Digital System Design Project 4 – OpenROAD

一、下載 OpenROAD(取自老師給的下載流程)

(0) 下載 OpenROAD

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

cd OpenROAD-flow-scripts

git checkout 14994593a4eefbbea55e812b3bb6a1ef1ffb2d1e

(1) 安裝 OpenROAD 相依性套件

sudo ./setup.sh

(2) 若虛擬機內存不足，則需要透過 swap 分區創建

安裝指令

sudo mkdir -p /var/cache/swap/

sudo dd if=/dev/zero of=/var/cache/swap/swap0 bs=128M count=64

sudo chmod 0600 /var/cache/swap/swap0

sudo mkswap /var/cache/swap/swap0

sudo swapon /var/cache/swap/swap0

確認指令

sudo swapon -s

刪除指令

sudo swapoff /var/cache/swap/swap0

sudo rm /var/cache/swap/swap0

https://blog.csdn.net/weixin_44796670/article/details/121234446

(3) 安裝 OpenROAD 相依性套件

sudo ./setup.sh

(4) 編譯 OpenROAD

./build_openroad.sh --local

(5) 驗證 OpenROAD 是否正確編譯完成

環境參數設定

source ./env.sh

確認 yosys 及 openroad 可以執行並出現 help 說明

yosys -help

openroad -help

(6) 執行 OpenROAD

cd ~/OpenROAD-flow-scripts/flow

make

(7) 可觀察比較之結果

make gui_ <TAB>

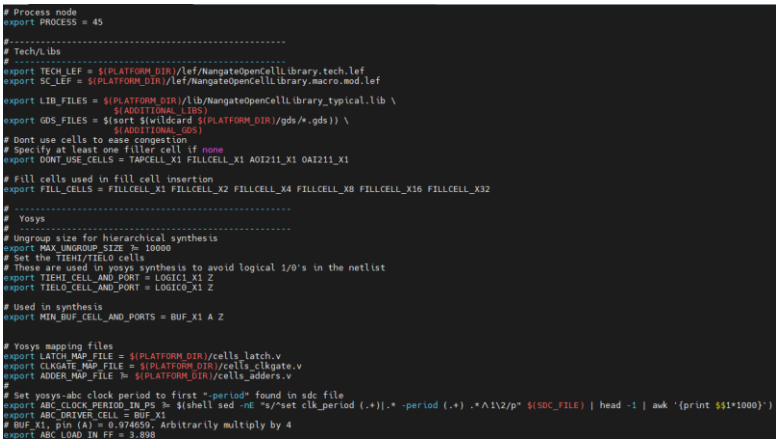
例如: make gui_final

二、GCD 晶片的設計流程

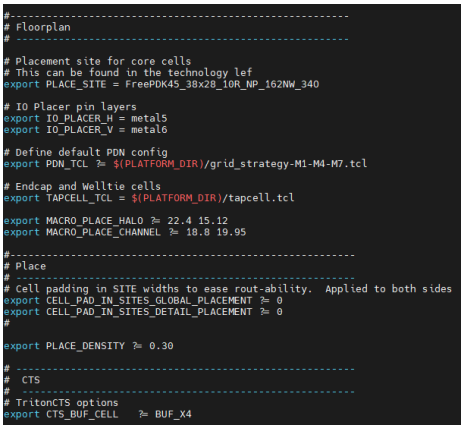
(0) 前處理

1. 分析製程: 這個文件可以分析使用的製程(process)、使用的 Floorplan、Place、CTS 參數等等。

   less ./platforms/nangate45/config.mk



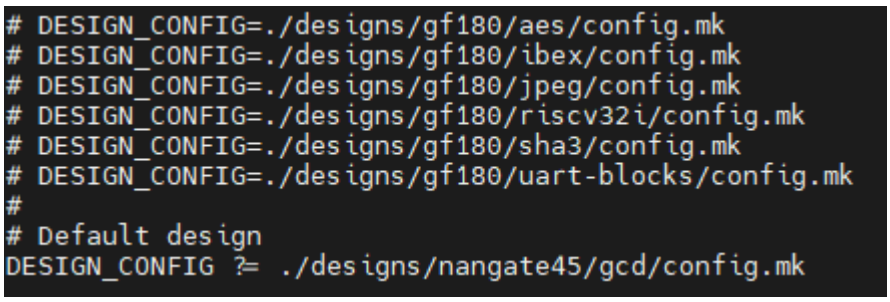
```
# Process node
export PROCESS = 45

#-----------------------------------------------------------
# Tech/Libs
#-----------------------------------------------------------
export TECH_LEF = $(PLATFORM_DIR)/lef/NangateOpenCellLibrary.tech.lef
export SC_LEF = $(PLATFORM_DIR)/lef/NangateOpenCellLibrary.macro.mod.lef

export LIB_FILES = $(PLATFORM_DIR)/lib/NangateOpenCellLibrary_typical.lib \
                   $(ADDITIONAL_LIBS)
export GDS_FILES = $(sort $(wildcard $(PLATFORM_DIR)/gds/*.gds)) \
                   $(ADDITIONAL_GDS)
# Dont use cells to ease congestion
# Specify at least one filler cell if none
export DONT_USE_CELLS = TAPCELL_X1 FILLCELL_X1 AOI211_X1 OAI211_X1

# Fill cells used in fill cell insertion
export FILL_CELLS = FILLCELL_X1 FILLCELL_X2 FILLCELL_X4 FILLCELL_X8 FILLCELL_X16 FILLCELL_X32

#-----------------------------------------------------------
#  Yosys
#-----------------------------------------------------------
# Ungroup size for hierarchical synthesis
export MAX_UNGROUP_SIZE ?= 10000
# Set the TIEHI/TIELO cells
# These are used in yosys synthesis to avoid logical 1/0's in the netlist
export TIEHI_CELL_AND_PORT = LOGIC1_X1 Z
export TIELO_CELL_AND_PORT = LOGIC0_X1 Z

# Used in synthesis
export MIN_BUF_CELL_AND_PORTS = BUF_X1 A Z


# Yosys mapping files
export LATCH_MAP_FILE = $(PLATFORM_DIR)/cells_latch.v
export CLKGATE_MAP_FILE = $(PLATFORM_DIR)/cells_clkgate.v
export ADDER_MAP_FILE ?= $(PLATFORM_DIR)/cells_adders.v
#
# Set yosys-abc clock period to first *-period* found in sdc file
export ABC_CLOCK_PERIOD_IN_PS ?= $(shell sed -nE "s/^set clk_period (.+)|.* -period (.+) .*/\1\2/p" $(SDC_FILE) | head -1 | awk '{print $$1*1000}')
export ABC_DRIVER_CELL = BUF_X1
# BUF_X1, pin (A) = 0.974659. Arbitrarily multiply by 4
export ABC_LOAD_IN_FF = 3.898
```



```
#-----------------------------------------------------------
# Floorplan
#-----------------------------------------------------------

# Placement site for core cells
# This can be found in the technology lef
export PLACE_SITE = FreePDK45_38x28_10R_NP_162NW_340

# IO Placer pin layers
export IO_PLACER_H = metal5
export IO_PLACER_V = metal6

# Define default PDN config
export PDN_TCL ?= $(PLATFORM_DIR)/grid_strategy-M1-M4-M7.tcl

# Endcap and Welltie cells
export TAPCELL_TCL = $(PLATFORM_DIR)/tapcell.tcl

export MACRO_PLACE_HALO ?= 22.4 15.12
export MACRO_PLACE_CHANNEL ?= 18.8 19.95

#-----------------------------------------------------------
# Place
#-----------------------------------------------------------
# Cell padding in SITE widths to ease rout-ability.  Applied to both sides
export CELL_PAD_IN_SITES_GLOBAL_PLACEMENT ?= 0
export CELL_PAD_IN_SITES_DETAIL_PLACEMENT ?= 0
#

export PLACE_DENSITY ?= 0.30

#-----------------------------------------------------------
#  CTS
#-----------------------------------------------------------
# TritonCTS options
export CTS_BUF_CELL    ?= BUF_X4
```

2. 編譯檔案: 此報告將使用預設 nangate45 的 gcd 做分析，不用改動 Makefile 即可開始後開始編譯

   make



```
# DESIGN_CONFIG=./designs/gf180/aes/config.mk
# DESIGN_CONFIG=./designs/gf180/ibex/config.mk
# DESIGN_CONFIG=./designs/gf180/jpeg/config.mk
# DESIGN_CONFIG=./designs/gf180/riscv32i/config.mk
# DESIGN_CONFIG=./designs/gf180/sha3/config.mk
# DESIGN_CONFIG=./designs/gf180/uart-blocks/config.mk
#
# Default design
DESIGN_CONFIG ?= ./designs/nangate45/gcd/config.mk
```

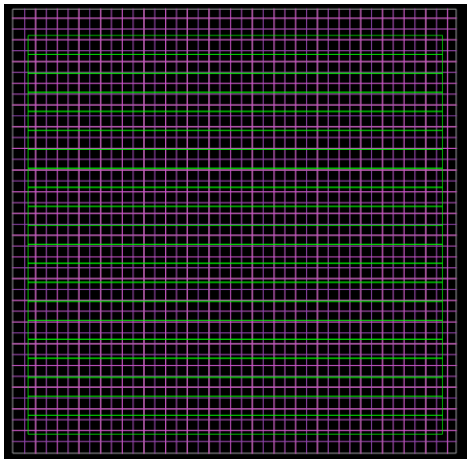
   以下透過 GUI 分析每一步驟的設計配置和流程

(1) Synthesis: 用 HDL code 等方法，做出晶片的功能、面積等最佳化

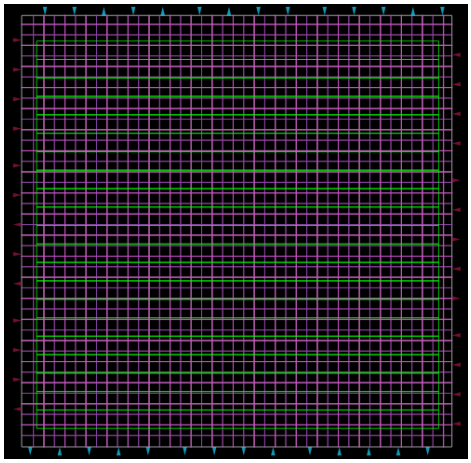
```
wire \dpath.a_lt_b$in1[5] ;
wire \dpath.a_lt_b$in1[6] ;
wire \dpath.a_lt_b$in1[7] ;
wire \dpath.a_lt_b$in1[8] ;
wire \dpath.a_lt_b$in1[9] ;
input [31:0] req_msg;
output req_rdy;
input req_val;
input reset;
output [15:0] resp_msg;
input resp_rdy;
output resp_val;
INV_X2 _345_ (
  .A(\dpath.a_lt_b$in1[1] ),
  .ZN(_036_)
);
BUF_X2 _346_ (
  .A(\dpath.a_lt_b$in0[1] ),
  .Z(_037_)
);
NAND2_X1 _347_ (
  .A1(_036_),
  .A2(_037_),
  .ZN(_038_)
);
INV_X1 _348_ (
  .A(\dpath.a_lt_b$in1[0] ),
  .ZN(_039_)
);
NOR2_X1 _349_ (
  .A1(_039_),
  .A2(\dpath.a_lt_b$in0[0] ),
  .ZN(_040_)
);
NOR2_X1 _350_ (
  .A1(_036_),
  .A2(_037_),
  .ZN(_041_)
);
```

(2) Floor Planning：決定晶片布局

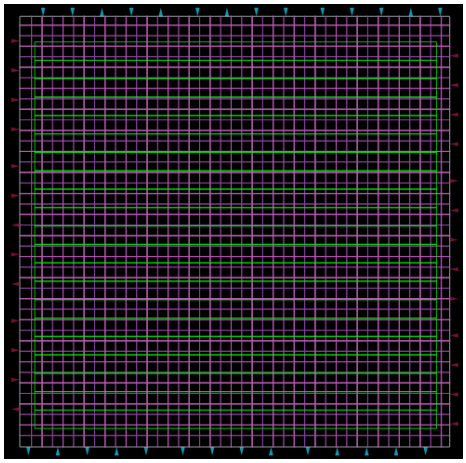
1. Translate verilog to odb (把 verilog 轉變成 obd)



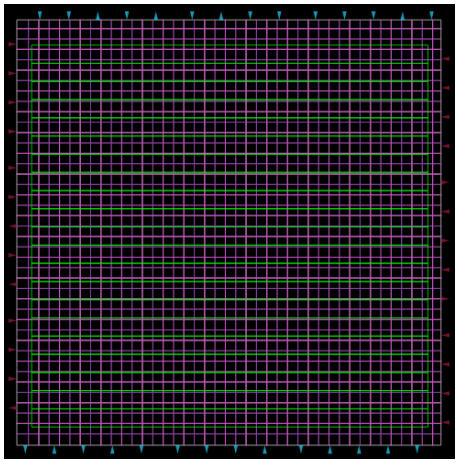
2. IO Placement (隨機擺放 IO)



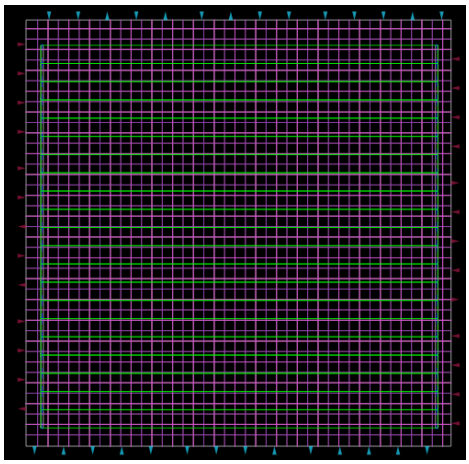
3. Timing Driven Mixed Sized Placement(擺放)



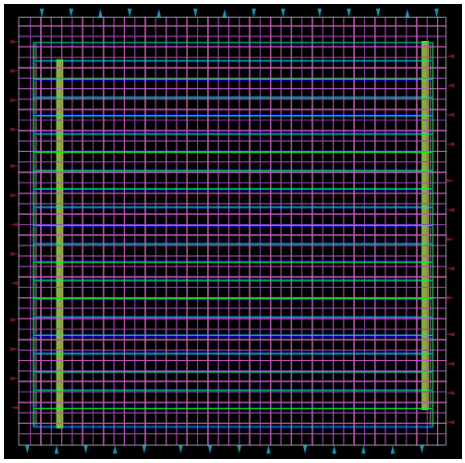
4. Macro Placement(宏觀的擺放)



5. Tapcell and Welltie insertion(tap cell 與 well tie 的擺放)

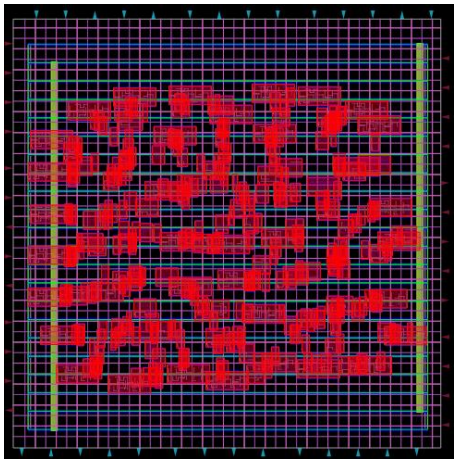


6. PDN generation(電源規劃)

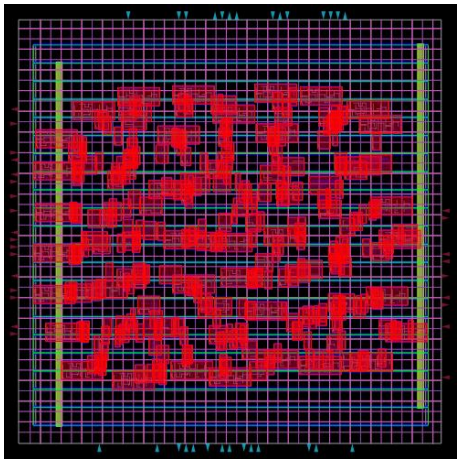


(3) Placement：放入邏輯閘、元件

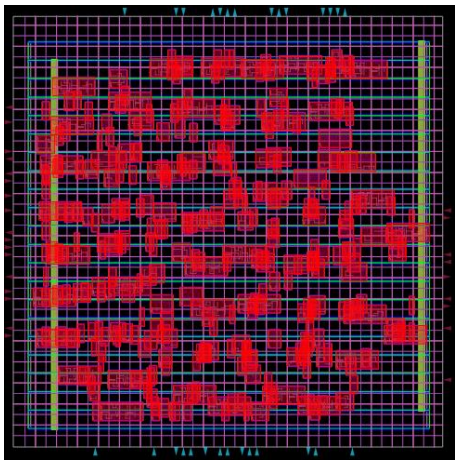
1. Global placement without placed IOs, timing-driven, and routability-driven(大致上的布局，擺放 flip flop 等)



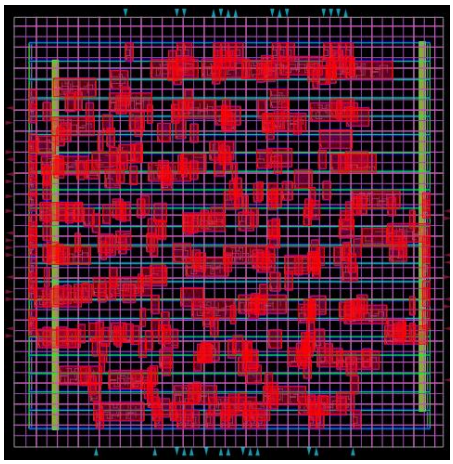
2. IO placement (non-random)(再次布置 IO)



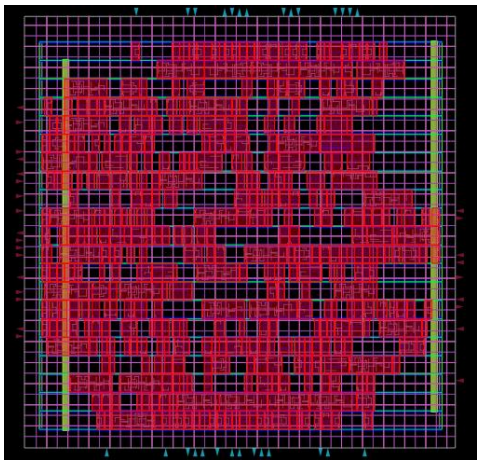
3. Global placement with placed IOs, timing-driven, and routability-driven(根據 IO 等放置 flip flop 等)



4. Resizing & Buffering(重設大小)

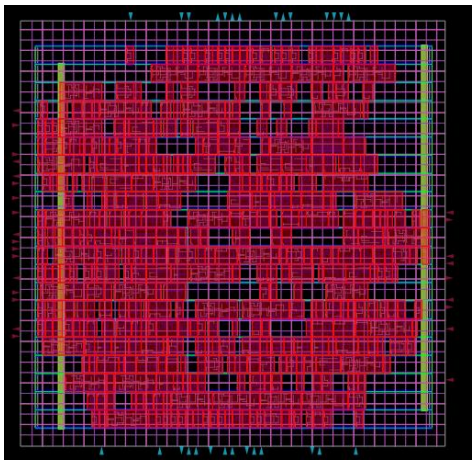


5. Detail placement(詳細的調整，flip flop 對齊電源網路等等)

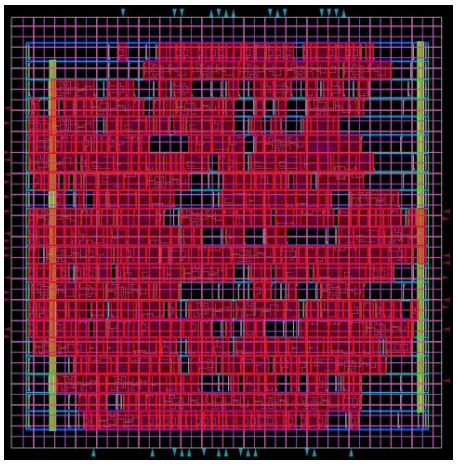


(4) Clock Tree Synthesis(CTS)：設定路徑把 clock 信號傳遞給所有 flip flop

1. CTS (確定 clock 連接點的位置，但在這一步中尚未加入 filter)

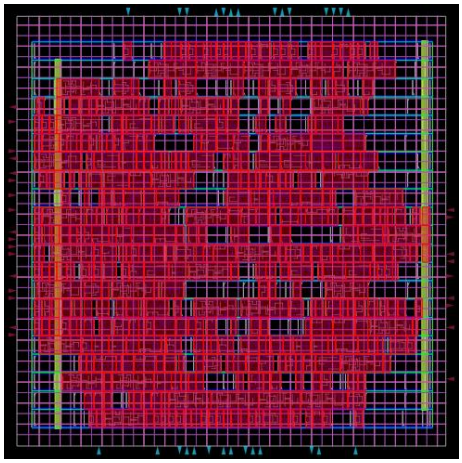


2. CTS fill cell (使用 filter 填補還未安裝 flip-flop 的空位)

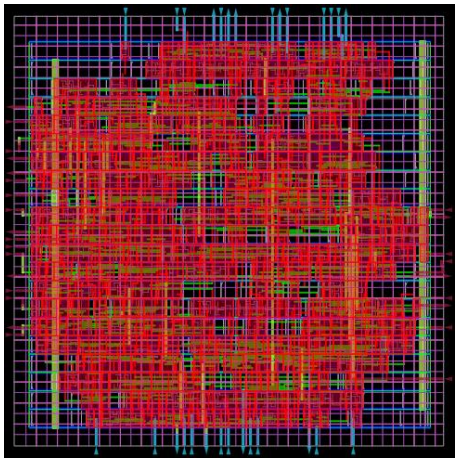


(5) Routing：繞線

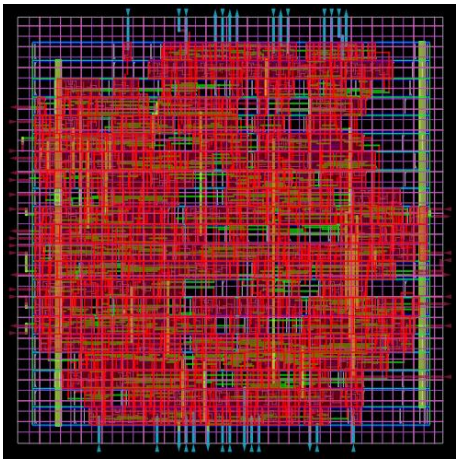
1. Run global route



2. Run detailed route



(6) Finishing：完成設計



(7) 參考資料

[20231207] Digital System Design Flow (youtube.com)

10 分鐘認識 IC 開發流程(IC Design Flow)｜方格子 vocus

PowerPoint 簡報 (nthu.edu.tw)

数字 ASIC 设计概要：Tap Cell - 非是非 (noyesno.net)

https://github.com/The-OpenROAD-Project/OpenROAD/blob/master/src/pdn/README.md

OpenROAD — OpenROAD documentation

三、從 open-source IC design flow 的學習

我大概追蹤了一下流程，雖然沒有完全理解，但大概知道流程，從用寫 verilog 找最佳化、晶片布局、放入元件、設定 clock、繞線到完成設計，慢慢地了解一點點 IC 設計的輪廓，不過由於沒有親自設計過，而且也沒接觸龐大的 IC 設計方法，仍有許多不了解的地方，期待未來若有機會，我能修課學會，並且能更看得懂這些流程。

四、課程反饋

老師是我認為資工裡最認真教課的老師，除了會錄影給每個同學重複聽課的機會，對於課程的準備也很認真，而且可以藉由前三個 Project，訓練程式能力，雖然我常常去找老師問問題，但老師還是不厭其煩的回答我的問題。

至於建議的部分，我希望能多一點 socrative 的練習題目，像是 self-correct 和 counter divided by x 等可以多幾題練習加強觀念；project3 我曾經建議範例要修改，但後來我覺得過於麻煩，對於老師有點負擔，因此我認為至少劃出比較線，或是那一小段部分重錄會讓大家比較了解；在最後 IC 設計的流程，需要有實際的圖來說明會更好一點，只有流程圖，可能仍然不清楚，或是可以以比喻的方式來講解，像是有清大的學長以蓋房子的宏觀方式比喻 IC 的微觀設計。
PowerPoint 簡報 (nthu.edu.tw)

最後，感謝老師這一年來的陪伴，讓我從連電路都不懂的高中生，學到了各種邏輯設計，希望未來有機會能修到老師的課。