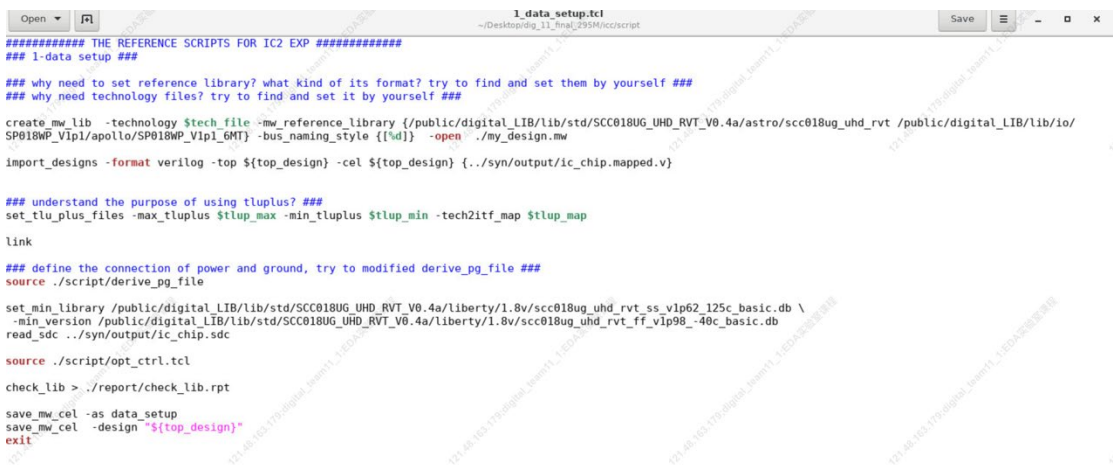


图.环境建立与数据导入的脚本

下面说明每个关键指令的具体含义：

create_mw_lib 为创建新的库。我们在 ICC 中保存的每一版后端设计都会保存在该库中，可以说一个库就是一个工程文件。并且指定库需要依赖的工艺文件。

Import_designs 为载入综合后的网表。

Set_tlu_plus_files 为加载该工艺库对应的 TLU+延迟参数库。

随后加载标准单元库和 SDC 约束文件，即 `set_min_library` 和 `read_sdc` 指令。前者为设计提供器件的标准单元，后者提供整体设计的时序、面积等约束。

最后生成汇报文件，并保存为一个“CEL”。至此，第一步环境建立与数据导入结束。

8.2. 布局规划

我们借助脚本来阐述布局规划阶段的具体流程。



```
## 2: floorplan and PG Network Synthesis ##
open_mw lib ./my_design.mw
open_mw cel ${top_design}

#1000# modify the pad description files according to your design.
source ./script/pad_cell_cons.tcl

#1000# from ICC windows, create floorplan manually by using default setting
create_floorplan -control_type width_and_height -core_width 500 -core_height 500 -left_io2core 20 -bottom_io2core 20 -right_io2core 20 -top_io2core 20
#1000# then measure the core size(???um X ??? um )

#1000# set the space between io2core (4 sides)
#1000# then create floorplan finally

#1000# modify below file for insertting pad filler.
source ./script/insert_pad_filler.tcl

#1000# modify below file to define the P/G conection.
source ./script/connect_pg.tcl

#add_tap_cell_array -master_cell_name FILLTIEUHD -distance 30 -pattern stagger_every_other_row -connect_power_name VDD -connect_ground_name VSS
add_tap_cell_array -master_cell_name FILLTIEUHD -distance 30 -pattern normal -connect_power_name VDD -connect_ground_name VSS

create_pad_rings

save_mw cel -as floorplan_init

#1000# which metal layers (METAL?) should be used for power ring? (vertical and horizontal)
#1000# which metal layers (METAL?) should be used for power strap? (vertical and horizontal)
#1000# define the width for power ring and strap respectively
```

图.布局规划使用的脚本文件

布局规划的第一步是导入引脚布局，这是一个需要我们自己定义和编写的脚本文件。



```
## pad count--total 28 pins
##
## signal-----21
## VDD for core (PVDD1R)----2 (double bonding)
## GND for core (PVSS1R)----2 (double bonding)
## VDD for IO (PVDD2R)----1
## GND for IO (PVSS1R)----2 (double bonding)

#Create corners and P/G pads
create_cell {cornerll cornerlr cornerur cornerur} PCORNERW

create_cell {vdd_core_1} PVDD1W
create_cell {vdd_core_2} PVDD1W

create_cell {vss_core_1} PVSS1W
create_cell {vss_core_2} PVSS1W

create_cell {vdd_io_1} PVDD2W
create_cell {vss_io_1} PVSS2W
create_cell {vss_io_2} PVSS2W

#Define corner pad locations
set_pad_physical_constraints -pad_name "cornerur" -side 1
set_pad_physical_constraints -pad_name "cornerur" -side 2
set_pad_physical_constraints -pad_name "cornerlr" -side 3
set_pad_physical_constraints -pad_name "cornerll" -side 4

#Define signal and PG pad locations

#Left side
set_pad_physical_constraints -pad_name "IO_PAD3*" -side 1 -order 1
set_pad_physical_constraints -pad_name "IO_PAD4*" -side 1 -order 2
set_pad_physical_constraints -pad_name "IO_PAD5*" -side 1 -order 3
set_pad_physical_constraints -pad_name "IO_PAD6*" -side 1 -order 4
set_pad_physical_constraints -pad_name "IO_PAD7*" -side 1 -order 5
set_pad_physical_constraints -pad_name "IO_PAD8*" -side 1 -order 6
set_pad_physical_constraints -pad_name "IO_PAD9*" -side 1 -order 7
```

图.引脚约束脚本

在引脚布局规划的脚本中，我们先是在芯片的四个角定义了 Corner，用于填充位置，因为芯片的四个角不用于任何功能。为了让 ICC 能够识别电源和低，我

们随后定义了 7 个电源引脚，分别为 4 个核心供电引脚，3 个 IO 供电引脚。
最后是分配引脚的位置，Side1~4 分别对应了芯片的左侧、上侧、右侧、下侧。
Order1~7 分别对应顺时针方向的顺序。至此，定义引脚的脚本编写完毕

布局规划的**第二步是创建布局规划**。我们设置了 500*500 大小的核心，核心与四边 IO Pad 间距为 20，随后可以观察到版图布局发生的变化。

```
#top side
set_pad_physical_constraints -pad_name "IO_PAD10" -side 2 -order 1
set_pad_physical_constraints -pad_name "vdd_io_1" -side 2 -order 2
set_pad_physical_constraints -pad_name "vss_io_1" -side 2 -order 3
set_pad_physical_constraints -pad_name "vss_io_2" -side 2 -order 4
set_pad_physical_constraints -pad_name "vss_core_2" -side 2 -order 5
set_pad_physical_constraints -pad_name "vdd_core_2" -side 2 -order 6
set_pad_physical_constraints -pad_name "IO_PAD1" -side 2 -order 7

#right side
set_pad_physical_constraints -pad_name "IO_PAD16" -side 3 -order 7
set_pad_physical_constraints -pad_name "IO_PAD17" -side 3 -order 6
set_pad_physical_constraints -pad_name "IO_PAD18" -side 3 -order 5
set_pad_physical_constraints -pad_name "IO_PAD19" -side 3 -order 4
set_pad_physical_constraints -pad_name "IO_PAD20" -side 3 -order 3
set_pad_physical_constraints -pad_name "IO_PAD21" -side 3 -order 2
set_pad_physical_constraints -pad_name "IO_PAD22" -side 3 -order 1

#bottom side
set_pad_physical_constraints -pad_name "IO_PAD11" -side 4 -order 1
set_pad_physical_constraints -pad_name "IO_PAD15" -side 4 -order 2
set_pad_physical_constraints -pad_name "IO_PAD13" -side 4 -order 3
set_pad_physical_constraints -pad_name "vdd_core_1" -side 4 -order 4
set_pad_physical_constraints -pad_name "vss_core_1" -side 4 -order 5
set_pad_physical_constraints -pad_name "IO_PAD14" -side 4 -order 6
set_pad_physical_constraints -pad_name "IO_PAD2" -side 4 -order 7
```



上图.引脚约束文件 下图.布局规划完成后版图界面

布局规划的**第三步是插入 Filler**，用于填充 Pad 之间的区域，然后连接电源和地（虚拟连接，并为布线）。

布局规划的第三步是创建电源环和电源支路（Straps）。

```

set_fp_rail_constraints -add_layer -layer METAL6 -direction horizontal -max_strap 50 -min_strap 10 -max_width 6 -min_width 3 -spacing 1
set_fp_rail_constraints -add_layer -layer METAL5 -direction vertical -max_strap 50 -min_strap 10 -max_width 6 -min_width 3 -spacing 1
set_fp_rail_constraints -set_ring -nets {VDD VSS} -horizontal_ring_layer {METAL4} -vertical_ring_layer {METAL3} -ring_max_width 4 -ring_min_width 2 -
extend_strap core_ring -ring_spacing 0.8 -ring_offset 4
synthesize_fp_rail -nets {VDD VSS} -voltage_supply 1.8 -target_voltage_drop 160 -synthesize_power_plan -power_budget 300 -pad_masters {VDD:PVDD1W.FRAME
VSS:PVSS1W.FRAME}
commit_fp_rail
preroute_instances
preroute_standard_cells -fill_empty_rows -remove_floating_pieces
#1000# Check the max IR-drop of your design?
analyze_fp_rail -nets {VDD VSS} -power_budget 300 -voltage_supply 1.8 -pad_masters {VDD:PVDD1W.FRAME VSS:PVSS1W.FRAME}
save_mv_cel -as floorplan_pns
#avoid to place cell under the cross-point between M4 and M5
#set_pnet_options -complete "METAL4 METAL5"

report_timing > ./report/timing_fp.rpt
save_mv_cel -as floorplan_complete
save_mv_cel -design "Stop_design"
exit

```

图.布局规划脚本

分别在第六层金属、第五层金属创建水平和垂直电源支路，在第四层金属、第三层金属创建水平和垂直电源环。创建完成后，进行供电网络的综合，用于将上述提到的电源网络相连接，满足一定的电压降要求。我们在脚本中设定目标的电压降为 160mV。

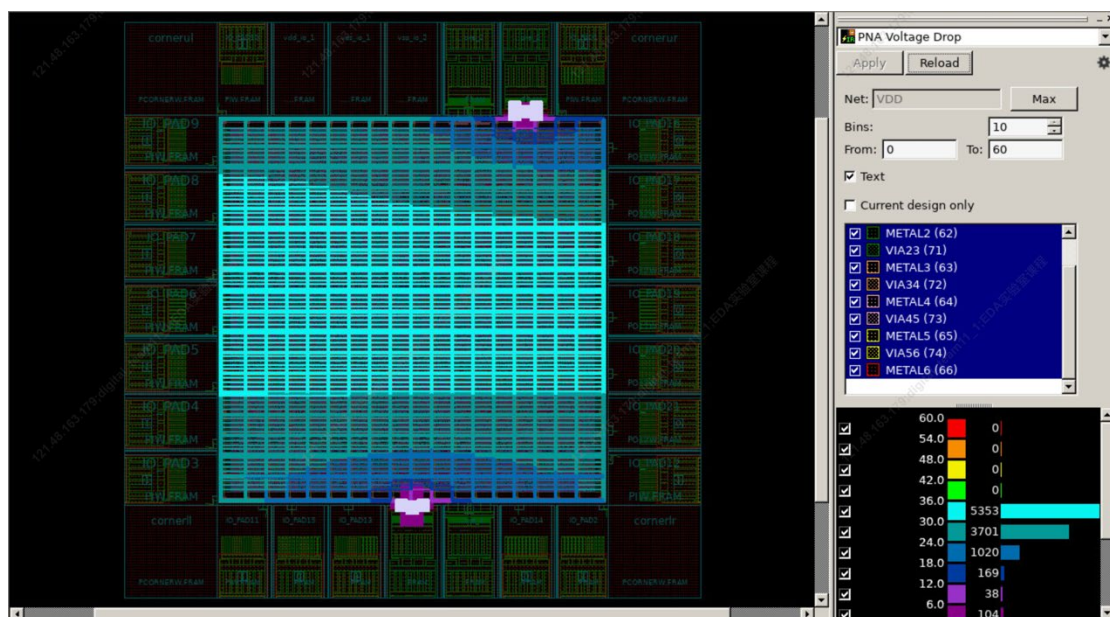
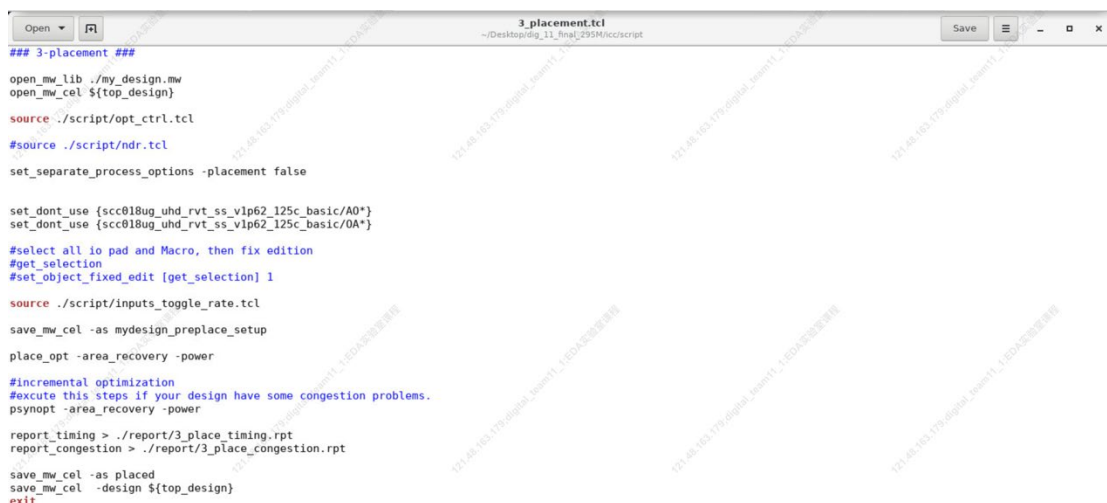


图.电压降分析界面

在电压降分析中，可以看到**最大电压降为 36mV**，远低于题目要求的 180mV。保存对应的文件，至此布局规划结束。

8.3. 布局

结合脚本来阐述布局阶段主要步骤



```
## 3-placement ##
open_mw_lib ./my_design.mw
open_mw_cel ${top_design}
source ./script/opt_ctrl.tcl
#source ./script/ndr.tcl
set_separate_process_options -placement false

set_dont_use {scc018ug_uhd_rvt_ss_vlp62_125c_basic/A0*}
set_dont_use {scc018ug_uhd_rvt_ss_vlp62_125c_basic/OA*}

#select all io pad and Macro, then fix edition
#get_selection
#set_object_fixed_edit [get_selection] 1

source ./script/inputs_toggle_rate.tcl

save_mw_cel -as mydesign_preplace_setup
place_opt -area_recovery -power

#incremental optimization
#execute this steps if your design have some congestion problems.
psynopt -area_recovery -power

report_timing > ./report/3_place_timing.rpt
report_congestion > ./report/3_place_congestion.rpt

save_mw_cel -as placed
save_mw_cel -design ${top_design}
exit
```

图.布局使用的脚本

布局阶段需要修改 `inputs_toggle_rate` 脚本文件，使输入的翻转率等于我们目标时钟频率



```
#set_switching_activity -period 20 -static_probability 0.5 [all_inputs]
set_switching_activity -toggle_rate 2 -period 3.33 -static_probability 0.5 [all_inputs]
```

图.设置翻转率

随后进行布局操作，即 `place_opt` 指令，进行第一轮布局，并且设定回收面积和优化功耗的选项。

进行第一轮布局操作后，我们进行布局完成时序汇报，系统会根据当前布局好的情况生成一个大概的时序报告，汇报违例的关键路径，我们在此发现了少许的建立时间违例。

于是我们使用 `psynopt` 进行增量优化，希望系统帮我们修复时序违例，但结果是建立时间依然存在违例现象。

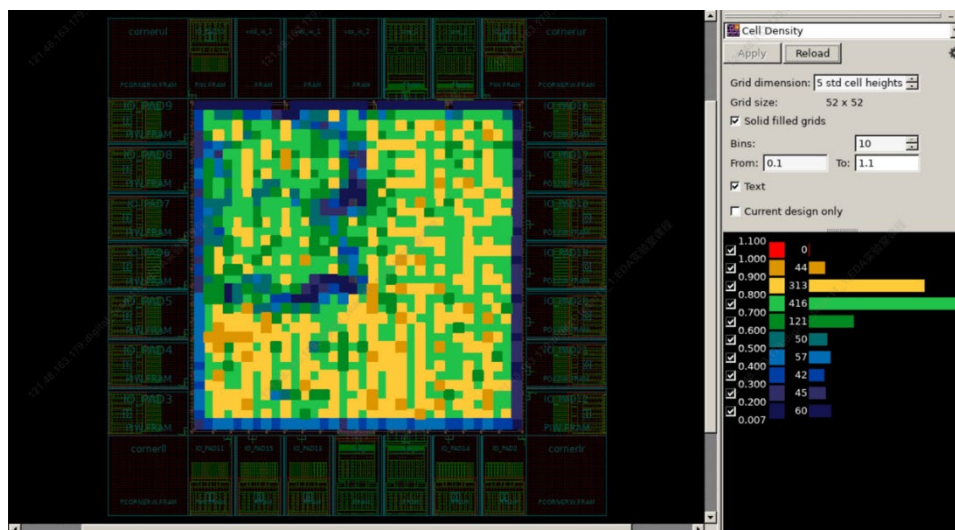


图.布局完成后阻塞情况

查看布局的阻塞情况，可以发现并未出现明显阻塞。至此布局阶段结束

8.4. 时钟树综合

```
## 4-clock tree synthesis ##

open_mw_lib ./my_design.mw
open_mw_cel ${top_design}

set_dont_use {scc018ug_uhd_rvt_ss_vip62_125c_basic/A0*}
set_dont_use {scc018ug_uhd_rvt_ss_vip62_125c_basic/OA*}

remove_ideal_network -all

set_separate_process_options -placement false

set_clock_tree_options -target_skew 0.15

set_clock_uncertainty -setup 0.0666 [get_clocks clk_io]
set_clock_uncertainty -hold 0.0333 [get_clocks clk_io]

set_clock_tree_references {CLKBUFUHDV1 CLKBUFUHDV2 CLKBUFUHDV3 CLKBUFUHDV4 CLKBUFUHDV6 CLKBUFUHDV8 CLKBUFUHDV16 CLKBUFUHDV20 CLKBUFUHDV24 \
CLKINUHDV1 CLKINUHDV2 CLKINUHDV3 CLKINUHDV4 CLKINUHDV6 CLKINUHDV8 CLKINUHDV16 CLKINUHDV20 CLKINUHDV24 }

set_min_library scc018ug_uhd_rvt_ss_vip62_125c_basic.db \
-min_version scc018ug_uhd_rvt_ff_vip98_-40c_basic.db

set_operating_conditions -analysis_type bc wc \
-max_library scc018ug_uhd_rvt_ss_vip62_125c_basic -max ss_vip62_125c \
-min_library scc018ug_uhd_rvt_ff_vip98_-40c_basic -min ff_vip98_-40c

#source ./script/ndr.tcl

clock_opt -only_cts -no_clock_route

save_mw_cel -as clock_opt_cts

report_constraint -all > ./report/4_cts_only_constraint.rpt

remove_ideal_network -all
set_propagated_clock [all_clocks]
set_fix_hold [all_clocks]
```

图.时钟树综合阶段脚本

时钟树综合阶段，我们首先要给目标的时钟树设定约束条件，实际的约束需要结合目标时钟频率动态调整。我们此处的目标时钟频率是 300MHz，因此我们选择目标：时钟偏斜 0.15ns、建立时间不确定性 0.0666ns、保持时间不确定性 0.0333ns。

其次是设定参考时钟树、电路工作条件，分别使用 `set_clock_tree_references` 和 `set_operating_conditions` 命令。

设定完成后，开始第一轮时钟树综合。第一轮时钟树综合只综合，不布线，更像是整体电路时序进行分析，生成时序汇报文件。

```
#source ./script/ndr.tcl

clock_opt -only_cts -no_clock_route

save_mw_cel -as clock_opt_cts

report_constraint -all > ./report/4_cts_only_constraint.rpt

remove_ideal_network -all
set_propagated_clock [all_clocks]
set_fix_hold [all_clocks]

set physopt_area_critical_range 1

extract_rc

clock_opt -only_psyn -area_recovery -no_clock_route

route_zrt_group -all_clock_nets -reuse_existing_global_route true

report_constraint -all > ./report/4_cts_opt_constraint.rpt
report_clock_tree > ./report/4_cts_clock_tree.rpt
report_clock_timing -type skew > ./report/4_cts_skew.rpt
report_timing > ./report/4_cts_timing.rpt
report_timing -delay_type min >> ./report/4_cts_timing.rpt

save_mw_cel -as clock_opt_route
save_mw_cel -design ${top_design}
exit
```

图.时钟树综合使用的脚本

第二轮综合为**增量综合**，软件尽可能的修复第一轮综合出现的时序违例情况。使用 `set_fix_hold` 命令，标记需要解决保持时间违例的时钟。

增量综合完成后，开始**时钟树的布线**。因为时钟树在数字电路中的重要性以及复杂程度，需要优先布线。查看时钟树的布线情况，发现时钟树布线始于时钟 IO Pad，终止于芯片的边界。

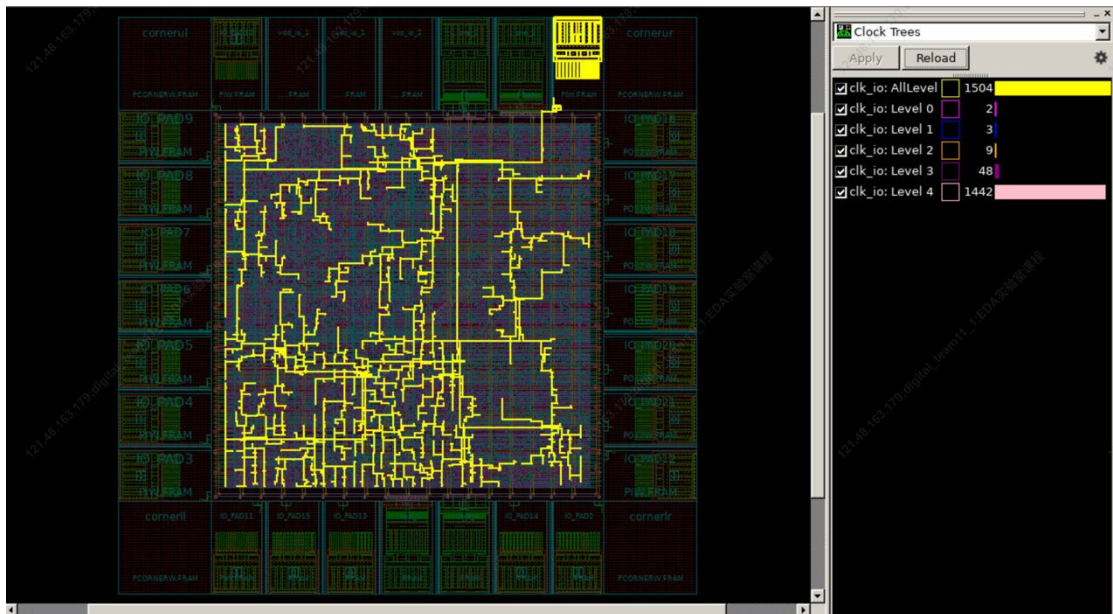


图.时钟树布线结果

保存文件，汇报时序违例情况，发现建立时间还是存在些许违例，保持时间无违例。具体分析将在第七节汇报。

8.5. 布线

```
open_mw_lib ./my_design.mw
open_mw_cel ${top_design}

set_dont_use [get_cells scc018ug_uhd_rvt_ss_vlp62_125c_basic/A0*]
set_dont_use [get_cells scc018ug_uhd_rvt_ss_vlp62_125c_basic/OA*]

set_separate_process_options -placement false

remove_ideal_network -all
set_propagated_clock [all_clocks]
set_fix_hold [all_clocks]

### setting if the design have special requirement ###
#source ./script/common_optimization_settings.icc.tcl
#source ./script/common_placement_settings.tcl
#source ./script/common_post_cts_timing_settings.tcl

report_constraint -all > ./report/5_timing_before_route.rpt

clock_opt -only_hold_time

verify_pg_nets

preroute_standard_cells -remove_floating_pieces

verify_pg_nets

set_route_zrt_common_options -post_detail_route_redundant_via_insertion medium
set_route_zrt_detail_options -optimize_wire_via_effort_level hard

route_opt -initial_route_only

route_opt -skip_initial_route -power
```

图.布线阶段使用的脚本

布线阶段首先生成时序汇报，汇报布线之前的时序违例情况。随后再进行一轮时钟树优化，用于插入 Buffer，修复保持时间违例。并且确认电源与地连线正常。

我们使用 `-optimize_wire_via_effort_level hard` 指令，增加布线优化力度，这会消耗更长的时间和更多的 CPU 资源，可以得到更好的布线结果。

第一轮布线为初始布线。

第二轮布线跳过了初始布线，进行详细布线。

第三轮布线为增量布线，用于修复布线中的 DRC 和 LVS 违例情况。

```
### DRC
verify_zrt_route > ./report/5_route_DRC.rpt

### LVS
verify_lvs > ./report/5_route_LVS.rpt

### route incrementally if have DRC and LVS issues.
route_opt -incremental

save_mw_cel -as route
save_mw_cel -design ${top_design}
exit
```

图.布线阶段使用的脚本（续）

布线完成后，使用 `verify_zrt_route`、`verify_lvs` 进行布线后的 DRC 和 LVS 检查，可以看到报告中看到 DRC 和 LVS 检查结果。

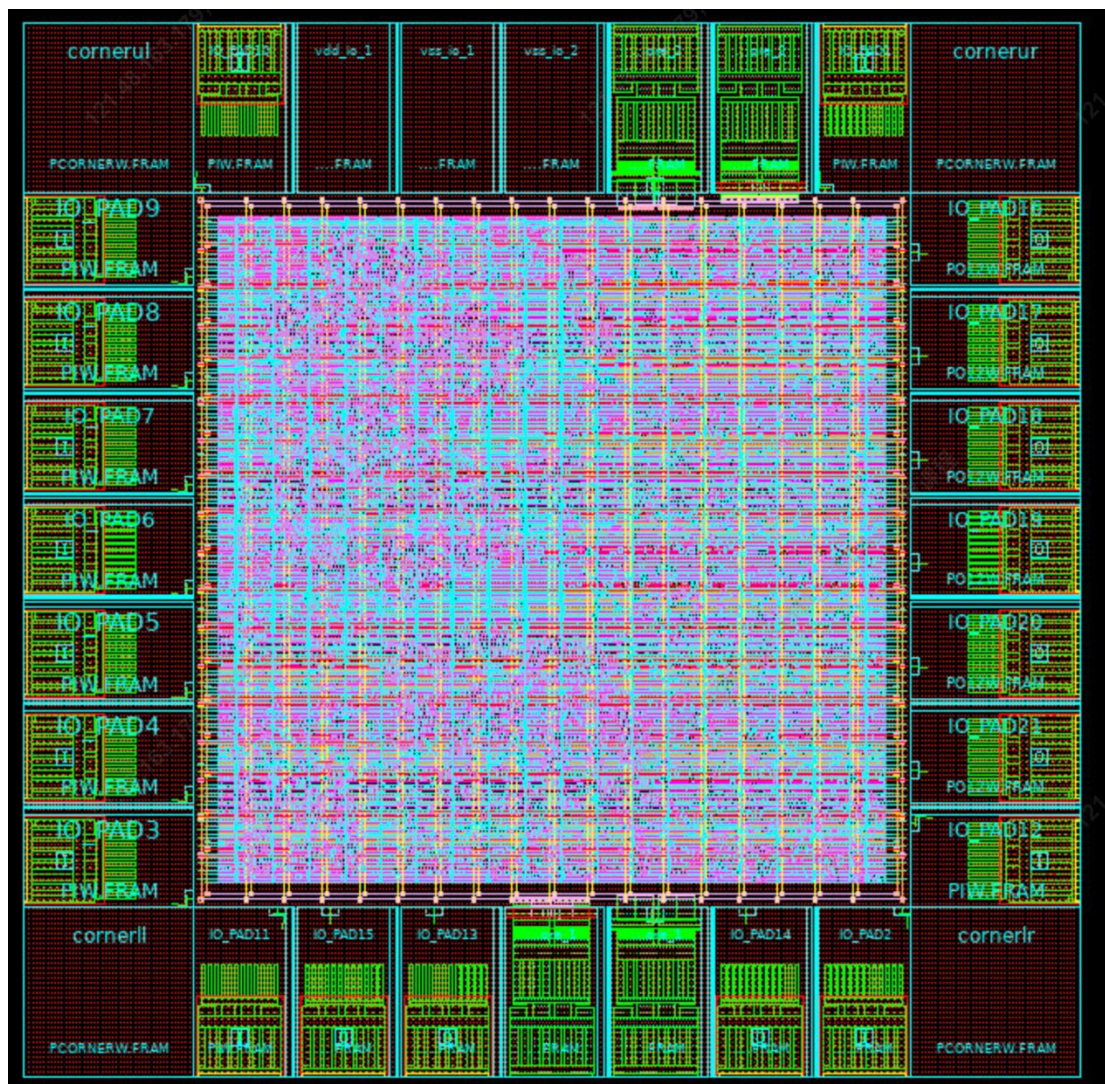


图.布线后版图视图

8.6. 收尾与检查

收尾与检查阶段主要是生成最终的报告，包括 DRC、LVS 检查报告、时序违例报告，天线效应检查报告，以及导出标准延迟文件、GDSII 文件和用于后仿真的网表文件。

该阶段比较关键的命令有：

spread_zrt_wires、widen_zrt_wires: 进行连线扩展和加宽连线以减小关键面积，由于在实际制造中，间距较小的连线容易短路，较细的连线容易断路，该命令旨在提高芯片的可制造性。

Write_sdf-version 1.0: 用于进行 1.0 版本的 SDF 文件的输出，1.0 版本的 SDF 文件较于 2.1 版本的文件在后仿中的反标率更高。

```

###6-Chip Finishing#####
open_mw_lib ./my_design.mw
open_mw_cel ${top_design}

set_dont_use [get_cells scc818ug_uhd_rvt_ss_vlp62_125c_basic/A0*]
set_dont_use [get_cells scc818ug_uhd_rvt_ss_vlp62_125c_basic/OA*]

set_separate_process_options -placement false

#####
# LOAD AND ANALYZE DESIGN #
#####

# Check for DRC violations
verify_zrt_route > ./report/6_finish_1st_DRC.rpt

# Check for LVS violations
verify_lvs > ./report/6_finish_1st_LVS.rpt

# Check for constraint violations
report_constraints -all > ./report/6_finish_1st_constraint.rpt

#####
# CRITICAL AREA REDUCTION #
#####
#spread & widen wires

spread_zrt_wires
widen_zrt_wires

verify_zrt_route > ./report/6_finish_2nd_DRC.rpt
verify_lvs > ./report/6_finish_2nd_LVS.rpt
report_constraints -all > ./report/6_finish_2nd_constraint.rpt
rc

save_mw_cel -as chip_finish_des

#####
# FIXING ANTENNA RULE VIOLATIONS WITH DIODES #
#####

source -echo ./script/antenna_6m_1tm.tcl
report_antenna_rules > ./report/6_antenna.rule
verify_zrt_route

set_route_zrt_detail_options -insert_diodes_during_routing true
route_zrt_detail -incremental true

verify_lvs
derive_pg_connection -power_net VDD -power_pin VDD -ground_net VSS -ground_pin VSS
derive_pg_connection -power_net VDD -ground_net VSS -tie

verify_zrt_route > ./report/6_finish_antenna_DRC.rpt
verify_lvs > ./report/6_finish_antenna_LVS.rpt
rc

save_mw_cel -as chip_finish_antenna

#####
# INSERT STANDARD CELL FILLERS #
#####

remove_stdcell_filler -stdcell

set fillers "F_FILLUHD32 F_FILLUHD16 F_FILLUHD8 F_FILLUHD4 F_FILLUHD2 F_FILLUHD1"
insert_stdcell_filler -respect_keepout -connect_to_power VDD -connect_to_ground VSS -cell_with_metal $fillers

derive_pg_connection -power_net VDD -power_pin VDD -cells [get_flat_cells *] -reconnect
derive_pg_connection -ground_net VSS -ground_pin VSS -cells [get_flat_cells *] -reconnect

verify_zrt_route
verify_lvs
rc

#####
# SAVE DESIGN AND STREAM OUT #
#####

save_mw_cel -as chip_finish_final
save_mw_cel -design ${top_design}

report_timing > ./report/6_finish_timing.rpt
report_timing -delay_type min >> ./report/6_finish_timing.rpt

write_sdf -significant_digits 6 ./output/ic.sdf
write_verilog -no_core_filler_cells -no_pad_filler_cells -no_corner_pad_cells ./output/ic.output1.v
write_verilog -no_physical_only_cells -supply_statement none ./output/ic.output2.v

close_mw_cel

set_write_stream_options \
-child_depth 255 \
-map_layer ./script/gds20utlayer_twinwell.map \
-output_filling_fill \
-output_outdated_fill \
-keep_data_type \
-max_name_length 255 \
-output_net_name_as_property 1 \
-output_instance_name_as_property 1 \
-output_pin {geometry Text} \
-output_polygon_pin \
-output_design_intent

write_stream -cells $top_design -format gds ./output/ic.gds

```

图.收尾与检查使用的脚本文件

8.7. 后仿真

后仿真阶段，我们使用以下命令进行时序反标和仿真，分别进行了 max 情况的仿真与 min 情况下的仿真，对应建立时间与保持时间。

```
Vcs -full64 -f filelist_post -R +sdfverbose -diag=sdf +v2k -timescale=1ns/10ps -
debug_all +define+SDF -sdf max:ic_chip: ic.sdf +vpdfile+"max.vpd" +memcbk
```


+neg_tchk

Vcs -full64 -f filelist_post -R +sdfverbose -diag=sdf +v2k -timescale=1ns/10ps -
debug_all +define+SDF -sdf min:ic_chip: ic.sdf +vpdfile+"max.vpd" +memcbk

+neg_tchk

其中+sdfverbose -sdf max:ic_chip:ic.sdf 表示启用 sdf 反标，+neg_tchk 表示
考虑负的延迟。

九、实验数据及结果分析：

9.1. 后端设计最终的时序情况：

clock clk_io (rise edge)	0.00	0.00	ting Conditions: SP018WP_V1p1_max Library: SP018WP_V1p1_max
clock network delay (propagated)	1.17	1.17	
u_CNN_TOP_POOL_CTL_inst_state_reg_1/CK (DRQUHDV3)	0.00	1.17 r	mation: Percent of Arnoldi-based delays = 13.57%
u_CNN_TOP_POOL_CTL_inst_state_reg_1/Q (DRQUHDV3)	0.62	1.79 r	
U2030/ZN (INUHDV3)	0.08 &	1.87 f	rtpoint: mode_io (input port clocked by clk_io)
U428/ZN (CLKNAND2UHDV8)	0.12 &	1.99 r	point: u_CNN_TOP_WEIGHT_inst_weight_REG_reg_0_6_2_
U1413/ZN (CLKINUHDV24)	0.11 @	2.09 f	(rising edge-triggered flip-flop clocked by clk_io)
U6419/ZN (NAND2XBUHDV1)	0.11 @	2.20 r	h Group: INPUTS
U5744/ZN (INUHDV2)	0.09 &	2.29 f	h Type: max
U3663/ZN (NOR2UHDV6)	0.17 &	2.46 r	
U6326/ZN (NAND2UHDV3)	0.09 &	2.55 f	
U5751/ZN (CLKNAND2UHDV6)	0.09 &	2.64 r	nt
U6367/ZN (NAND3UHDV2)	0.12 &	2.76 f	Incr Path
U1235/ZN (NAND4UHDV2)	0.12 &	2.88 r	ck clk_io (rise edge)
U1603/ZN (NAND4XBBUHDV3)	0.14 &	3.02 f	ck network delay (propagated)
U1059/ZN (CLKINUHDV3)	0.08 &	3.09 r	ut external delay
U2380/ZN (NAND2XBUHDV2)	0.08 &	3.17 f	e io (in)
U2087/ZN (NAND2UHDV2)	0.08 &	3.25 r	PAD12/C (PIW)
U255/ZN (INUHDV2)	0.06 &	3.31 f	41/ZN (NAND2UHDV1)
U6019/Z (MAJ23UHDV2)	0.30 &	3.61 f	40/ZN (INUHDV1)
U3739/ZN (NAND3UHDV2)	0.10 &	3.71 r	58/ZN (NAND2UHDV1)
U3738/ZN (NAND3UHDV2)	0.12 &	3.83 f	0/Z (AND2UHDV8)
U2825/ZN (NAND2UHDV3)	0.11 &	3.93 r	2/Z (BUFUHDV2)
U1100/ZN (INUHDV2)	0.07 &	4.01 f	80/ZN (NAND2UHDV0P4)
U5786/ZN (CLKNOR2UHDV8)	0.11 &	4.11 r	82/ZN (NAND3UHDV1)
U5679/ZN (NAND2UHDV1)	0.09 &	4.20 f	NN_TOP_WEIGHT_inst_weight_REG_reg_0_6_2_/D (DRQUHDV0P7)
U5855/ZN (NAND3UHDV1)	0.10 &	4.31 r	0.00 &
u_CNN_TOP_FC_DATA_inst_FC_REG_reg_8_3_/D (DRQUHDV3)	0.00 &	4.31 r	a arrival time
data arrival time	0.00 &	4.31	3.84
clock clk_io (rise edge)	3.33	3.33	ck clk_io (rise edge)
clock network delay (propagated)	1.08	4.41	ck network delay (propagated)
clock uncertainty	-0.07	4.35	ck uncertainty
u_CNN_TOP_FC_DATA_inst_FC_REG_reg_8_3_/CK (DRQUHDV3)	0.00	4.35 r	NN_TOP_WEIGHT_inst_weight_REG_reg_0_6_2_/CK (DRQUHDV0P7)
library setup time	-0.30	4.05	0.00
data required time		4.05	-0.31
data required time		4.05	4.03
data required time		4.05	4.03
data arrival time		-4.31	4.03
			-3.84
			ck (MET)
			0.19

图.最终的关键路径建立时间时序情况

通过时序报告我们可以得知，在经历了布局、时钟树综合、布线流程后，关键路径的建立时间并能为很好的收敛，因此我们最终的设计并不能运行在DC 综合时的频率之上，需要降低时钟频率使用，这一点也在后仿真结果得以体现。保持时间则收敛。

* Some/all delay information is back-annotated.			Startpoint: u_CNN_TOP_CNN_CORE_CTL_inst_cnn_ready_reg_reg_2_ (rising edge-triggered flip-flop clocked by clk_io)		
Operating Conditions: SP018WP_V1p1_max Library: SP018WP_V1p1_max			Endpoint: u_CNN_TOP_CNN_CORE_CTL_inst_cnn_ready_reg_reg_3_ (rising edge-triggered flip-flop clocked by clk_io)		
Information: Percent of Arnoldi-based delays = 13.57%			Path Group: clk_io		
Startpoint: data_in_io[7] (input port clocked by clk_io)			Path Type: min		
Endpoint: u_CNN_TOP_WEIGHT_inst_filter_REG_reg_2_8_7_ (rising edge-triggered flip-flop clocked by clk_io)					
Path Group: INPUTS					
Path Type: min					
Point	Incr	Path	Point	Incr	Path
clock clk_io (rise edge)	0.00	0.00	clock clk_io (rise edge)	0.00	0.00
clock network delay (propagated)	0.00	0.00	clock network delay (propagated)	1.06	1.06
input external delay	0.83	0.83 r	u_CNN_TOP_CNN_CORE_CTL_inst_cnn_ready_reg_reg_2_/CK (DQUHDV0P7)	0.00	1.06 r
data_in_io[7] (in)	0.00	0.83 r	u_CNN_TOP_CNN_CORE_CTL_inst_cnn_ready_reg_reg_2_/Q (DQUHDV0P7)	0.19	1.25 f
I0_PAD10/C (PIW)	0.00	1.63 r	u_CNN_TOP_CNN_CORE_CTL_inst_cnn_ready_reg_reg_3_/D (DQUHDV0P7)	0.00	1.25 f
U10798/ZN (NAND2UHDV0P4)	0.07	1.70 f	data arrival time		1.25
U10801/ZN (NAND3UHDV0P4)	0.07	1.77 r	clock clk_io (rise edge)	0.00	0.00
u_CNN_TOP_WEIGHT_inst_filter_REG_reg_2_8_7_/D (DRQUHDV0P7)	0.00	1.77 r	clock network delay (propagated)	1.06	1.06
data arrival time		1.77	clock uncertainty	0.03	1.10
clock clk_io (rise edge)	0.00	0.00	u_CNN_TOP_CNN_CORE_CTL_inst_cnn_ready_reg_reg_3_/CK (DQUHDV0P7)	0.00	1.10 r
clock network delay (propagated)	1.05	1.05	library hold time		1.11
clock uncertainty	0.03	1.08	data required time		1.11
u_CNN_TOP_WEIGHT_inst_filter_REG_reg_2_8_7_/CK (DRQUHDV0P7)	0.00	1.08 r	data required time		1.11
library hold time	-0.02	1.06	data arrival time		-1.25
data required time		1.06	slack (MET)		0.14
data arrival time		-1.77			
slack (MET)		0.71			

图.最终的关键路径保持时间时序情况

DRC、LVS 检查情况:

Verify Summary:

Total number of nets = 14631, of which 0 are not extracted
Total number of open nets = 0, of which 0 are frozen
Total number of excluded ports = 0 ports of 0 unplaced cells connected to 0 nets
0 ports without pins of 0 cells connected to 0 nets
0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 0
Total number of antenna violations = no antenna rules defined
Total number of voltage-area violations = no voltage-areas defined
Total number of tie to rail violations = not checked
Total number of tie to rail directly violations = not checked

Memory usage for zrouter task 11403 Mbytes -- main task 5772 Mbytes.
Router separate process finished successfully.

```
-- LVS START : --
Total area error in layer 0 is 0. Elapsed = 0:00:00, CPU = 0:00:00
Total area error in layer 1 is 0. Elapsed = 0:00:01, CPU = 0:00:01
Total area error in layer 2 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 3 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 4 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 5 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 6 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 7 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 8 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 9 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 10 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 11 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 12 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 13 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 14 is 0. Elapsed = 0:00:02, CPU = 0:00:02
Total area error in layer 15 is 0. Elapsed = 0:00:02, CPU = 0:00:02
ERROR : OUTPUT PortInst U9130 S doesn't connect to any net.
ERROR : OUTPUT PortInst U7670 S doesn't connect to any net.
ERROR : OUTPUT PortInst U7510 S doesn't connect to any net.

** Total Floating ports are 3.
** Total Floating Nets are 0.
** Total SHORT Nets are 0.
** Total OPEN Nets are 0.
** Total Electrical Equivalent Error are 0.
** Total Must Joint Error are 0.

-- LVS END : --
Elapsed = 0:00:02, CPU = 0:00:02
Update error cell ...
1
```

左图.DRC 报告 右图.LVS 报告

DRC 实现了清零、LVS 汇报了三个浮动端口，但不影响最终设计。

电源网络情况：

```
Name of design : ic_chip
Number of cell instances in the design : 15777
Number of cell instance masters in the library : 135

Processing power net VSS ...
Number of power net wires : 137
Number of vias : 6351
Average power dissipation in ic_chip : 300.00 mW
Power supply voltage : 1.80 V
Assigning vias begins
Assigning power net wires ...
Reading virtual power pads
Extracting power net VSS ...
Number of power pad nodes connecting to power net VSS : 20
Number of power pad nodes reaching to the power ports of the leaf cells or blocks : 20
Number of resistors in VSS : 10567
Number of nodes in VSS : 8591
Performing network simulation
Assigning netlists to simulation engine begins
Assigning netlists to simulation engine finished
Maximum IR drop in ic_chip : 33.17 mV
Maximum current in ic_chip : 41.467 mA

Processing power net VDD ...
Number of power net wires : 135
Number of vias : 6355
Average power dissipation in ic_chip : 300.00 mW
Power supply voltage : 1.80 V
Assigning vias begins
Assigning power net wires ...
Reading virtual power pads
Extracting power net VDD ...
Number of power pad nodes connecting to power net VDD : 22
Number of power pad nodes reaching to the power ports of the leaf cells or blocks : 22
Number of resistors in VDD : 10616
Number of nodes in VDD : 8599
Performing network simulation
Assigning netlists to simulation engine begins
Assigning netlists to simulation engine finished
Maximum IR drop in ic_chip : 33.37 mV
Maximum current in ic_chip : 74.078 mA
Overall takes 0.26 seconds
```

图.电源网络分析报告

从电源网络分析报告中可以得知，芯片最大电压降为 33.37mW，远低于要求的 180mW。

后仿真结果：

```
Command: /public/NEWS4/home/digital_team11_3/work/ai_chip_MIN_all_pass/sim/./simv +sdfverbose +v2k +define=SDF +vpdfile=./vpd/vcs_post_max.vpd -a ./log/vcs_post_max.log +memcbk +neg_tchk
Chronologic VCS simulator copyright 1991-2018
Contains Synopsys proprietary information.
Compiler version 0-2018.09-SP2_Full64; Runtime version 0-2018.09-SP2_Full64; Dec 27 10:20 2024
Doing SDF annotation ..... Done
Message: From $vcdpluson at time 0 in file ./testbench/tb_chip.v line 163: [VCD+-SVFN]:
Setting VPD File by "+vpdfile=" switch to ./vpd/vcs_post_max.vpd.

VCD+ Writer 0-2018.09-SP2_Full64 Copyright (c) 1991-2018 by Synopsys Inc.
out 0 : out = 01111111, ture = 01111111, ***PASS***
out 1 : out = 00011010, ture = 00011010, ***PASS***
out 2 : out = 10111010, ture = 10111010, ***PASS***
out 3 : out = 11100011, ture = 11100011, ***PASS***
out 4 : out = 01001001, ture = 01001001, ***PASS***
out 5 : out = 00101010, ture = 00101010, ***PASS***
out 6 : out = 10011001, ture = 10011001, ***PASS***
out 7 : out = 10000000, ture = 10000000, ***PASS***
out 8 : out = 01000101, ture = 01000101, ***PASS***
out 9 : out = 11000001, ture = 11000001, ***PASS***
out 10 : out = 01001001, ture = 01001001, ***PASS***

Command: /public/NEWS4/home/digital_team11_3/work/ai_chip_100M_all_pass/sim/./simv +sdfverbose +v2k +define=SDF +vpdfile=./vpd/vcs_post_min.vpd -a ./log/vcs_post_min.log +memcbk +neg_tchk
Chronologic VCS simulator copyright 1991-2018
Contains Synopsys proprietary information.
Compiler version 0-2018.09-SP2_Full64; Runtime version 0-2018.09-SP2_Full64; Dec 26 17:40 2024
Doing SDF annotation ..... Done
Message: From $vcdpluson at time 0 in file ./testbench/tb_chip.v line 163: [VCD+-SVFN]:
Setting VPD File by "+vpdfile=" switch to ./vpd/vcs_post_min.vpd.

VCD+ Writer 0-2018.09-SP2_Full64 Copyright (c) 1991-2018 by Synopsys Inc.
out 0 : out = 01111111, ture = 01111111, ***PASS***
out 1 : out = 01111111, ture = 01111111, ***PASS***
out 2 : out = 10000000, ture = 10000000, ***PASS***
out 3 : out = 10100100, ture = 10100100, ***PASS***
out 4 : out = 00000011, ture = 00000011, ***PASS***
out 5 : out = 00010101, ture = 00010101, ***PASS***
out 6 : out = 00001100, ture = 00001100, ***PASS***
out 7 : out = 00011010, ture = 00011010, ***PASS***
out 8 : out = 00001111, ture = 00001111, ***PASS***
out 9 : out = 00010001, ture = 00010001, ***PASS***
```

图.后仿真报告

我们的设计最终在 294MHz（周期 3.4ns）的时钟频率下通过后仿真，远高于题目要求的 50MHz。

十、实验结论：

经过一流程的后端设计，我们得出以下结论：

1.经过后端的布局、布线、时钟树综合，关键路径的时序可能不收敛

我们的设计在 DC 综合完成后的建立时间裕量为 0，后端设计中，由于加入了布局布线的物理信息，连线的物理延迟，最终的建立时间裕量小于 0 是合情合理的。

2.在布局阶段，优化阻塞和优化时序是矛盾的

在布局过程中，优化阻塞和优化时序往往存在冲突。同一模块的标准单元放置得越近可以降低延迟，但过于紧密的布局会导致布线拥塞和空隙不足的问题

3.在后端设计，适当增加核心面积，也许会有更好的时序结果与阻塞情况

在后端设计中，适当增加核心面积可能改善时序结果，同时缓解布局阻塞问题。

4.软件只会按照给的约束条件尽力优化设计，而不会自行过约束

软件仅按照用户提供的约束条件优化设计。如果目标频率约束低于实际设计能力（如设置为 50MHz，而设计可达 100MHz），后端优化结果也只能满足较低的约束目标。

5.后端设计是一个环环相扣的系统的流程，任意一个流程没有做好都会导致设计结果出现不收敛情况。

比如一开始我们没有正确设置电源网络，导致最后布线的时候 DRC、LVS 不收敛，有几十几百万个 DRC 报错。在时钟树综合中错误的约束，会导致综合出来的设计完全不可用，裕量为-20ns。

十一、总结及心得体会：

后端设计中需要综合考虑时序、面积、功耗等多方面因素，远比书本中的设计理论复杂。每个决策都需要权衡不同约束条件，实践中更注重细节的优化。

后端设计流程的核心目的之一是保障芯片的流片成功率。每一步设计都需要经过严格验证，以确保逻辑与物理实现的一致性，并最大限度地避免制造问

题。因此，细致与严谨在后端设计中十分重要，每一行脚本都有他的作用，都不可或缺。

十二、对本实验过程及方法、手段的改进建议：

- 1.希望老师可以再多讲讲 ICC 得图形化界面的使用
- 2.希望多学一些后端软件，比如 Prime Time 等