CAD Design Project 1 – OpenROAD (Part I)

A.OpenROAD 安裝步驟:

1. OS 為 Windows 要先安裝 WSL，並安裝 Ubuntu。

   以系統管理員身分開啟終端機，

   第一次使用 WSL:

   ```
   wsl --install
   ```

   已使用過:

   ```
   wsl --install -d Ubuntu
   ```

2. OpenROAD 安裝

   git clone --recursive https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts

   相依性套件:

   cd ~/OpenRoad-flow-script/tools/OpenROAD/

   sudo ./etc/DependencyInstaller.sh -dev

   編譯:

   cd ~/OpenROAD-flow-scripts

   ./build_openroad.sh –local

   設定環境:

   source ./env.sh -dev

確認:

yosys -help

openroad -help

3. Klayout

wget https://www.klayout.org/downloads/Ubuntu-22/klayout_0.29.1-1_amd64.deb

klayout 相依性套件:

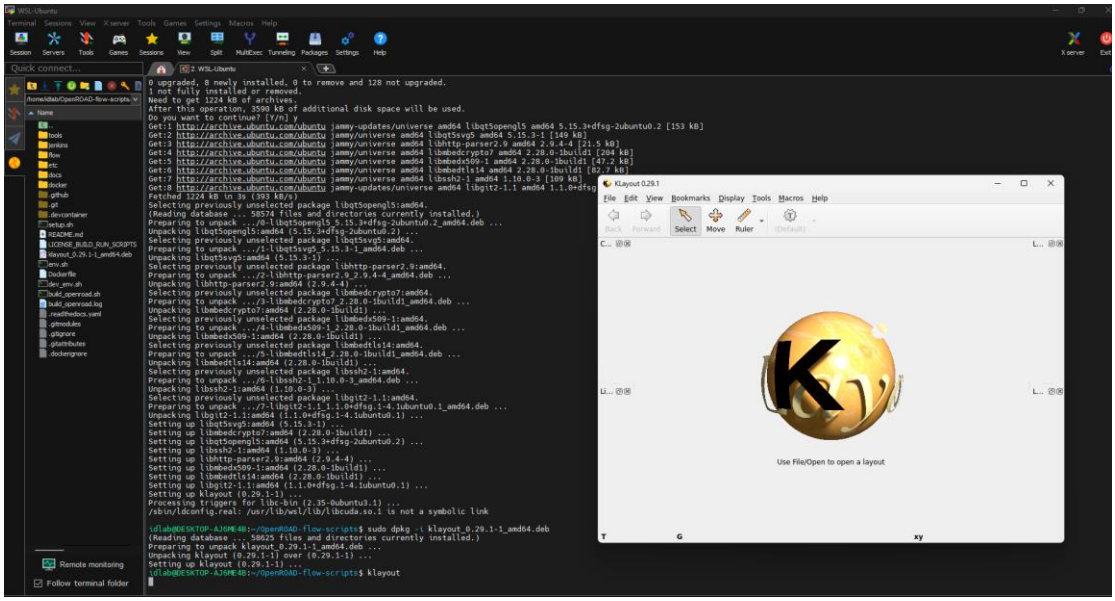
sudo apt install libqt5designer5 libqt5multimedia5

libqt5multimediawidgets5 libqt5xmlpatterns5 libruby3.0

安裝:

sudo apt --fix-broken install

sudo dpkg -i klayout_0.29.1-1_amd64.deb



B.IC Design Flow:

執行步驟

cd flow

make

執行默認的 ip:gcd

```
# Default design
DESIGN_CONFIG ?= ./designs/nangate45/gcd/config.mk
```

執行結果

```
cp results/nangate45/gcd/base/6_1_merged.gds results/nangate45
./logs/nangate45/gcd/base
Log                             Elapsed seconds  Peak Memory/MB
1_1_yosys                                     0              36
1_1_yosys_canonicalize                        0              30
1_1_yosys_hier_report                         0              11
2_1_floorplan                                 0             118
2_2_floorplan_io                              0             117
2_3_floorplan_tdms                            0             117
2_4_floorplan_macro                           0             117
2_5_floorplan_tapcell                         0             115
2_6_floorplan_pdn                             0             119
3_1_place_gp_skip_io                          0             116
3_2_place_iop                                 0             116
3_3_place_gp                                  1             322
3_4_place_resized                             0             320
3_5_place_dp                                  0             119
4_1_cts                                       3             409
5_1_grt                                       1             416
5_2_fillcell                                  0             118
5_3_route                                     4             865
6_1_merge                                     0             407
6_report                                      1             256
Total                                        10             865
```

內部完成 file

ls -al

```
idlab@DESKTOP-AJ6ME4B:~/OpenROAD-flow-scripts/flow$ ls -al
total 100
drwxr-xr-x 12 idlab idlab  4096 Sep  6 20:30 .
drwxr-xr-x 11 idlab idlab  4096 Sep  4 23:27 ..
-rw-r--r--  1 idlab idlab    60 Sep  4 18:48 .gitignore
-rw-r--r--  1 idlab idlab 45213 Sep  6 18:22 Makefile
drwxr-xr-x 15 idlab idlab  4096 Sep  4 18:48 designs
drwxr-xr-x  3 idlab idlab  4096 Sep  6 20:30 logs
drwxr-xr-x  3 idlab idlab  4096 Sep  6 20:30 objects
drwxr-xr-x 12 idlab idlab  4096 Sep  4 18:48 platforms
drwxr-xr-x  3 idlab idlab  4096 Sep  6 20:30 reports
drwxr-xr-x  3 idlab idlab  4096 Sep  6 20:30 results
drwxr-xr-x  2 idlab idlab  4096 Sep  4 18:48 scripts
drwxr-xr-x  2 idlab idlab  4096 Sep  4 18:48 test
drwxr-xr-x  3 idlab idlab  4096 Sep  4 18:48 tutorials
drwxr-xr-x  3 idlab idlab  4096 Sep  4 18:48 util
```

result 內部會有每個 stage 的結果

cd result/

cd nangate45/

cd gcd/

cd base/

ls

```
idlab@DESKTOP-AJ6ME4B:~/OpenROAD-flow-scripts/flow/results/nangate45/gcd/base$ ls
1_1_yosys.v          2_2_floorplan_io.odb       2_floorplan.odb           3_4_place_resized.odb
1_synth.rtlil        2_3_floorplan_tdms.odb     2_floorplan.sdc           3_5_place_dp.odb
1_synth.sdc          2_4_floorplan_macro.odb    3_1_place_gp_skip_io.odb  3_place.odb
1_synth.v            2_5_floorplan_tapcell.odb  3_2_place_iop.odb         3_place.sdc
2_1_floorplan.odb    2_6_floorplan_pdn.odb      3_3_place_gp.odb          4_1_cts.odb
```

```
4_cts.odb          5_route.odb      6_final.def   6_final.v           updated_clks.sdc
4_cts.sdc          5_route.sdc      6_final.gds   clock_period.txt
5_1_grt.odb        6_1_fill.odb     6_final.odb   keep_hierarchy.tcl
5_2_fillcell.odb   6_1_fill.sdc     6_final.sdc   mem.json
5_3_route.odb      6_1_merged.gds   6_final.spef  route.guide
```

designs 內部會有用來執行的 config.mk file，紀錄設計相關的設定值

cd design

cd nangate45/gcd

nano config.mk

```
idlab@DESKTOP-AJ6ME4B:~/OpenROAD-flow-scripts/flow/designs/nangate45/gcd$ ls
config.mk  constraint.sdc  metadata-base-ok.json  rules-base.json
```

```
  GNU nano 6.2
export DESIGN_NAME = gcd
export PLATFORM    = nangate45

export VERILOG_FILES = ./designs/src/$(DESIGN_NAME)/gcd.v
export SDC_FILE      = ./designs/$(PLATFORM)/$(DESIGN_NAME)/constraint.sdc
export ABC_AREA      = 1

# Adders degrade GCD
export ADDER_MAP_FILE :=

export CORE_UTILIZATION ?= 55
export PLACE_DENSITY_LB_ADDON = 0.20
export TNS_END_PERCENT        = 100
export REMOVE_CELLS_FOR_EQY   = TAPCELL*
```

src 中會放 source code

```
idlab@DESKTOP-AJ6ME4B:~/OpenROAD-flow-scripts/flow/designs/src$ ls
aes       ariane     ariane136     bp_be_top  bp_multi_top  chameleon       coyote     dynamic_node  ethmac_lvt  gcd
aes_lvt   ariane133  black_parrot  bp_fe_top  bp_quad       chameleon_hier  coyote_tc  ethmac        fifo        harness
idlab@DESKTOP-AJ6ME4B:~/OpenROAD-flow-scripts/flow/designs/src$ cd gcd
idlab@DESKTOP-AJ6ME4B:~/OpenROAD-flow-scripts/flow/designs/src/gcd$ ls
README.md  gcd.v
```

script 存放個步驟的 tcl 檔，當 makefile 開始執行整個 flow，就會進來抓需

要的 tcl 檔

```
idlab@DESKTOP-AJ6ME4B:~/OpenROAD-flow-scripts/flow/designs/src/gcd$ cd ~/OpenROAD-flow-scripts/flow/scripts
idlab@DESKTOP-AJ6ME4B:~/OpenROAD-flow-scripts/flow/scripts$ ls
abc_area.script       deleteRoutingObstructions.tcl  floorplan.tcl            io_placement_random.tcl  noop.tcl
abc_speed.script      density_fill.tcl               generate_abstract.tcl    io_placement_util.tcl    pdn.tcl
add_routing_blk.tcl   detail_place.tcl               global_place.tcl         klayout.tcl              placement_blockages.tcl
cdl.tcl               detail_route.tcl               global_place_skip_io.tcl load.tcl                 read_liberty.tcl
cts.tcl               escape.sh                      global_route.tcl         macro_place.tcl          read_macro_placement.tcl
deleteNonClkNets.tcl  fillcell.tcl                   gui.tcl                  macro_place_util.tcl     report_metrics.tcl
deletePowerNets.tcl   final_report.tcl               io_placement.tcl         mem_dump.py              resize.tcl
```

```
save_images.tcl          synth_preamble.tcl  write_verilog.tcl
set_place_density.tcl    tapcell.tcl         yosys.tcl
sta-synth.tcl            tdms_place.tcl
synth.tcl                util.tcl
synth_canonicalize.tcl   view_cells.tcl
synth_hier_report.tcl    write_def.tcl
synth_metrics.tcl        write_ref_sdc.tcl
```

6 major design-flow stages

Synthesis:

```
# ============================================================================
#  ____  _____ _   _ _____ ____ ___ ____
# / ___|| ____| \ | |_   _|  _ \_ _/ ___|
# \___ \|  _| |  \| | | | | |_) | |\___ \
#  ___) | |___| |\  | | | |  _ <| | ___) |
# |____/|_____|_| \_| |_| |_| \_\___|____/
#
.PHONY: synth
synth: $(RESULTS_DIR)/1_synth.v \
       $(RESULTS_DIR)/1_synth.sdc

.PHONY: synth-report
synth-report: synth
	$(UNSET_AND_MAKE) do-synth-report

.PHONY: do-synth-report
do-synth-report:
	($(TIME_CMD) $(OPENROAD_CMD) $(SCRIPTS_DIR)/synth_metrics.tcl) 2>&1 | tee $(LOG_DIR)/1_1_yosys_metrics.log

.PHONY: memory
memory:
	python3 $(SCRIPTS_DIR)/mem_dump.py $(RESULTS_DIR)/mem.json

# ============================================================================


# Run Synthesis using yosys
#-------------------------------------------------------------------------------
```

使用 yosys 進行 Synthesis

將 RTL code 的部分轉換為 Gate level

```
module gcd
(
  input  wire clk,
  input  wire [  31:0] req_msg,
  output wire req_rdy,
  input  wire req_val,
  input  wire reset,
  output wire [  15:0] resp_msg,
  input  wire resp_rdy,
  output wire resp_val
);

  // ctrl temporaries
  wire   [   0:0] ctrl$is_b_zero;
  wire   [   0:0] ctrl$resp_rdy;
  wire   [   0:0] ctrl$clk;
  wire   [   0:0] ctrl$is_a_lt_b;
  wire   [   0:0] ctrl$req_val;
  wire   [   0:0] ctrl$reset;
  wire   [   1:0] ctrl$a_mux_sel;
  wire   [   0:0] ctrl$resp_val;
  wire   [   0:0] ctrl$b_mux_sel;
  wire   [   0:0] ctrl$b_reg_en;
  wire   [   0:0] ctrl$a_reg_en;
  wire   [   0:0] ctrl$req_rdy;

  GcdUnitCtrlRTL_0x4d0fc71ead8d3d9e ctrl
  (
    .is_b_zero ( ctrl$is_b_zero ),
    .resp_rdy  ( ctrl$resp_rdy ),
    .clk       ( ctrl$clk ),
    .is_a_lt_b ( ctrl$is_a_lt_b ),
    .req_val   ( ctrl$req_val ),
    .reset     ( ctrl$reset ),
    .a_mux_sel ( ctrl$a_mux_sel ),
    .resp_val  ( ctrl$resp_val ),
    .b_mux_sel ( ctrl$b_mux_sel ),
    .b_reg_en  ( ctrl$b_reg_en ),
    .a_reg_en  ( ctrl$a_reg_en ),
    .req_rdy   ( ctrl$req_rdy )
  );
```

```
  // dpath temporaries
  wire   [   1:0] dpath$a_mux_sel;
  wire   [   0:0] dpath$clk;
  wire   [  15:0] dpath$req_msg_b;
  wire   [  15:0] dpath$req_msg_a;
  wire   [   0:0] dpath$b_mux_sel;
  wire   [   0:0] dpath$reset;
  wire   [   0:0] dpath$b_reg_en;
  wire   [   0:0] dpath$a_reg_en;
  wire   [   0:0] dpath$is_b_zero;
  wire   [  15:0] dpath$resp_msg;
  wire   [   0:0] dpath$is_a_lt_b;

  GcdUnitDpathRTL_0x4d0fc71ead8d3d9e dpath
  (
    .a_mux_sel ( dpath$a_mux_sel ),
    .clk       ( dpath$clk ),
    .req_msg_b ( dpath$req_msg_b ),
    .req_msg_a ( dpath$req_msg_a ),
    .b_mux_sel ( dpath$b_mux_sel ),
    .reset     ( dpath$reset ),
    .b_reg_en  ( dpath$b_reg_en ),
    .a_reg_en  ( dpath$a_reg_en ),
    .is_b_zero ( dpath$is_b_zero ),
    .resp_msg  ( dpath$resp_msg ),
    .is_a_lt_b ( dpath$is_a_lt_b )
  );

  // signal connections
  assign ctrl$clk        = clk;
  assign ctrl$is_a_lt_b  = dpath$is_a_lt_b;
  assign ctrl$is_b_zero  = dpath$is_b_zero;
  assign ctrl$req_val    = req_val;
  assign ctrl$reset      = reset;
  assign ctrl$resp_rdy   = resp_rdy;
  assign dpath$a_mux_sel = ctrl$a_mux_sel;
  assign dpath$a_reg_en  = ctrl$a_reg_en;
  assign dpath$b_mux_sel = ctrl$b_mux_sel;
  assign dpath$b_reg_en  = ctrl$b_reg_en;
  assign dpath$clk       = clk;
  assign dpath$req_msg_a = req_msg[31:16];
  assign dpath$req_msg_b = req_msg[15:0];
  assign dpath$reset     = reset;
  assign req_rdy         = ctrl$req_rdy;
  assign resp_msg        = dpath$resp_msg;
  assign resp_val        = ctrl$resp_val;


endmodule // GcdUnit
```

把 gcd.v 轉為 1_synth.v

```
/* Generated by Yosys 0.44 (git sha1 80ba43d26, clang++ 14.0.0-1ubuntu1.1 -fPIC -O3) */

(* src = "./designs/src/gcd/gcd.v:11.1-102.10" *)
(* top =  1  *)
(* hdlname = "gcd" *)
module gcd(clk, req_msg, req_rdy, req_val, reset, resp_msg, resp_rdy, resp_val);
  wire _000_;
  wire _001_;
  wire _002_;
  wire _003_;
  wire _004_;
  wire _005_;
  wire _006_;
```

```
  INV_X1 _486_ (
    .A(\dpath.a_lt_b$in1[11] ),
    .ZN(_036_)
  );
  INV_X1 _487_ (
    .A(\dpath.a_lt_b$in1[10] ),
    .ZN(_037_)
  );
  INV_X2 _488_ (
    .A(\dpath.a_lt_b$in1[9] ),
    .ZN(_038_)
  );
  INV_X2 _489_ (
    .A(\dpath.a_lt_b$in1[8] ),
    .ZN(_039_)
  );
  NAND4_X1 _490_ (
    .A1(_036_),
    .A2(_037_),
    .A3(_038_),
    .A4(_039_),
    .ZN(_040_)
  );
  INV_X1 _491_ (
    .A(\dpath.a_lt_b$in1[15] ),
    .ZN(_041_)
  );
  INV_X2 _492_ (
    .A(\dpath.a_lt_b$in1[14] ),
    .ZN(_042_)
  );
  INV_X1 _493_ (
    .A(\dpath.a_lt_b$in1[13] ),
    .ZN(_043_)
  );
  INV_X2 _494_ (
    .A(\dpath.a_lt_b$in1[12] ),
    .ZN(_044_)
  );
  NAND4_X1 _495_ (
    .A1(_041_),
    .A2(_042_),
    .A3(_043_),
    .A4(_044_),
    .ZN(_045_)
  );
  INV_X2 _496_ (
    .A(\dpath.a_lt_b$in1[3] ),
    .ZN(_046_)
  );
```

```
  INV_X2 _497_ (
    .A(\dpath.a_lt_b$in1[2] ),
    .ZN(_047_)
  );
  INV_X2 _498_ (
    .A(\dpath.a_lt_b$in1[1] ),
    .ZN(_048_)
  );
  INV_X2 _499_ (
    .A(\dpath.a_lt_b$in1[0] ),
    .ZN(_049_)
  );
  NAND4_X1 _500_ (
    .A1(_046_),
    .A2(_047_),
    .A3(_048_),
    .A4(_049_),
    .ZN(_050_)
  );
  INV_X2 _501_ (
    .A(\dpath.a_lt_b$in1[7] ),
    .ZN(_051_)
  );
  INV_X2 _502_ (
    .A(\dpath.a_lt_b$in1[6] ),
    .ZN(_052_)
  );
  INV_X2 _503_ (
    .A(\dpath.a_lt_b$in1[5] ),
    .ZN(_053_)
  );
  INV_X1 _504_ (
    .A(\dpath.a_lt_b$in1[4] ),
    .ZN(_054_)
  );
  NAND4_X1 _505_ (
    .A1(_051_),
    .A2(_052_),
    .A3(_053_),
    .A4(_054_),
    .ZN(_055_)
  );
  NOR4_X1 _506_ (
    .A1(_040_),
    .A2(_045_),
    .A3(_050_),
    .A4(_055_),
    .ZN(_056_)
  );
  BUF_X1 _507_ (
    .A(\ctrl.state.out[2] ),
    .Z(_057_)
  );
```

合成後的結果

```
1_synth.rtlil
1_synth.sdc
1_synth.v
```

Floorplan:

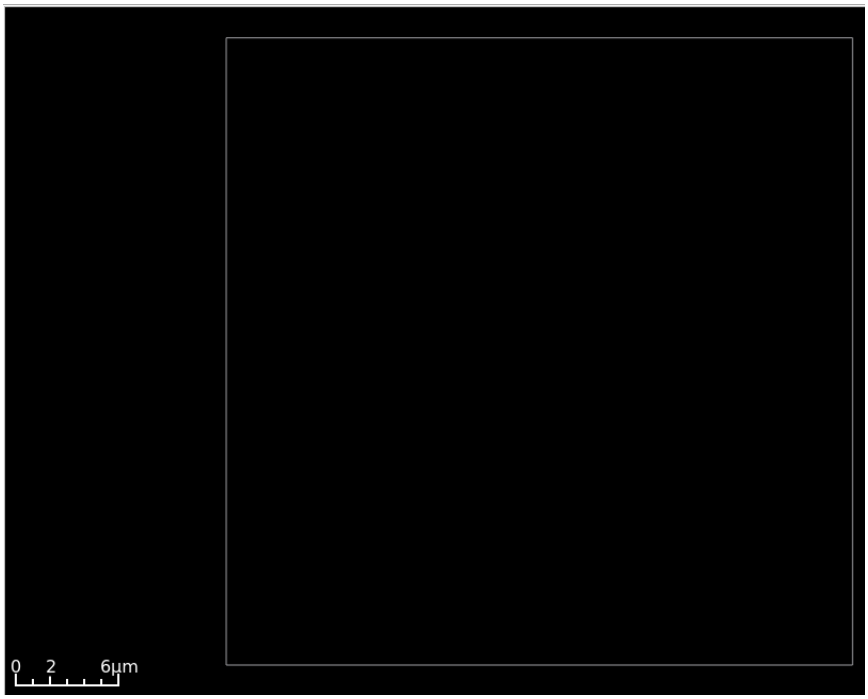
```
# ====================================================================
#
#  _____ _                      ____  _             _
# |  ___| | ___   ___  _ __    |  _ \| | __ _ _ __ | |
# | |_  | |/ _ \ / _ \| '__|   | |_) | |/ _` | '_ \| |
# |  _| | | (_) | (_) | |      |  __/| | (_| | | | |_|
# |_|   |_|\___/ \___/|_|      |_|   |_|\__,_|_| |_(_)
#
.PHONY: floorplan
floorplan: $(RESULTS_DIR)/2_floorplan.odb \
           $(RESULTS_DIR)/2_floorplan.sdc

# ====================================================================
```

```
# STEP 1: Translate verilog to odb
#-------------------------------------------------------------------------------
$(eval $(call do-step,2_1_floorplan,$(RESULTS_DIR)/1_synth.v $(RESULTS_DIR)/1_synth.sdc $(TECH_LEF) $(SC_LEF) $(ADDITIONAL_LEFS) $(FOOTPRINT) $(SIG_MAP_FILE) $(FOOTPRINT_TCL),floorplan))

# STEP 2: IO Placement (random)
#-------------------------------------------------------------------------------
ifndef IS_CHIP
$(eval $(call do-step,2_2_floorplan_io,$(RESULTS_DIR)/2_1_floorplan.odb $(IO_CONSTRAINTS),io_placement_random))
else
$(eval $(call do-copy,2_2_floorplan_io,2_1_floorplan.odb,$(IO_CONSTRAINTS)))
endif

# STEP 3: Timing Driven Mixed Sized Placement
#-------------------------------------------------------------------------------
ifneq ($(MACRO_PLACEMENT)$(MACRO_PLACEMENT_TCL),)
SKIP_TDMS:=1
endif
ifeq ($(RTLMP_FLOW), 1)
SKIP_TDMS:=1
endif
ifneq ($(SKIP_TDMS), 1)
$(eval $(call do-step,2_3_floorplan_tdms,$(RESULTS_DIR)/2_2_floorplan_io.odb $(RESULTS_DIR)/1_synth.v $(RESULTS_DIR)/1_synth.sdc $(LIB_FILES),tdms_place))
else
$(eval $(call do-copy,2_3_floorplan_tdms,2_2_floorplan_io.odb,$(RESULTS_DIR)/1_synth.v $(RESULTS_DIR)/1_synth.sdc $(LIB_FILES)))
endif

# STEP 4: Macro Placement
#-------------------------------------------------------------------------------
$(eval $(call do-step,2_4_floorplan_macro,$(RESULTS_DIR)/2_3_floorplan_tdms.odb $(RESULTS_DIR)/1_synth.v $(RESULTS_DIR)/1_synth.sdc $(MACRO_PLACEMENT) $(MACRO_PLACEMENT_TCL),macro_place))

# STEP 5: Tapcell and Welltie insertion
#-------------------------------------------------------------------------------
$(eval $(call do-step,2_5_floorplan_tapcell,$(RESULTS_DIR)/2_4_floorplan_macro.odb $(TAPCELL_TCL),tapcell))
```

```
# STEP 6: PDN generation
#-------------------------------------------------------------------------------
$(eval $(call do-step,2_6_floorplan_pdn,$(RESULTS_DIR)/2_5_floorplan_tapcell.odb $(PDN_TCL),pdn))

$(eval $(call do-copy,2_floorplan,2_6_floorplan_pdn.odb,))

$(RESULTS_DIR)/2_floorplan.sdc: $(RESULTS_DIR)/2_1_floorplan.odb

.PHONY: do-floorplan
do-floorplan:
	$(UNSET_AND_MAKE) do-2_1_floorplan do-2_2_floorplan_io do-2_3_floorplan_tdms do-2_4_floorplan_macro do-2_5_floorplan_tapcell do-2_6_floorplan_pdn do-2_floorplan

.PHONY: clean_floorplan
clean_floorplan:
	rm -f $(RESULTS_DIR)/2_*floorplan*.odb $(RESULTS_DIR)/2_floorplan.sdc $(RESULTS_DIR)/2_*.v $(RESULTS_DIR)/2_*.def
	rm -f $(REPORTS_DIR)/2_*
	rm -f $(LOG_DIR)/2_*
```

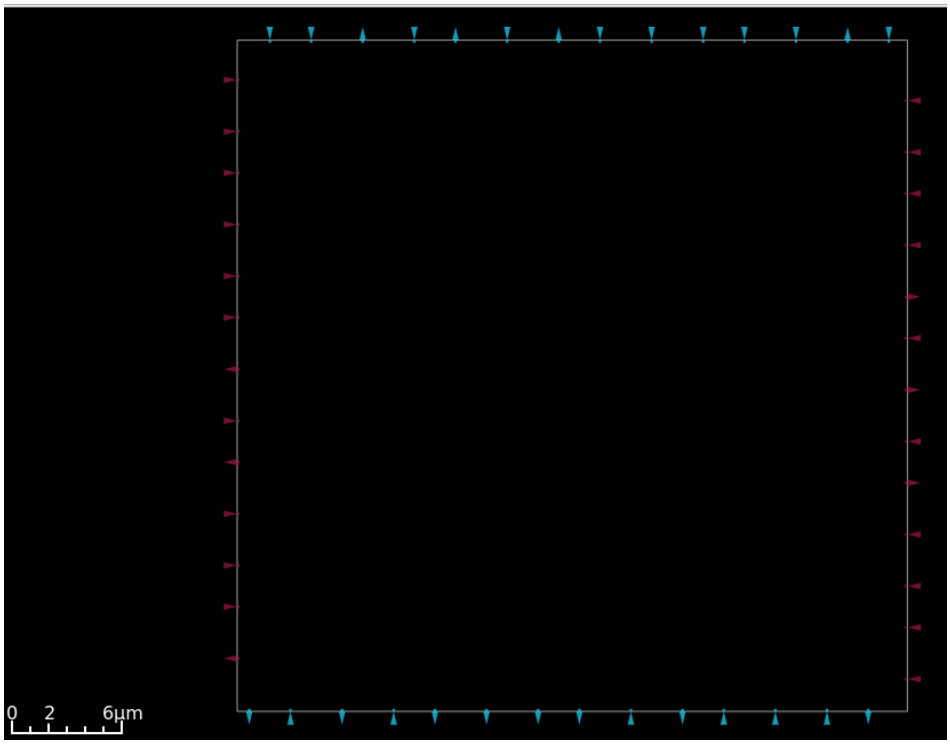
Step1 Translate Verilog to odb:

主要是把合成後的 verilog code 轉為 odb



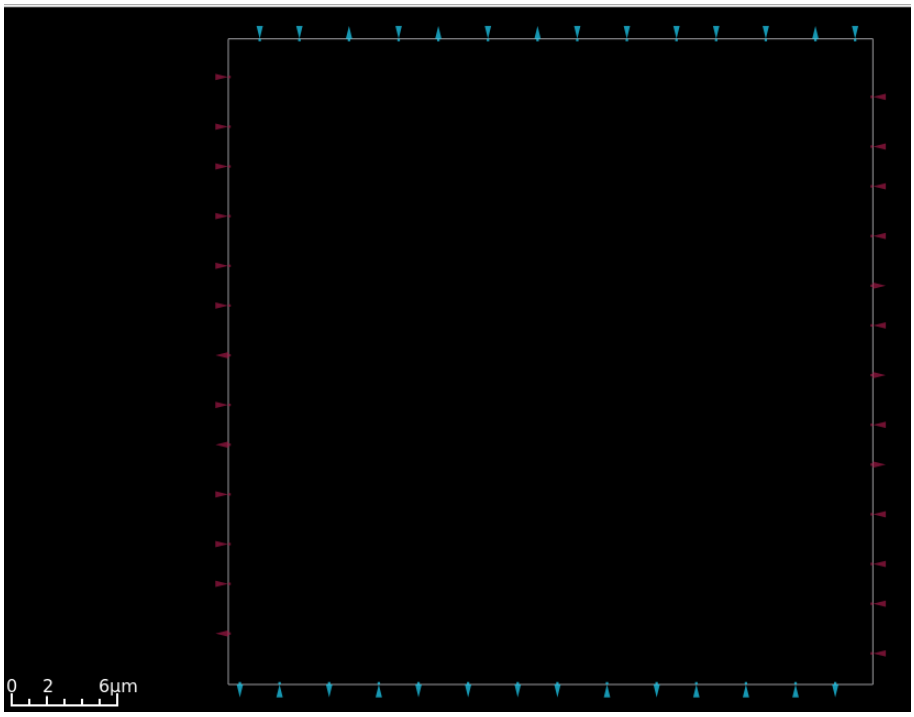
Step2 IO placement (random):

進行 IO 的擺放



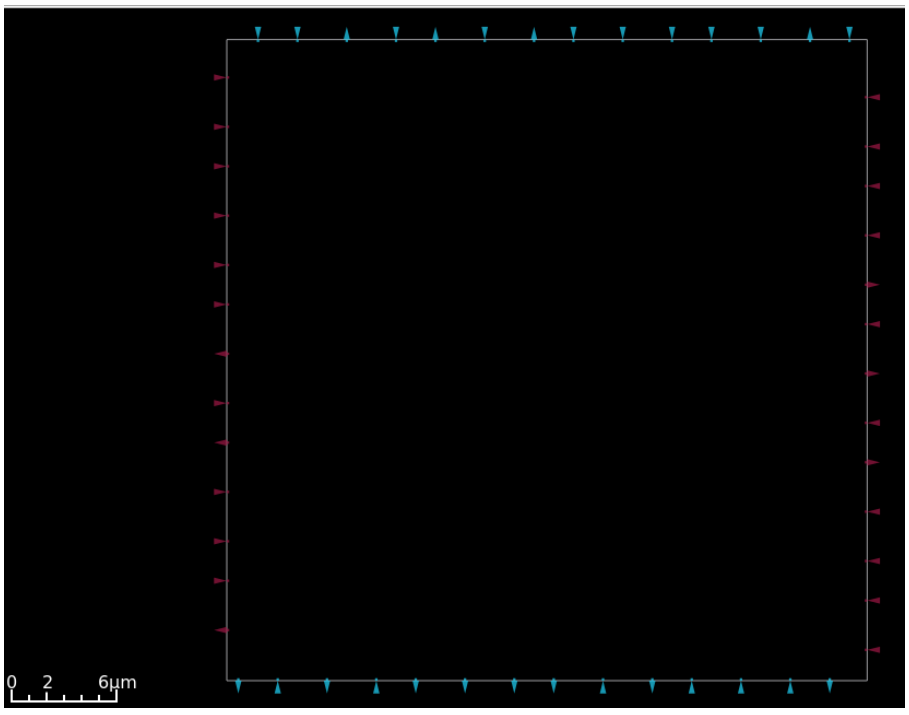
Step3 Timing Driven Mixed Sized Placement:

把 macro 跟 cell 放進去先放進去(gcd 中沒有 macro)



Step4 Macro Placement:

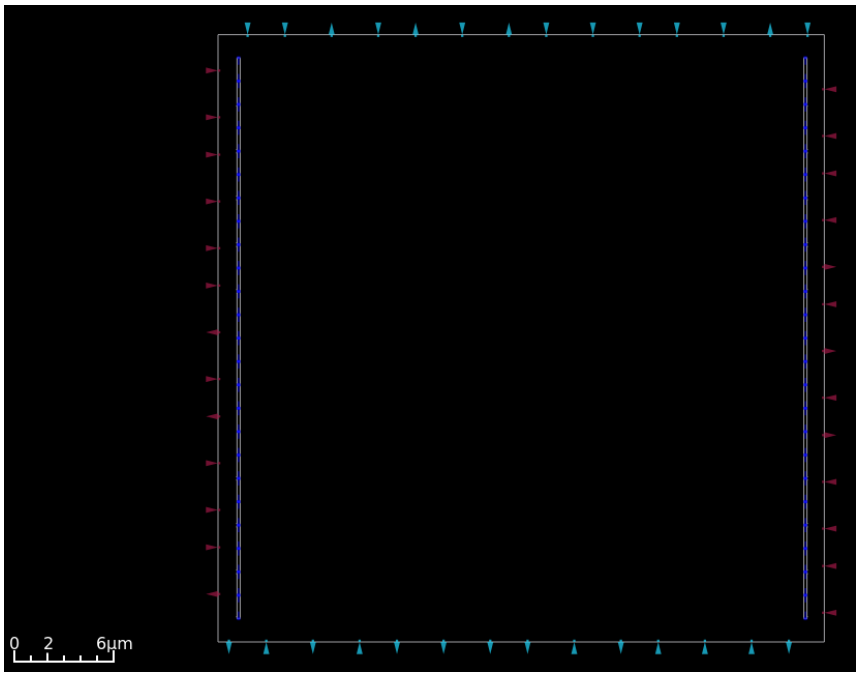
Macro 進行 placement



Step5 Tapcell and Welltie insertion:

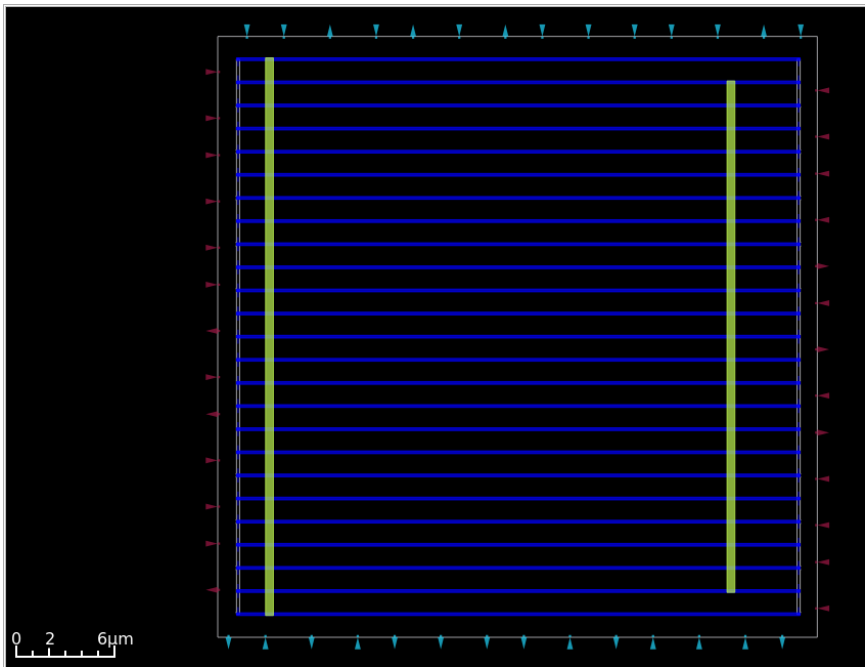
加入 Tapcell 跟 Welltie

是用來避免 Latch-up，讓 n-well 跟 p-well 正確接地或接電，控制漏電流



Step6 PDN generation:

佈 VDD 與 VSS 線

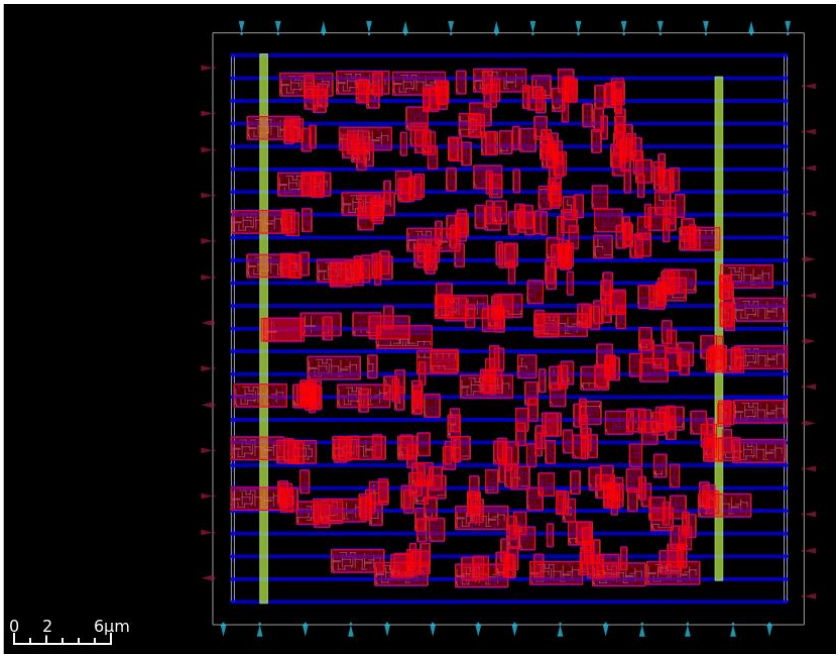


Placement:

```
#  ____  _____ ____  
# |  _ \| ____|  _ \ 
# | |_) |  _| | |_) |
# |  __/| |___|  _ < 
# |_|   |_____|_| \_\
#
.PHONY: place
place: $(RESULTS_DIR)/3_place.odb \
       $(RESULTS_DIR)/3_place.sdc
# ==========================================================================
# STEP 1: Global placement without placed IOs, timing-driven, and routability-driven.
#--------------------------------------------------------------------------
$(eval $(call do-step,3_1_place_gp_skip_io,$(RESULTS_DIR)/2_floorplan.odb $(RESULTS_DIR)/2_floorplan.sdc $(LIB_FILES),global_place_skip_io))

# STEP 2: IO placement (non-random)
#--------------------------------------------------------------------------
ifndef IS_CHIP
$(eval $(call do-step,3_2_place_iop,$(RESULTS_DIR)/3_1_place_gp_skip_io.odb $(IO_CONSTRAINTS),io_placement))
else
$(eval $(call do-copy,3_2_place_iop,3_1_place_gp_skip_io.odb,$(IO_CONSTRAINTS)))
endif

# STEP 3: Global placement with placed IOs, timing-driven, and routability-driven.
#--------------------------------------------------------------------------
$(eval $(call do-step,3_3_place_gp,$(RESULTS_DIR)/3_2_place_iop.odb $(RESULTS_DIR)/2_floorplan.sdc $(LIB_FILES),global_place))

# STEP 4: Resizing & Buffering
#--------------------------------------------------------------------------
$(eval $(call do-step,3_4_place_resized,$(RESULTS_DIR)/3_3_place_gp.odb $(RESULTS_DIR)/2_floorplan.sdc,resize))

.PHONY: clean_resize
clean_resize:
	rm -f $(RESULTS_DIR)/3_4_place_resized.odb

# STEP 5: Detail placement
#--------------------------------------------------------------------------
$(eval $(call do-step,3_5_place_dp,$(RESULTS_DIR)/3_4_place_resized.odb,detail_place))

$(eval $(call do-copy,3_place,3_5_place_dp.odb,))

$(eval $(call do-copy,3_place,2_floorplan.sdc,,.sdc))

.PHONY: do-place
do-place:
	$(UNSET_AND_MAKE) do-3_1_place_gp_skip_io do-3_2_place_iop do-3_3_place_gp do-3_4_place_resized do-3_5_place_dp do-3_place do-3_place.sdc

# Clean Targets
#--------------------------------------------------------------------------
.PHONY: clean_place
clean_place:
	rm -f $(RESULTS_DIR)/3_*place*.odb
	rm -f $(RESULTS_DIR)/3_place.sdc
	rm -f $(RESULTS_DIR)/3_*.def $(RESULTS_DIR)/3_*.v
```

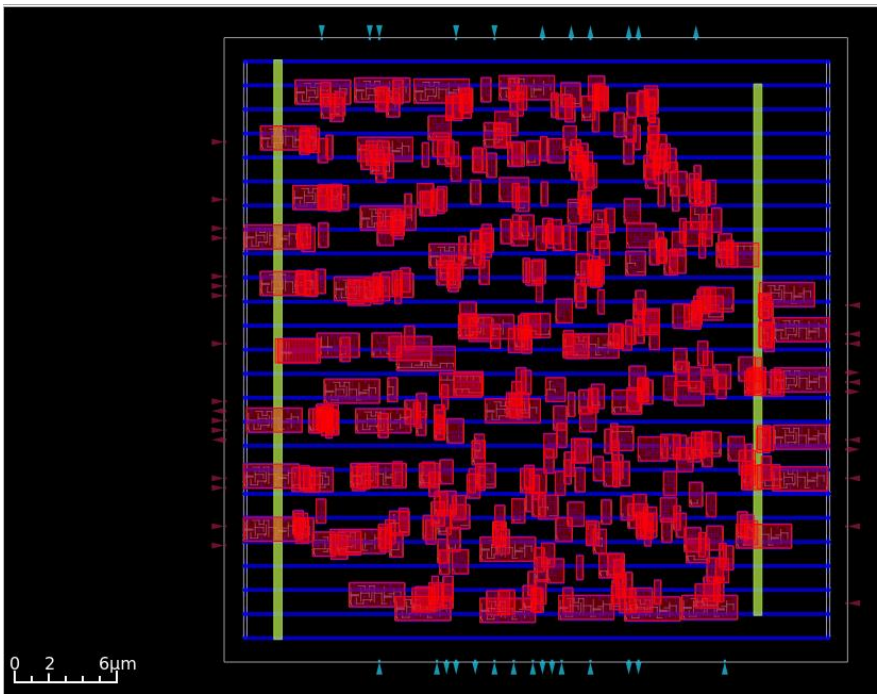
Step1 Global placement without placed IOs, timing-driven, and routability-driven:

沒有 IOs,timing,routing 的前提下對 instance 進行 global placement



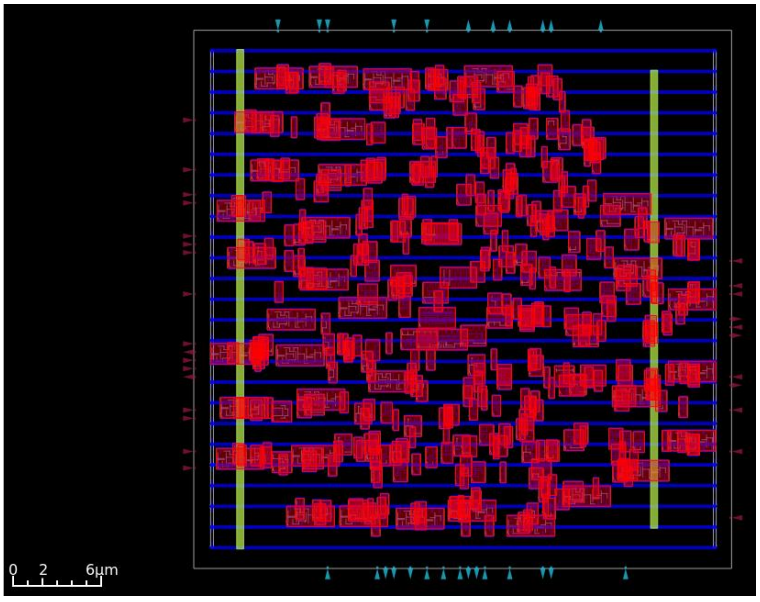
Step2 IO placement (non-random):

進行 IO placement



Step3 Global placement with placed IOs, timing-driven, and routability-driven:

完成 IO 之後，進行加上 routing,timing 再去做 placement

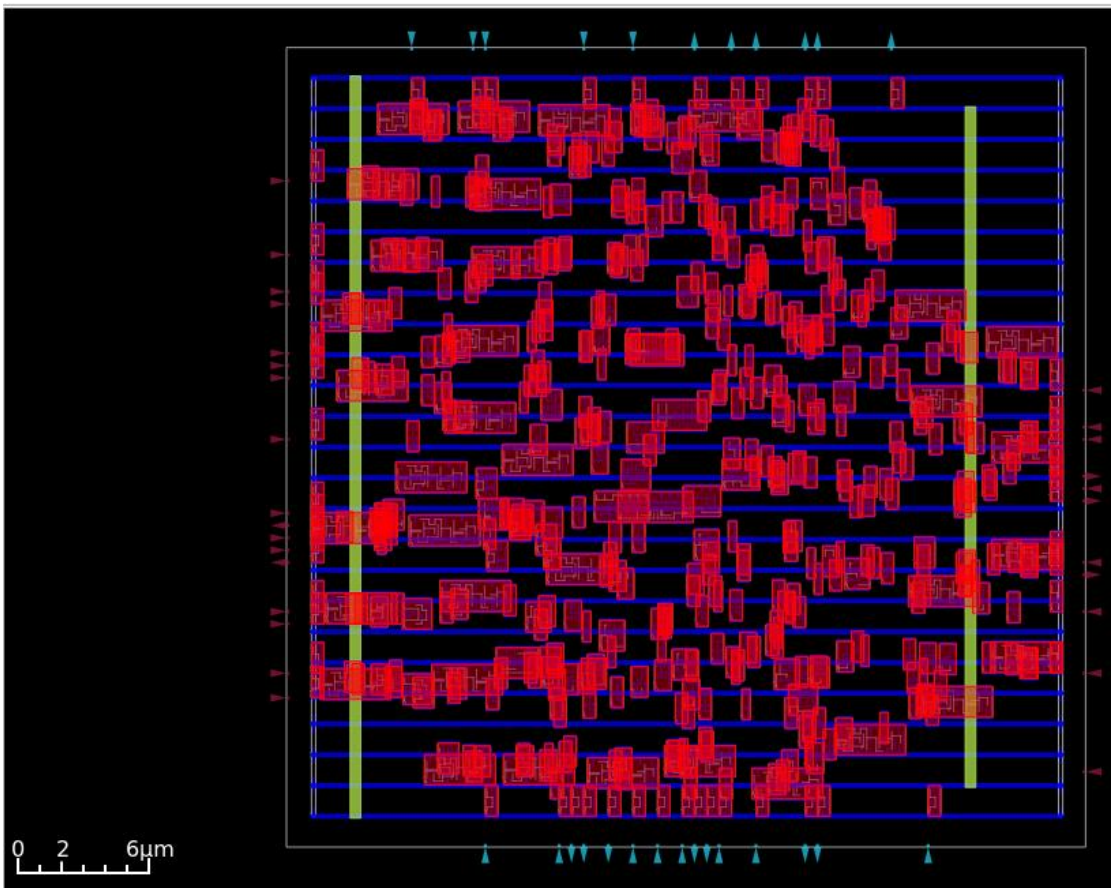


Step4 Resizing & Buffering:

進行 Resizing 和 Buffering

Resizing 讓電路滿足 timing constraints 提升效能

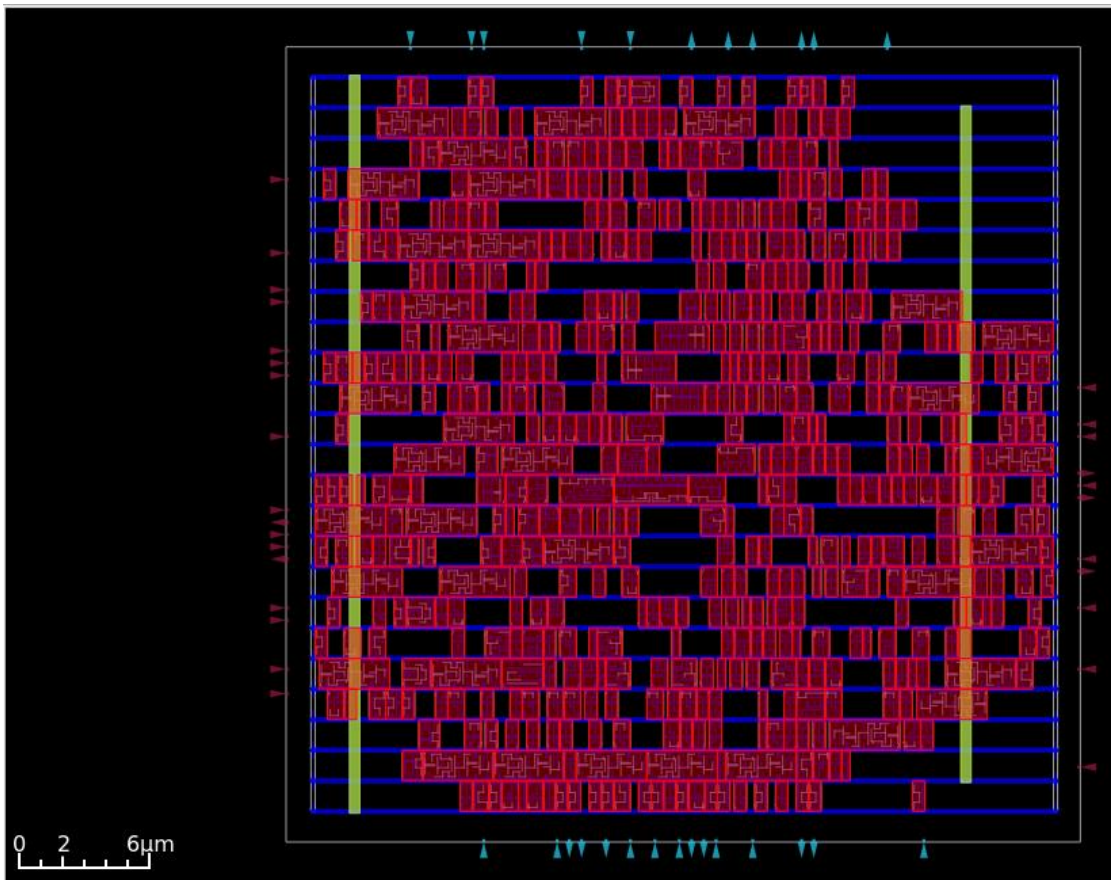
Buffering 是加入 buffer gates，透過減少電容或電阻等元件來改善 signal integrity



Step5 Detail placement:

進行 legalization 與 detail placement 使電路符合 constrains 並漸少 HPWL

(half-perimeter wirelength)

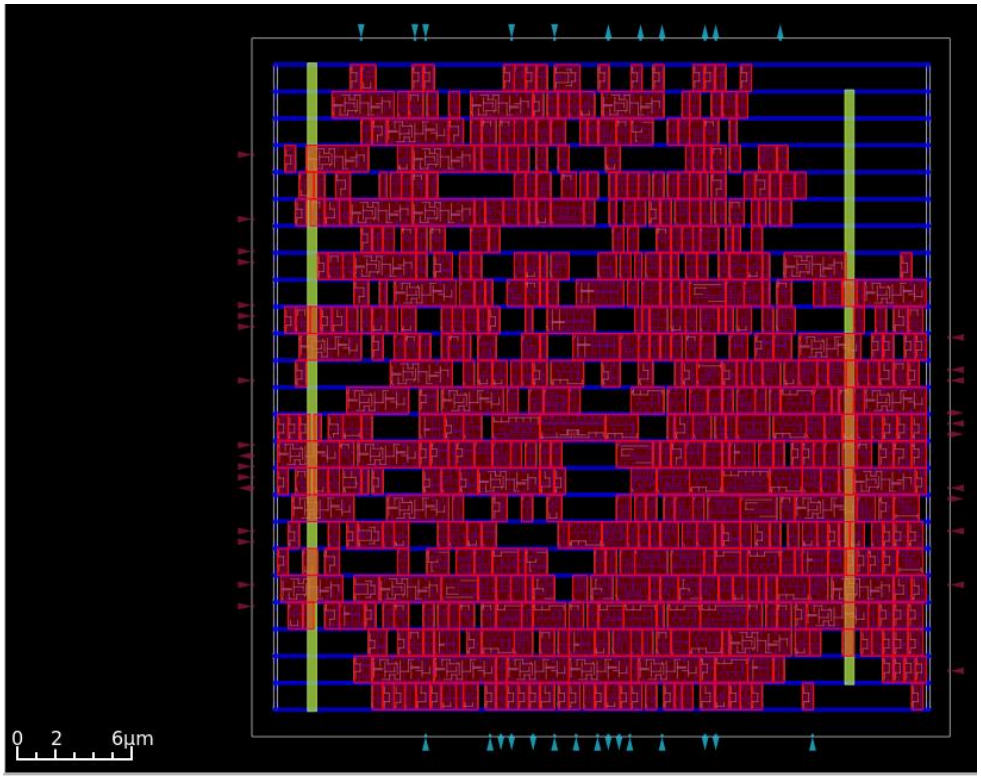


Clock Tree Synthesis:

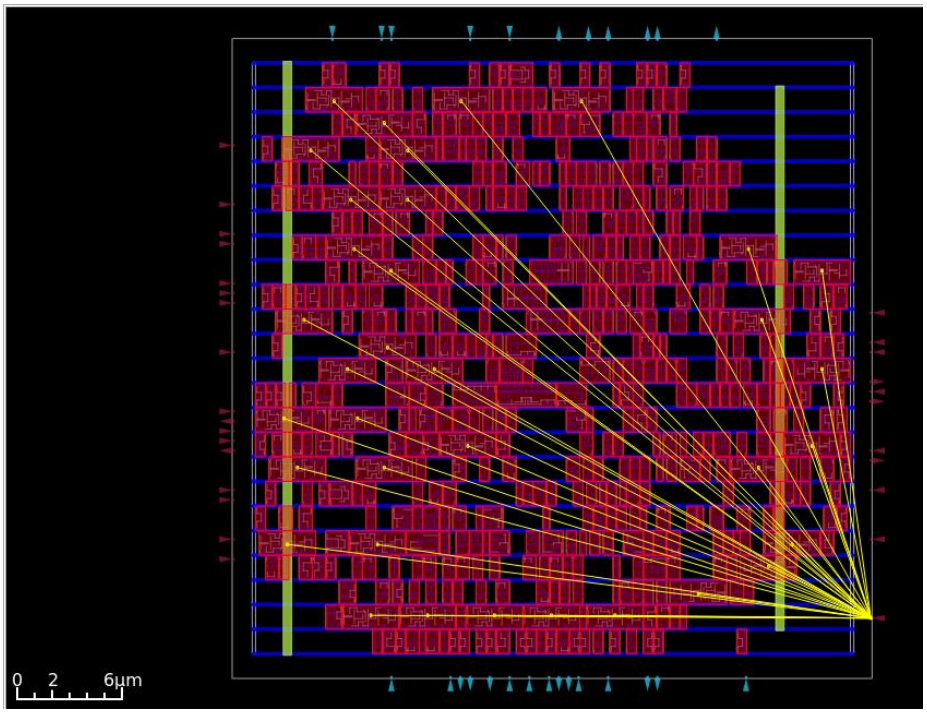
```
#   ___ ___ ___
#  / __|_ _/ __|
# | (__ | |\__ \
#  \___|___|___/
#
.PHONY: cts
cts: $(RESULTS_DIR)/4_cts.odb \
     $(RESULTS_DIR)/4_cts.sdc
# ==========================================================================

# Run TritonCTS
# --------------------------------------------------------------------------
$(eval $(call do-step,4_1_cts,$(RESULTS_DIR)/3_place.odb $(RESULTS_DIR)/3_place.sdc,cts))

$(RESULTS_DIR)/4_cts.sdc: $(RESULTS_DIR)/4_cts.odb

$(eval $(call do-copy,4_cts,4_1_cts.odb))

.PHONY: do-cts
do-cts:
        $(UNSET_AND_MAKE) do-4_1_cts do-4_cts

.PHONY: clean_cts
clean_cts:
        rm -rf $(RESULTS_DIR)/4_*cts*.odb $(RESULTS_DIR)/4_cts.sdc $(RESULTS_DIR)/4_*.v $(RESULTS_DIR)/4_*.def
        rm -f  $(REPORTS_DIR)/4_*
        rm -rf $(LOG_DIR)/4_*
```

CTS 是將 clock 分佈到電路的不同部分，來滿足 performance 和 reliability

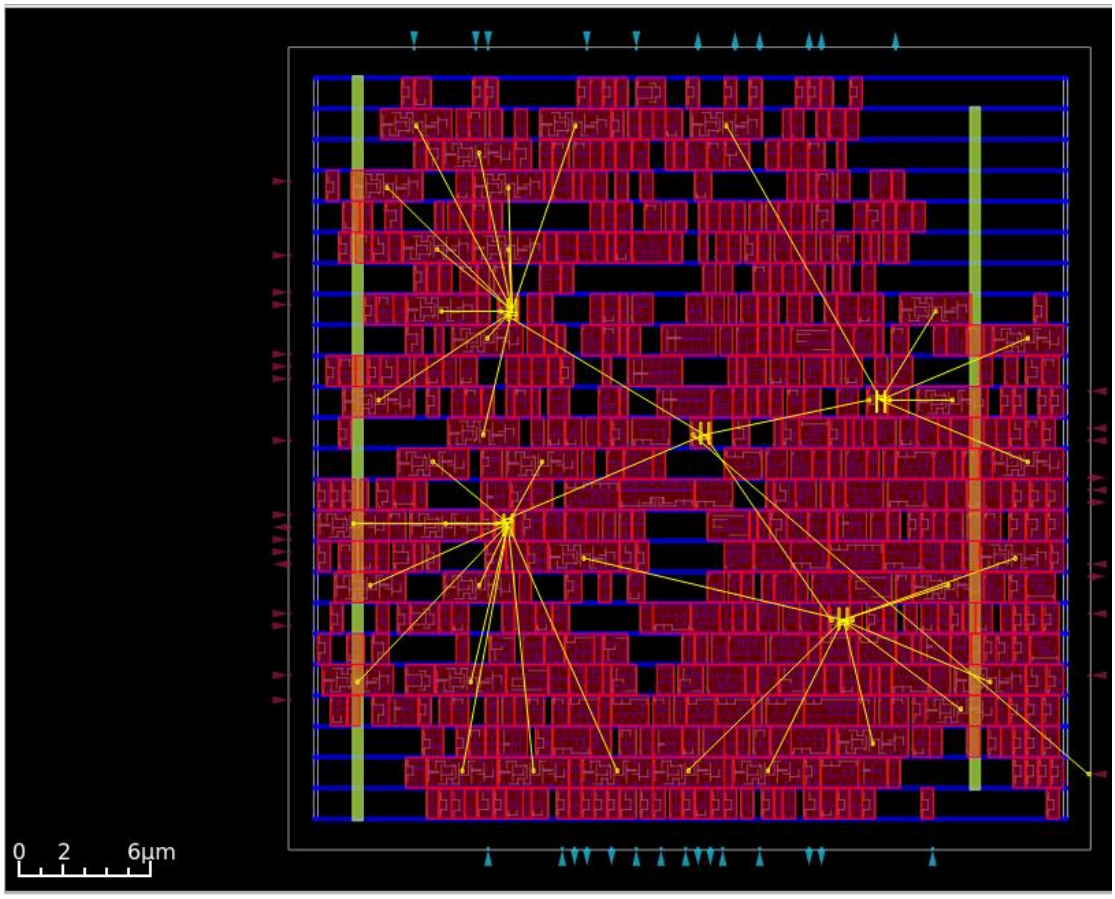
利用 TritonCTS 進行 CTS-award 的調整



CTS 之前的 clock



CTS 後



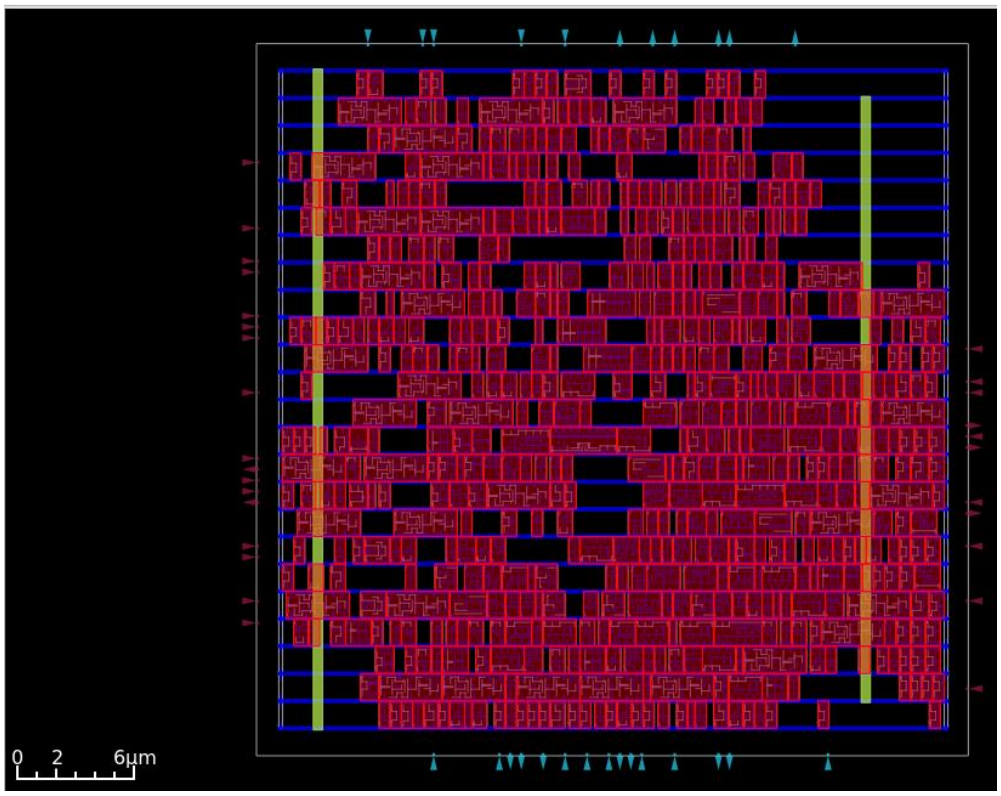
這五個 H 形狀 instance 代表的是 clock tree 連接處加入新的 buffer

Inspector

Name	Value
Type	Inst
Name	clkbuf_0_clk
Block	gcd
Module	<top>
Master	CLKBUF_X3
Description	Clock buffer
Placement status	PLACED
Source type	TIMING
Dont Touch	False
Orientation	R0
X	18.24 µm
Y	18.2 µm
ITerms	4 items
A	clk
Z	clknet_0_clk
VDD	<none>
VSS	<none>
BBox	(18.24, 18.2), (19.19, 19.6)
BBox Width, Height	(0.95, 1.4)

接著 Filler cell insertion 一樣用來提高電路的 performance 和 reliability，主要用來減少 clock tree 中 filler cells 跟 jitter

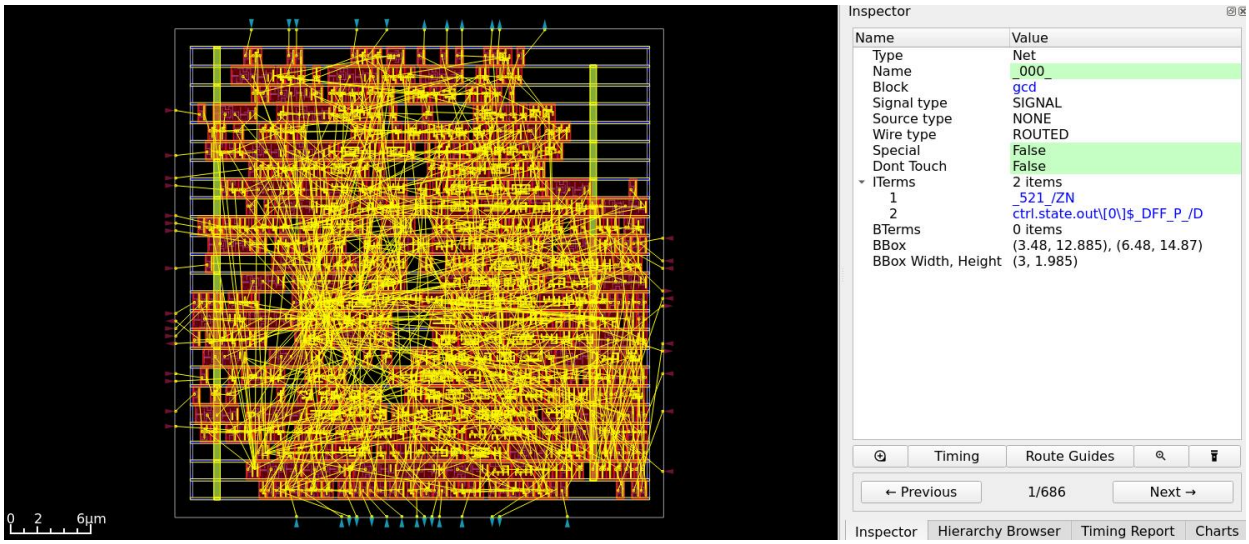
增加 clock tree 電容來降低 clock tree 的 power consumption



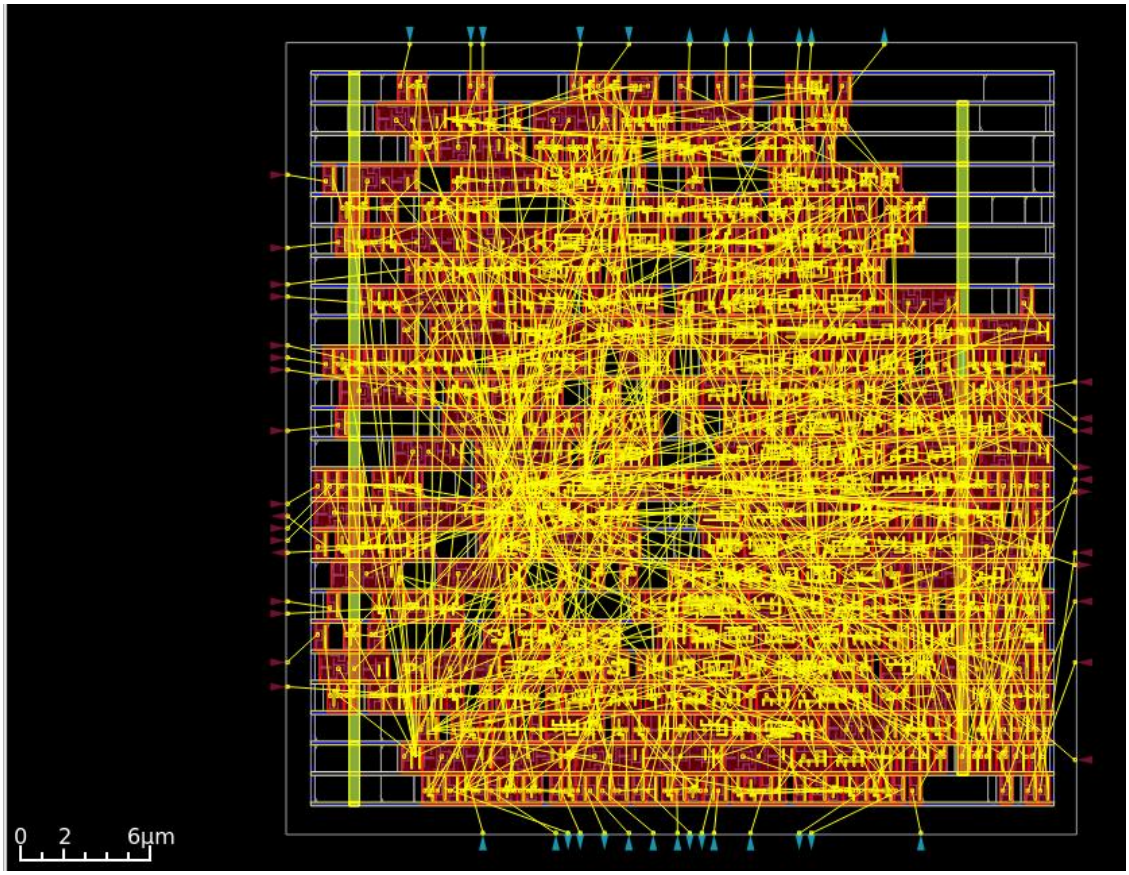
Routing:

```
#  ____   ___  _   _ _____ ___ _   _  ____
# |  _ \ / _ \| | | |_   _|_ _| \ | |/ ___|
# | |_) | | | | | | | | |  | ||  \| | |  _
# |  _ <| |_| | |_| | | |  | || |\  | |_| |
# |_| \_\\___/ \___/  |_| |___|_| \_|\____|
#
.PHONY: route
route: $(RESULTS_DIR)/5_route.odb \
       $(RESULTS_DIR)/5_route.sdc
# ========================================================================


# STEP 1: Run global route
#-------------------------------------------------------------------------------
$(eval $(call do-step,5_1_grt,$(RESULTS_DIR)/4_cts.odb $(FASTROUTE_TCL) $(PRE_GLOBAL_ROUTE),global_route))

# SEP 2: Filler cell insertion
# -------------------------------------------------------------------------------
$(eval $(call do-step,5_2_fillcell,$(RESULTS_DIR)/5_1_grt.odb,fillcell))

# STEP 3: Run detailed route
#-------------------------------------------------------------------------------
ifeq ($(USE_WXL),)
$(eval $(call do-step,5_3_route,$(RESULTS_DIR)/5_2_fillcell.odb,detail_route))
else
$(eval $(call do-step,5_3_route,$(RESULTS_DIR)/4_cts.odb,detail_route))
endif

$(eval $(call do-copy,5_route,5_3_route.odb))

$(eval $(call do-copy,5_route,4_cts.sdc,,.sdc))
```

Step1 :Run global route

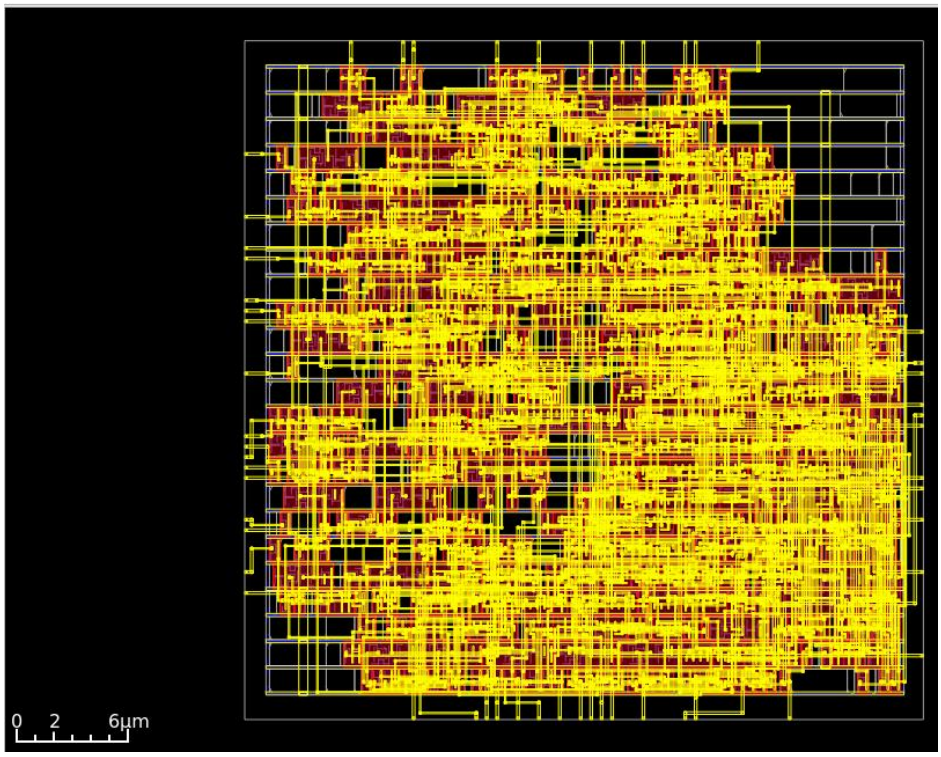


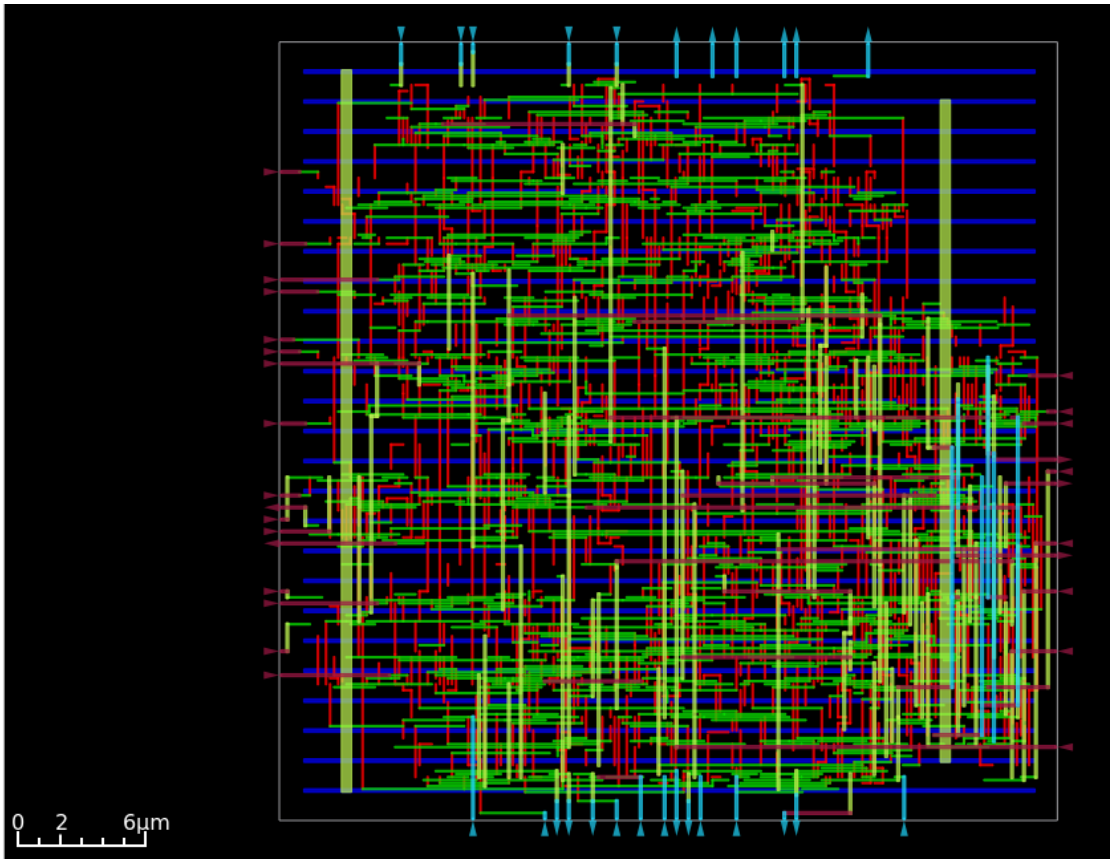
Step2 :Filler cell insertion



Step3 :Run detailed route

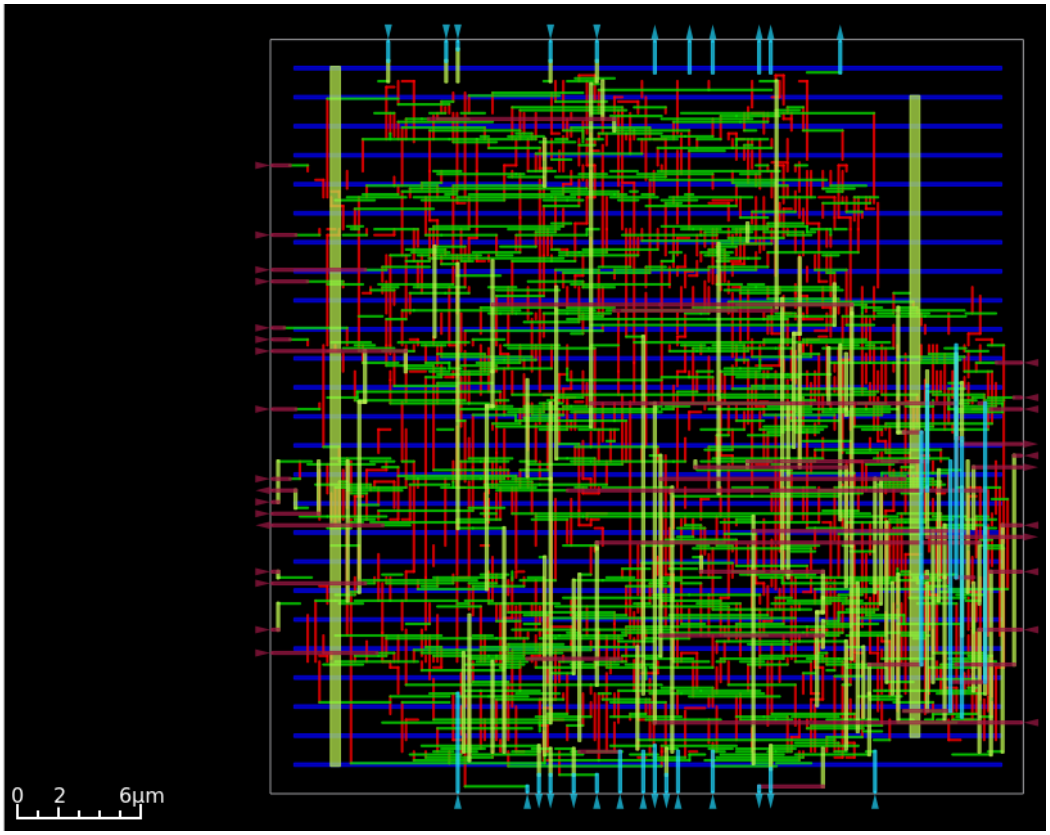
完成 global routing 後，完成 detailed routing





Finishing:

```
# ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____ ____
# |____|____|____|____|____|____|____|____|____|____|____|____|____|____|
# | |_  | ||  \| || |\___ \| |_| || ||  \| | |_ _|
# | _| | || |\  || |___) |  _  || || |\  | |_| |
# |_|   |___|_| \_|___|____/|_| |_|___|_| \_|\____|
#
GDS_FINAL_FILE = $(RESULTS_DIR)/6_final.$(STREAM_SYSTEM_EXT)
.PHONY: finish
finish: $(LOG_DIR)/6_report.log \
        $(RESULTS_DIR)/6_final.v \
        $(RESULTS_DIR)/6_final.sdc \
        $(GDS_FINAL_FILE)
        $(UNSET_AND_MAKE) elapsed

.PHONY: elapsed
elapsed:
        -@$(UTILS_DIR)/genElapsedTime.py -d $(BLOCK_LOG_FOLDERS) $(LOG_DIR)

# Useful when working with macros, see elapsed time for all macros in platform
.PHONY: elapsed-all
elapsed-all:
        @$(UTILS_DIR)/genElapsedTime.py -d $(shell find $(WORK_HOME)/logs/$(PLATFORM)/*/*/ -type d)

# ================================================================================

ifeq ($(USE_FILL),1)
$(eval $(call do-step,6_1_fill,$(RESULTS_DIR)/5_route.odb $(RESULTS_DIR)/5_route.sdc $(FILL_CONFIG),density_fill))
else
$(eval $(call do-copy,6_1_fill,5_route.odb))
endif

$(eval $(call do-copy,6_1_fill,5_route.sdc,,.sdc))

$(eval $(call do-copy,6_final,5_route.sdc,,.sdc))

$(eval $(call do-step,6_report,$(RESULTS_DIR)/6_1_fill.odb $(RESULTS_DIR)/6_1_fill.sdc,final_report,.log,$(LOG_DIR)))

$(RESULTS_DIR)/6_final.def: $(LOG_DIR)/6_report.log

# The final results are called 6_final.*, so it is convenient when scripting
# to have the names of the artifacts match the name of the target
.PHONY: do-final
do-final: do-finish

.PHONY: final
final: finish

.PHONY: do-finish
do-finish:
        $(UNSET_AND_MAKE) do-6_1_fill do-6_1_fill.sdc do-6_final.sdc do-6_report do-gds elapsed

.PHONY: generate_abstract
generate_abstract: $(RESULTS_DIR)/6_final.gds $(RESULTS_DIR)/6_final.def  $(RESULTS_DIR)/6_final.v $(RESULTS_DIR)/6_final.sdc
```



Final 的部分是做轉檔，把 odb 轉回 verilog

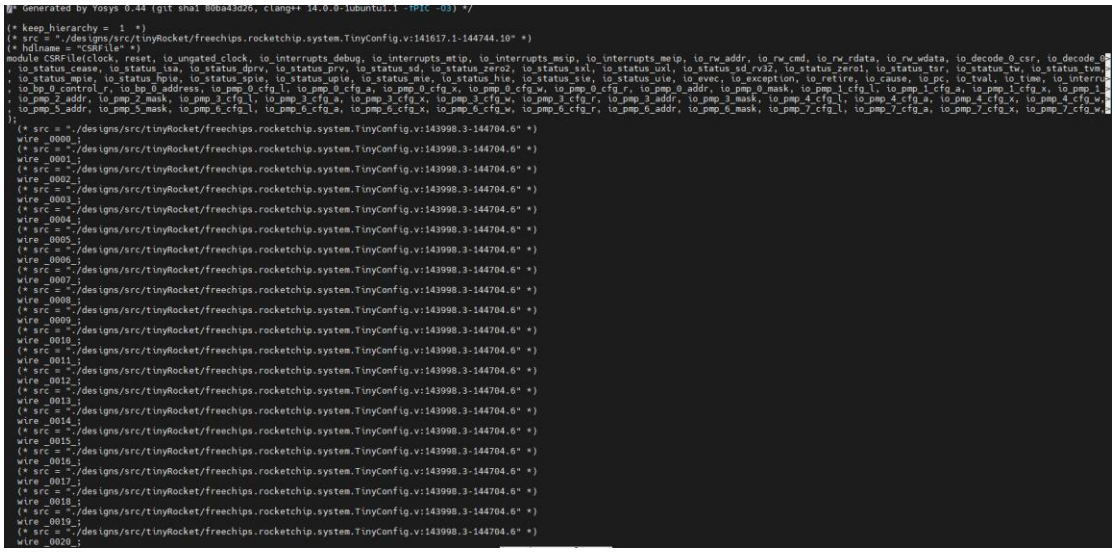
```
GDS_FINAL_FILE = $(RESULTS_DIR)/6_final.$(STREAM_SYSTEM_EXT)
.PHONY: finish
finish: $(LOG_DIR)/6_report.log \
        $(RESULTS_DIR)/6_final.v \
        $(RESULTS_DIR)/6_final.sdc \
        $(GDS_FINAL_FILE)
        $(UNSET_AND_MAKE) elapsed

.PHONY: elapsed
```

以下是另一個 IP 的結果對照，可以看到不同 IP 的不同差異性

這次選擇的是 nangate45 裡面的 tinyRocket

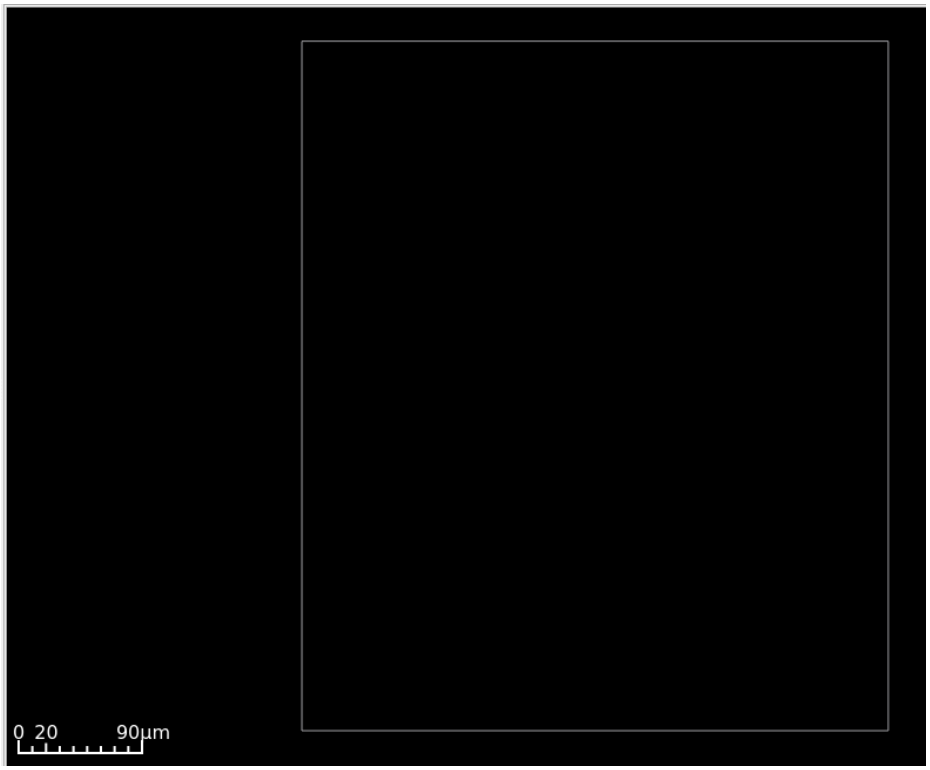
Log	Elapsed seconds	Peak Memory/MB
1_1_yosys	37	248
1_1_yosys_canonicalize	0	232
1_1_yosys_hier_report	33	163
2_1_floorplan	198	223
2_2_floorplan_io	0	153
2_4_floorplan_macro	21	216
2_5_floorplan_tapcell	0	142
2_6_floorplan_pdn	0	176
3_1_place_gp_skip_io	4	193
3_2_place_iop	0	161
3_3_place_gp	41	546
3_4_place_resized	10	431
3_5_place_dp	11	273
4_1_cts	276	599
5_1_grt	243	773
5_2_fillcell	0	212
5_3_route	50	2353
6_1_merge	2	642
6_report	33	811
Total	959	2353

Synthesis

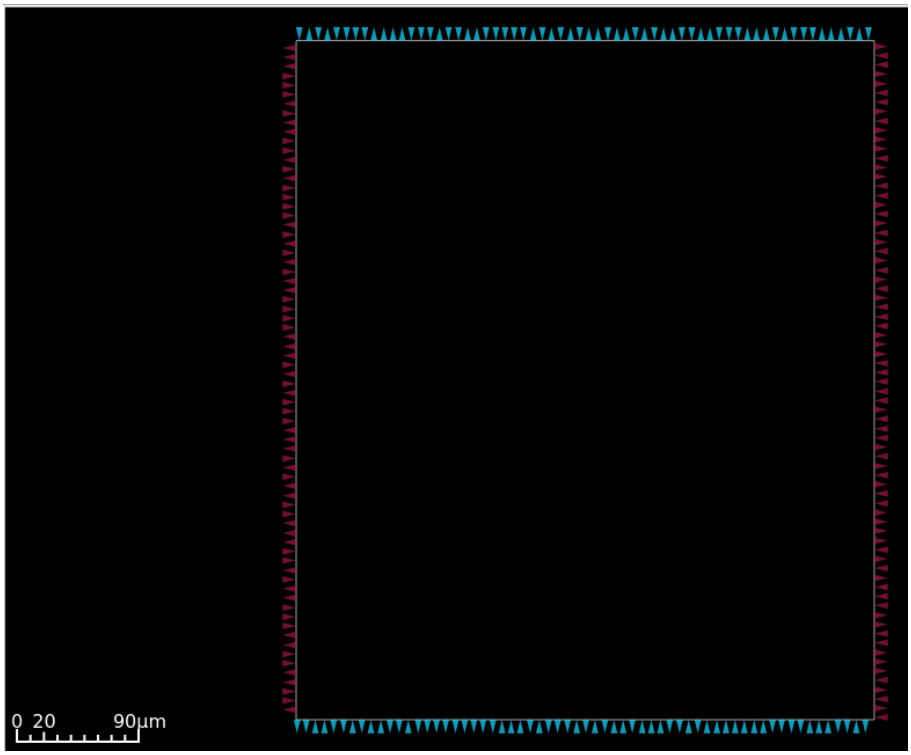


Floorplan

Step1 Translate Verilog to odb:

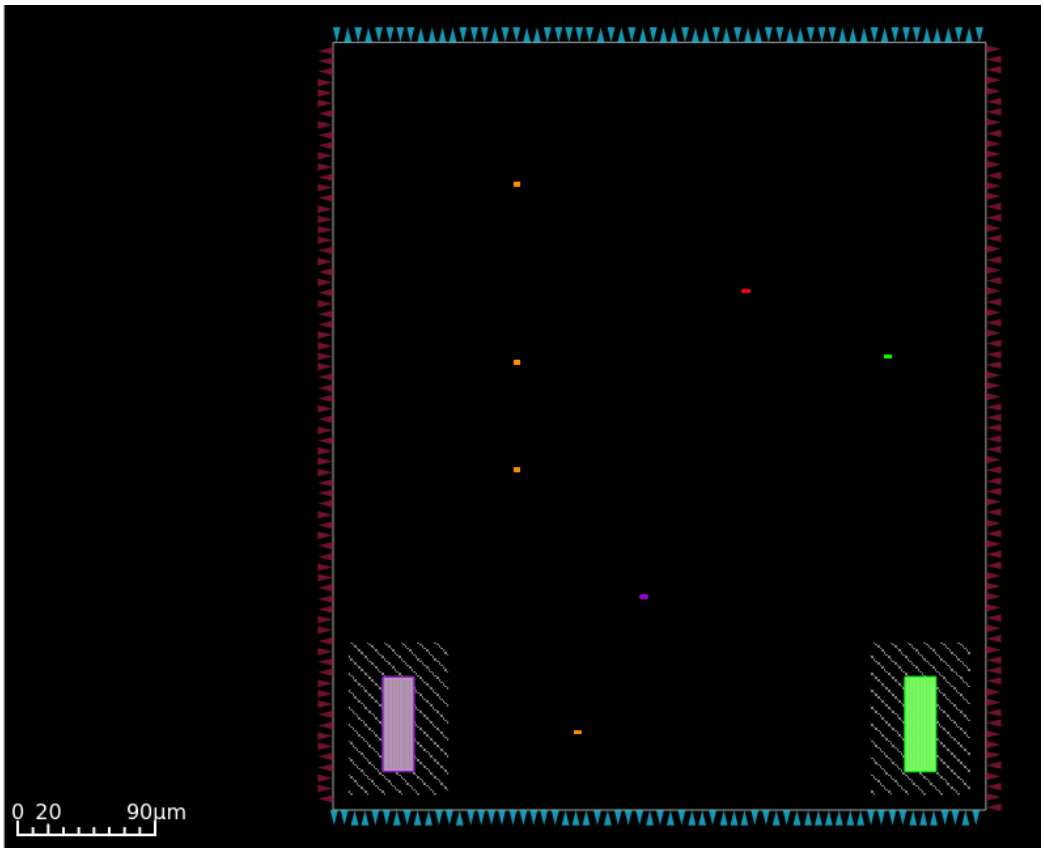


Step2 IO Placement (random)

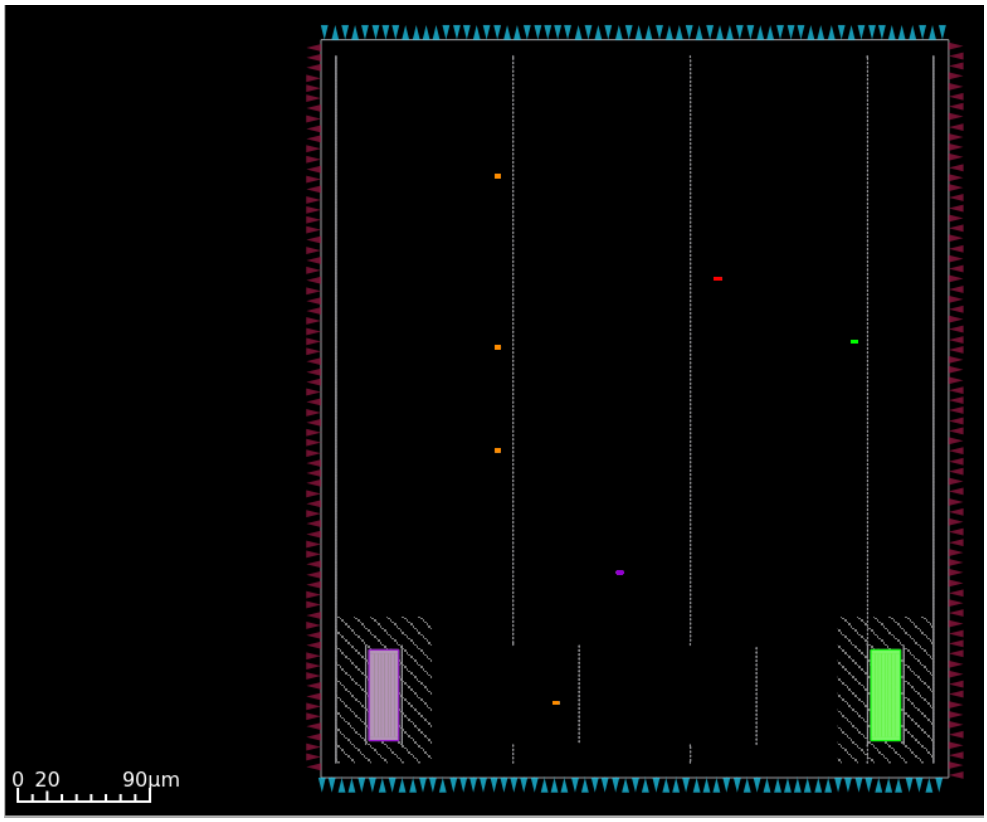


Step3 Timing Driven Mixed Sized Placement:

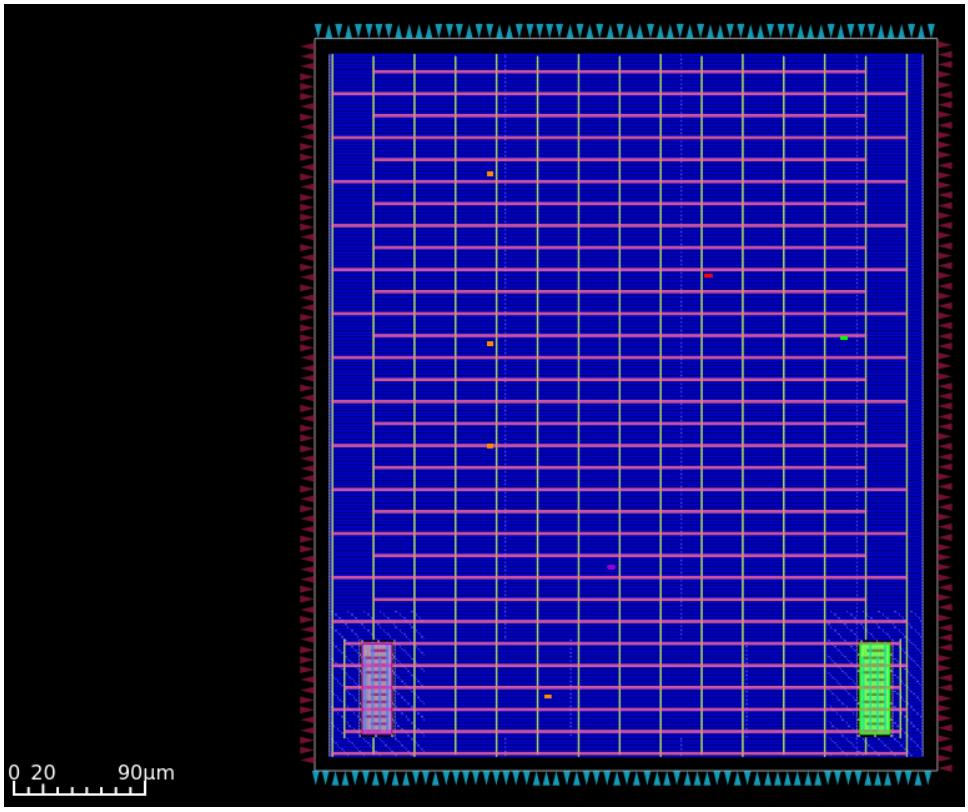
Step4 Macro Placement:



Step5 Tapcell and Welltie insertion:

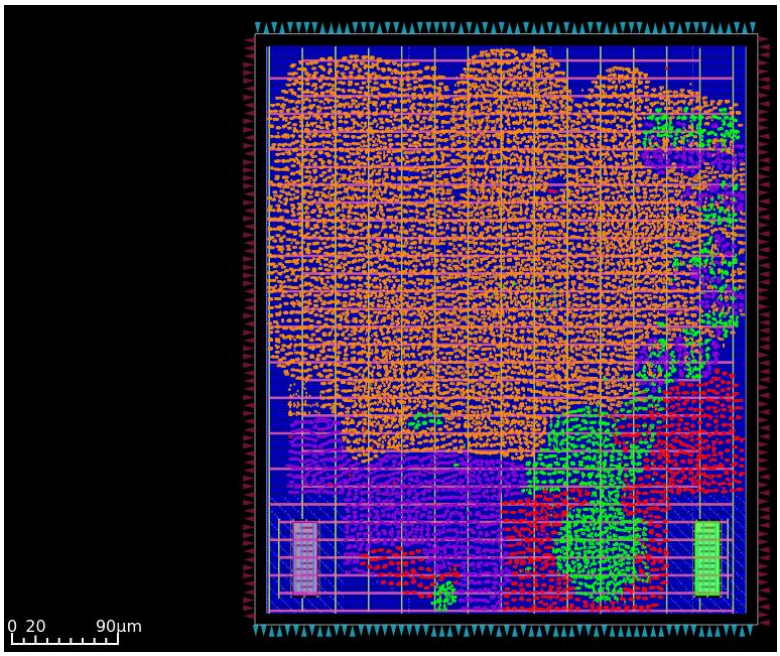


Step6 PDN generation:

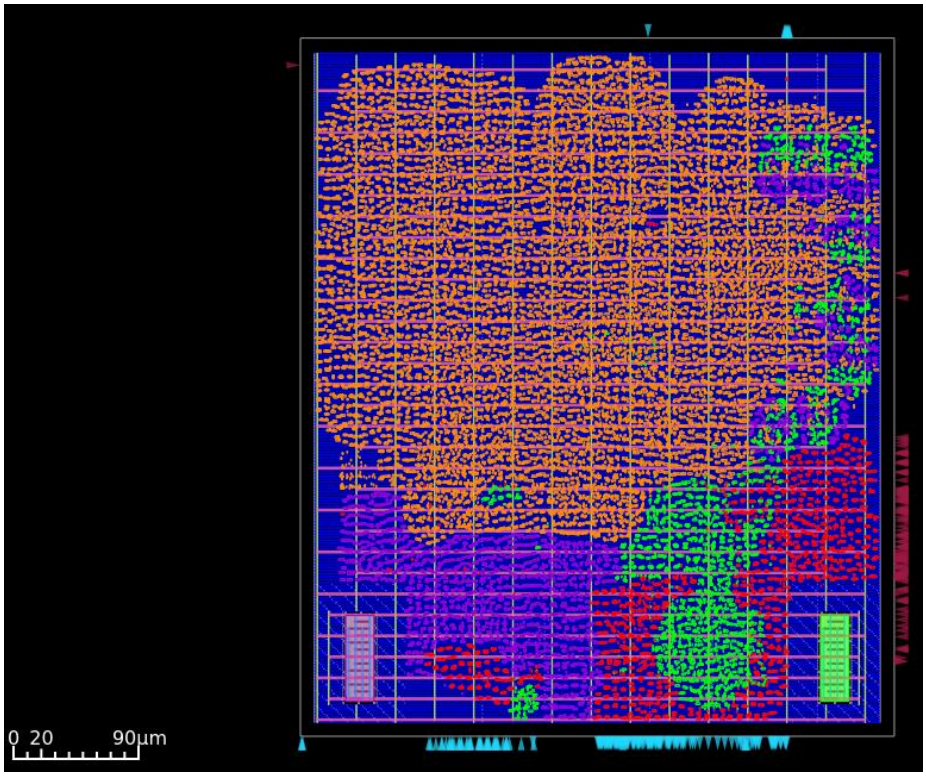


Placement:

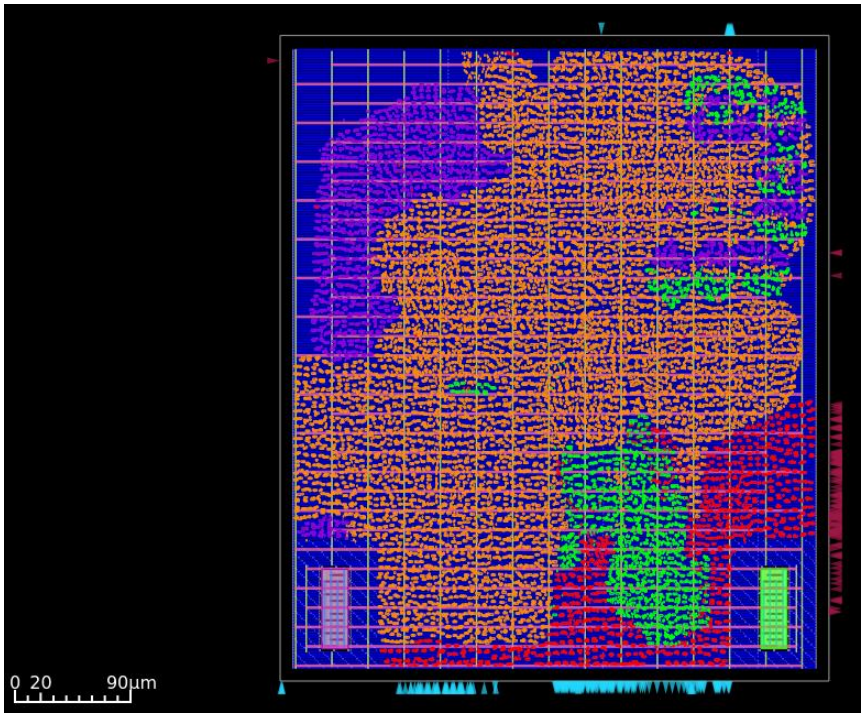
Step1 Global placement without placed IOs, timing-driven, and routability-driven:



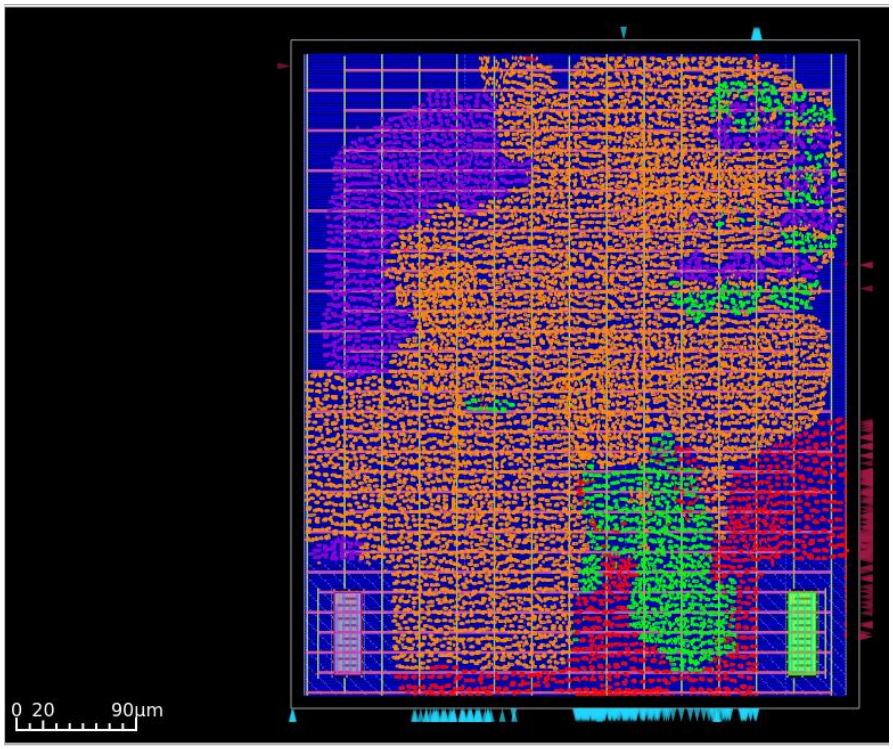
Step2 IO placement (non-random):



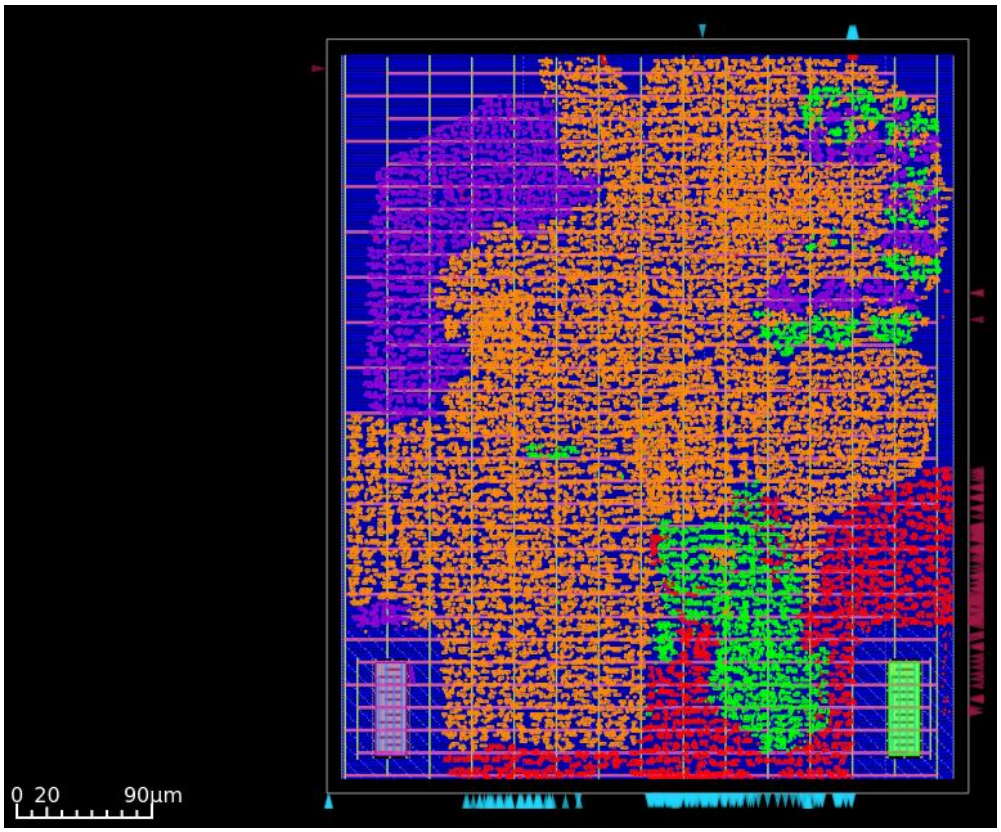
Step3 Global placement with placed IOs, timing-driven, and routability-driven:



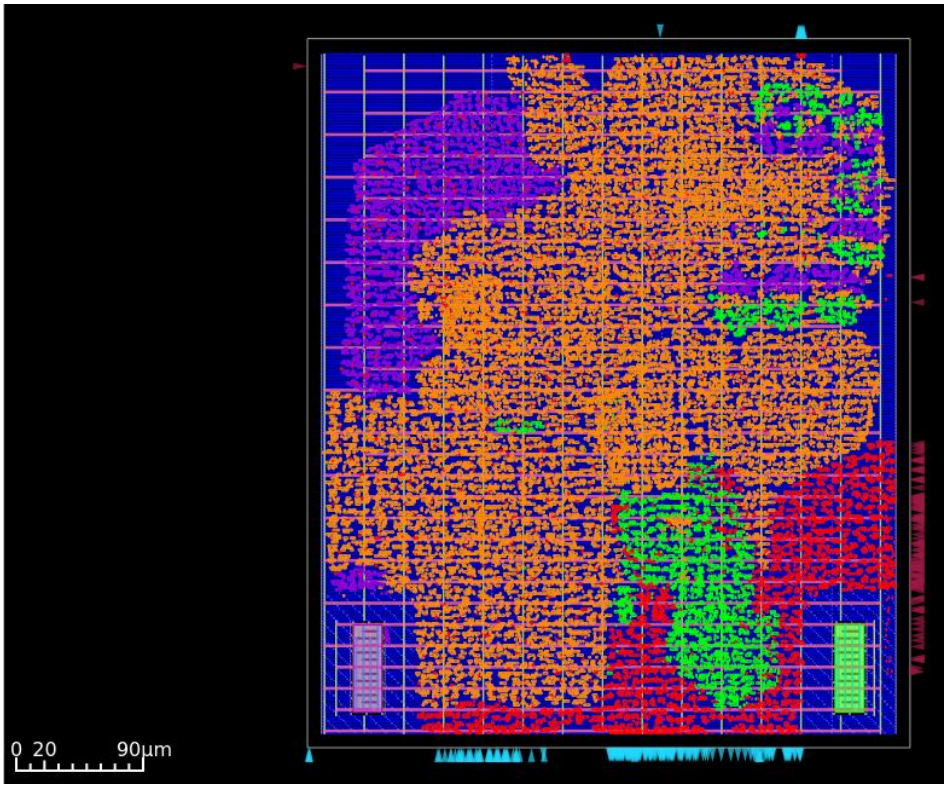
Step4 Resizing & Buffering:

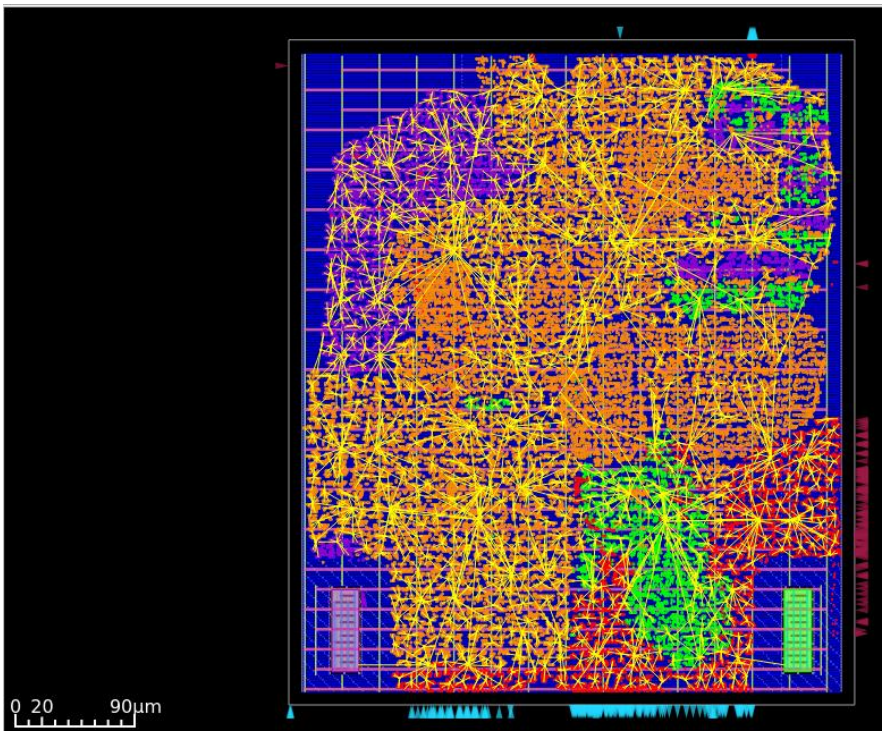


Step5 Detail placement:



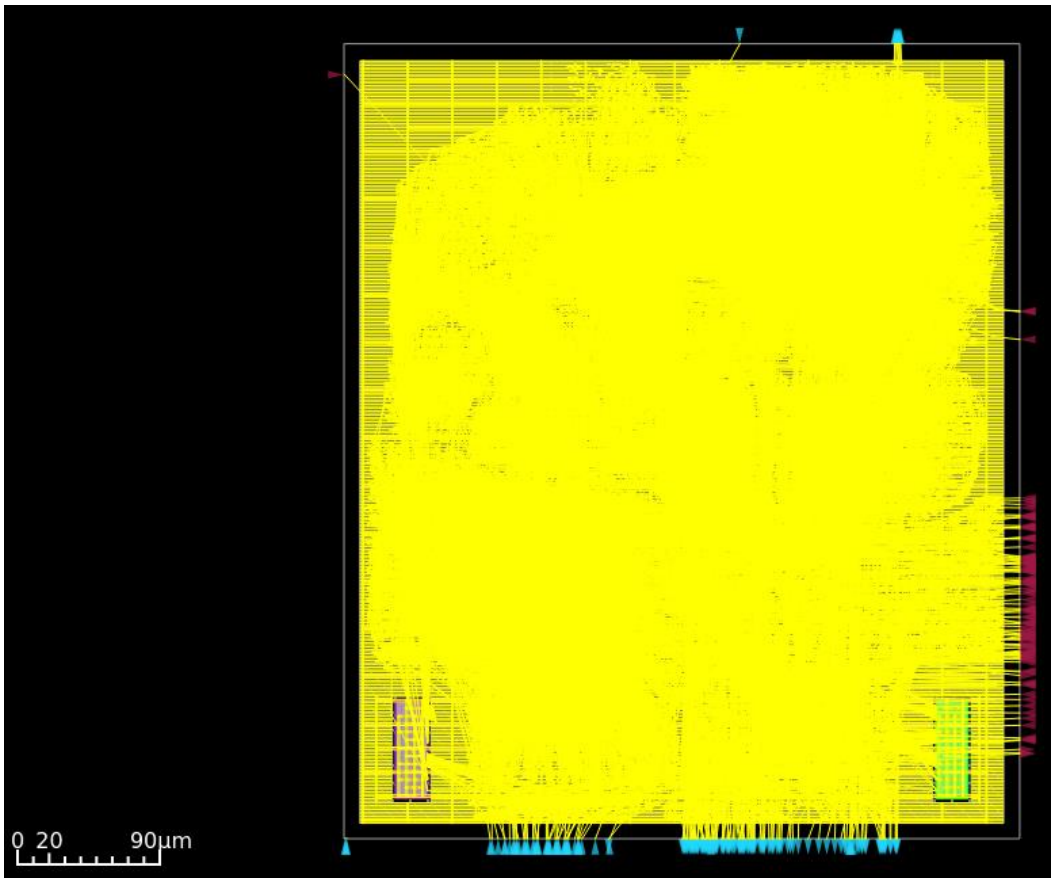
Clock Tree Synthesis:



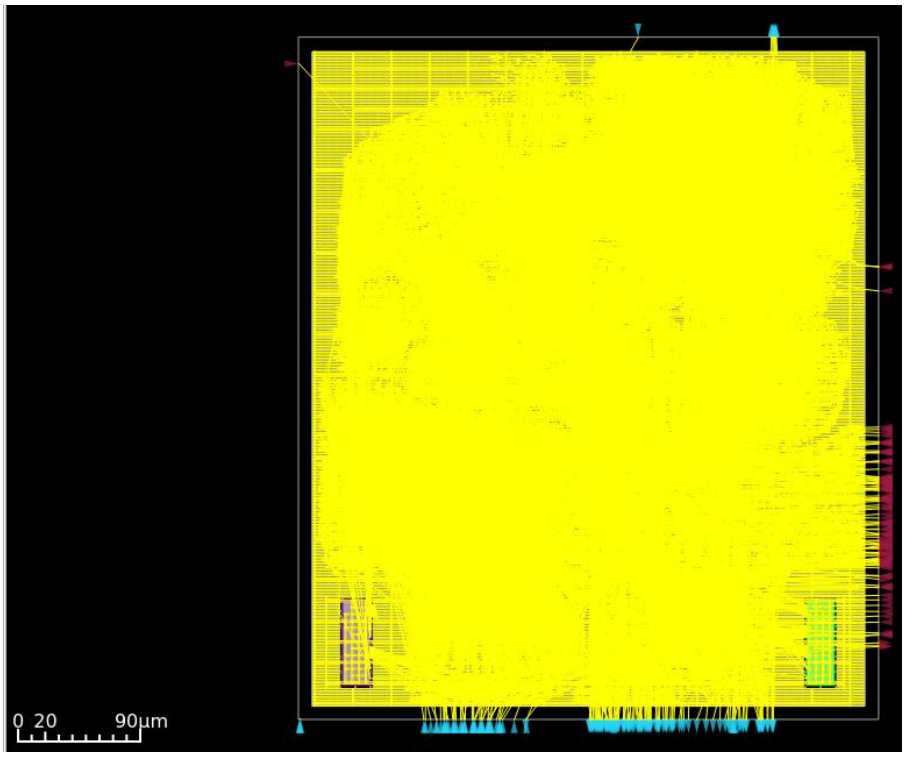


Routing:

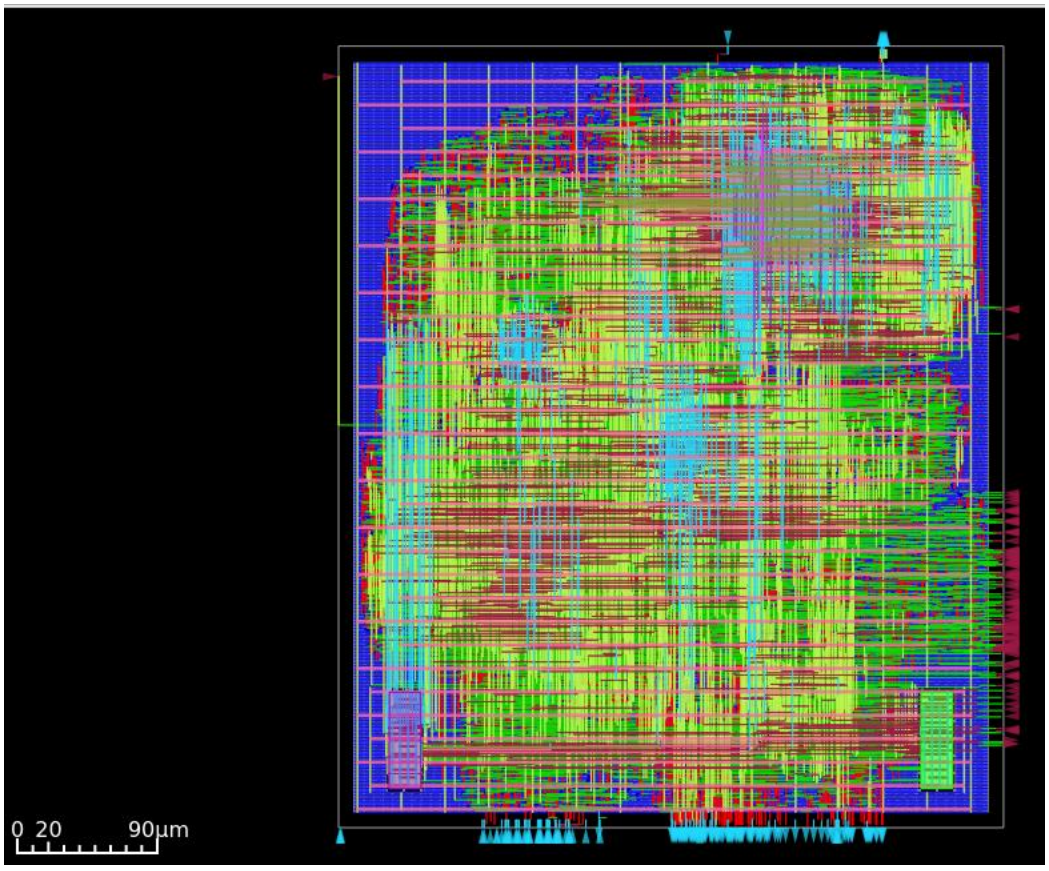
Step1 :Run global route



Step2 :Filler cell insertion



Step3 :Run detailed route



Finishing: