

Introduction of Openroad design flow

M11215060 陳軒緯

Openroad 的運作流程會將 RTL Verilog(.v), constraints (.sdc), liberty (.lib) 和 technology (.lef) 作為 input，最終可得到 tapeoutready GDSII file. (此檔案表示積體電路已設計完成，可交由代工廠生產)

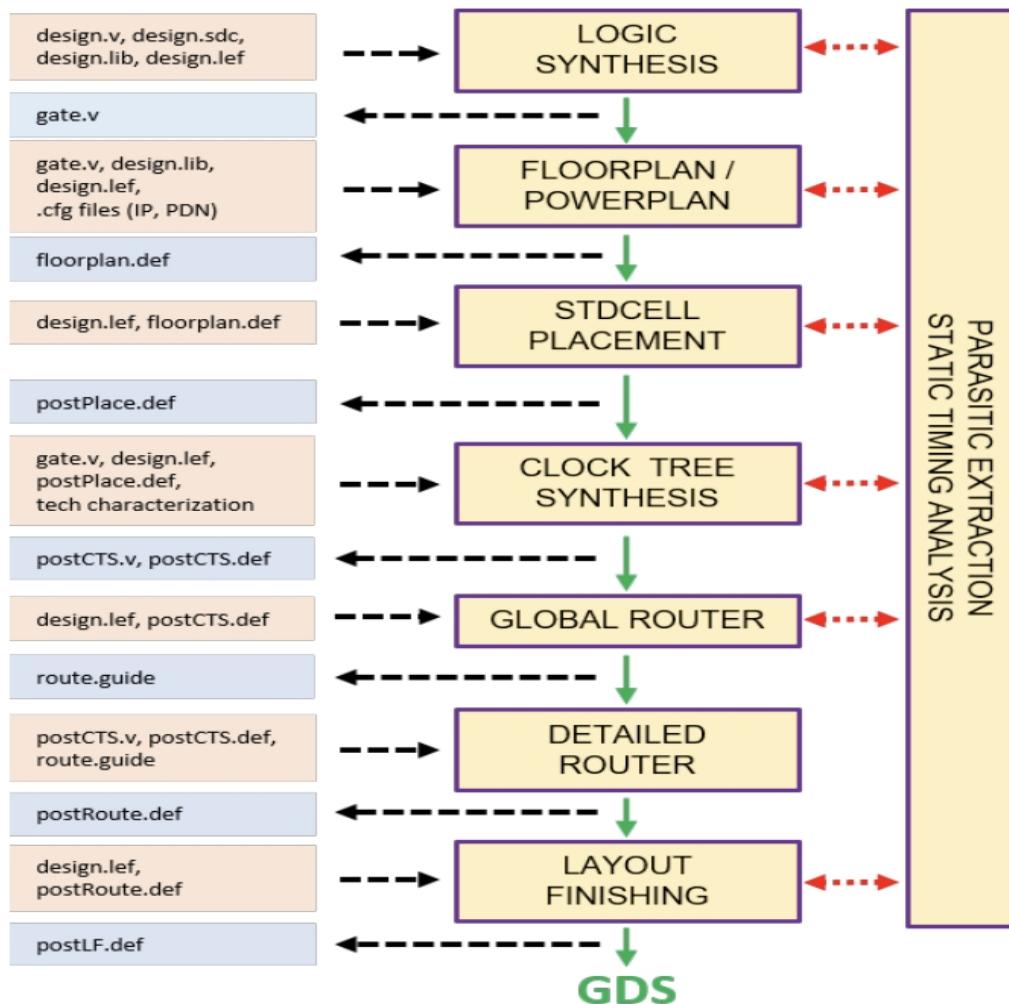


Figure 1: The OpenROAD flow.

Synthesis:

Openroad 使用 yosys 框架，並且使用強化式學習開發了可以自動生成 step-by-step synthesis scripts，以滿足時間限制，同時最小化總面積。此階段主要是將 RTL code 轉成 netlist 形式（邏輯閘 gate-level），gate-level 指的是把全部描述語言轉換成邏輯表示式，之後再做 placement 和 routing 時，就是以 netlist 為 input。

```

    .resp_msg ( dpath$resp_msg ),
    .ts_a_lt_b ( dpath$ts_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$sa_mux_sel = ctrl$sa_mux_sel;
assign dpath$sa_reg_en = ctrl$sa_reg_en;
assign dpath$sb_mux_sel = ctrl$sb_mux_sel;
assign dpath$sb_reg_en = ctrl$sb_reg_en;
assign dpath$cclk = clk;
assign dpath$req_msg_a = req_msg[31:16];
assign dpath$req_msg_b = req_msg[15:0];
assign dpath$reset = reset;
assign req_rdy = ctrl$resp_rdy;
assign resp_msg = dpath$resp_msg;
assign resp_val = ctrl$resp_val;

endmodule // GcdUnit

//-----  

// GcdUnitCtrlRTL_0x4d0fc71ead8d3d9e  

//-----  

// dump-vcd: False  

// verilator-xinit: zeros
module GcdUnitCtrlRTL_0x4d0fc71ead8d3d9e
(
    output reg [ 1:0] a_mux_sel,
    output reg [ 0:0] a_reg_en,
    output reg [ 0:0] b_mux_sel,
    output reg [ 0:0] b_reg_en,
    input wire [ 0:0] clk,
    input wire [ 0:0] is_a_lt_b,

```

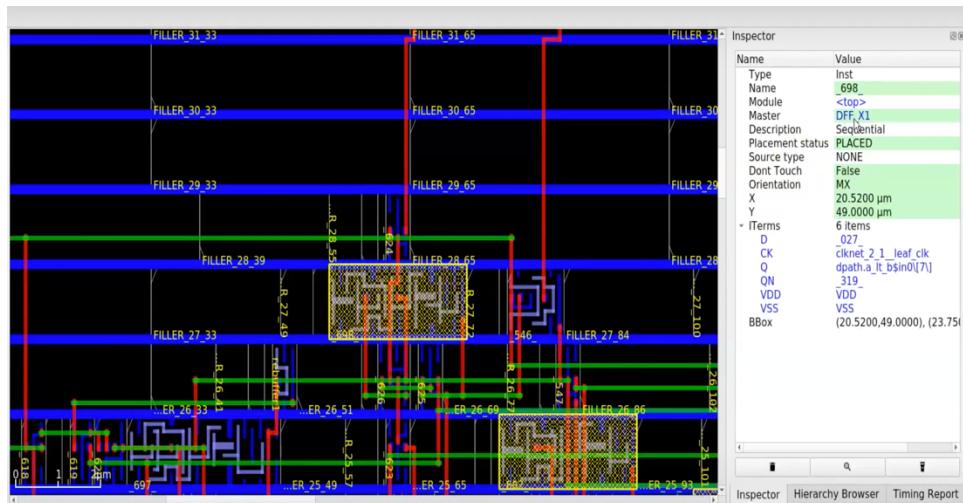
▲Rtl code

```

2205 );
2206 DFF_X1_695_ (
2207     .CK(clk),
2208     .D(.D04),
2209     .Q(dpath.a_lt_b$in0[4]),
2210     .QN(_322)
2211 );
2212 DFF_X1_696_ (
2213     .CK(clk),
2214     .D(.D05),
2215     .Q(dpath.a_lt_b$in0[5]),
2216     .QN(_321)
2217 );
2218 DFF_X1_697_ (
2219     .CK(clk),
2220     .D(.D06),
2221     .Q(dpath.a_lt_b$in0[6]),
2222     .QN(_320)
2223 );
2224 DFF_X1_698_ (
2225     .CK(clk),
2226     .D(.D07),
2227     .Q(dpath.a_lt_b$in0[7]),
2228     .QN(_319)
2229 );
2230 DFF_X1_699_ (
2231     .CK(clk),
2232     .D(.D08),
2233     .Q(dpath.a_lt_b$in0[8]),
2234     .QN(_318)
2235 );
2236 DFF_X1_700_ (
2237     .CK(clk),
2238     .D(.D09),
2239     .Q(dpath.a_lt_b$in0[9]),
2240     .QN(_317)
2241 );
2242 DFF_X1_701_ (
2243     .CK(clk),

```

▲邏輯閘 (ex:698 邏輯閘，下圖為 openroad 對應的邏輯閘 name698)



Floorplanning:

Floorplan 主要內容為確認晶片的形狀和尺寸，IO 單元（隨機）的 placement，macro 的 placement 和 Tapcell and Welltie 的規劃，最後生產電源分佈網路，也就是電源系統整體路線規劃。

Step 1: Translate Verilog to odb

Step 2: IO placement (random)

Step 3: Timing Driven Mixed Sized Placement

Step 4: Macro Placement

Step 5: Tapcell and Welltie insertion

Step 6: PDN generation

```
2_1_floorplan.odb  
2_2_floorplan_io.odb  
2_3_floorplan_tdms.odb  
2_4_floorplan_macro.odb  
2_5_floorplan_tapcell.odb  
2_6_floorplan_pdn.odb
```

▲每個步驟依序產生的檔案

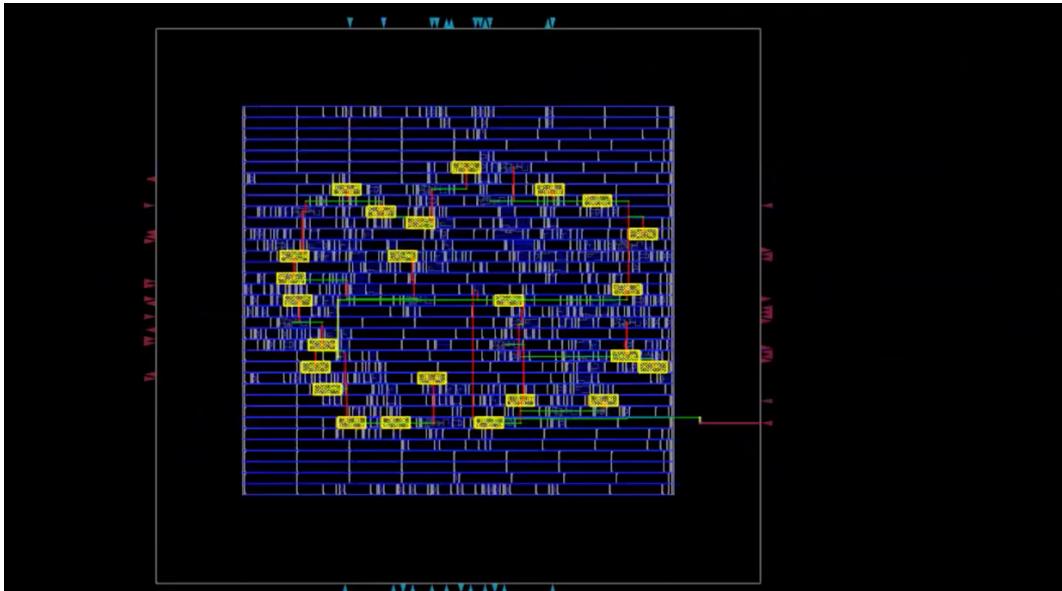
Placement:

Placement 決定標準元件在晶片上的實體位置，決定 IO 的 placement (非隨機)。這個階段，會接收 synthesized circuit netlist 和 technology library 並產出一個合法的 layout。這個 layout 根據前面步驟的優化已準備好 resizing 和 buffering。(即下圖 step3 至 step4)

```
PLACE  
place: ${RESULTS_DIR}/3_place.odb \  
$RESULTS_DIR/3_place.sdc  
# STEP 1: Global placement without placed IOs, timing-driven, and routability-driven.  
#-----  
${RESULTS_DIR}/3_1_place_gp_skip_io.odb: ${RESULTS_DIR}/2_floorplan.odb ${RESULTS_DIR}/2_floorplan.sdc ${LIB_FILES}  
($TIME_CMD) ${OPENROAD_CMD} ${SCRIPTS_DIR}/global_place_skip_io.tcl -metrics ${LOG_DIR}/3_1_place_gp_skip_io.json) 2>&1 | tee ${LOG_DIR}/3_1_place_gp_skip_io.log  
# STEP 2: IO placement (non-random)  
#-----  
${RESULTS_DIR}/3_2_place_iop.odb: ${RESULTS_DIR}/3_1_place_gp_skip_io.odb ${IO_CONSTRAINTS}  
ifndef IS_CHIP  
($TIME_CMD) ${OPENROAD_CMD} ${SCRIPTS_DIR}/io_placement.tcl -metrics ${LOG_DIR}/3_2_place_iop.json) 2>&1 | tee ${LOG_DIR}/3_2_place_iop.log  
else  
cp $< $@  
endif  
# STEP 3: Global placement with placed IOs, timing-driven, and routability-driven.  
#-----  
${RESULTS_DIR}/3_3_place_gp.odb: ${RESULTS_DIR}/3_2_place_top.odb ${RESULTS_DIR}/2_floorplan.sdc ${LIB_FILES}  
($TIME_CMD) ${OPENROAD_CMD} ${SCRIPTS_DIR}/global_place.tcl -metrics ${LOG_DIR}/3_3_place_gp.json) 2>&1 | tee ${LOG_DIR}/3_3_place_gp.log  
# STEP 4: Resizing & Buffering  
#-----  
${RESULTS_DIR}/3_4_place_resized.odb: ${RESULTS_DIR}/3_3_place_gp.odb ${RESULTS_DIR}/2_floorplan.sdc  
($TIME_CMD) ${OPENROAD_CMD} ${SCRIPTS_DIR}/resize.tcl -metrics ${LOG_DIR}/3_4_resizer.json) 2>&1 | tee ${LOG_DIR}/3_4_resizer.log  
clean_resize:  
rm -f ${RESULTS_DIR}/3_4_place_resized.odb  
# STEP 5: Detail placement
```

CTS:

產生 clock tree，將外部 clock 妥善分配給內部的各個元件（clock tree synthesis）。CTS 是一個 clock balancing 的技術，用以維持訊號的完整性，降低 clock skew 和 clock latency。在 CTS 這個階段會對 clock tree 作分析、優化 clock 的擺放位置、在 clock 路徑上加 buffer 來推動 clock tree。



▲在 Openroad 中 Clock tree (綠、紅線) 將 Flip-Flop (黃色區塊) 連接起來

Routing:

在 placement 摆好 cell 之後，對擺放好的 cell 作拉線。在 Openroad 中，會先 run global route，接著再 run detailed route。

```
# STEP 1: Run global route
# $(RESULTS_DIR)/5_1_grt.odb: $(RESULTS_DIR)/4_cts.odb $(FASTRUTE_TCL) $(PRE_GLOBAL_ROUTE)
#   ($TIME_CMD) $(OPENROAD_CMD) $(SCRIPTS_DIR)/global_route.tcl -metrics $(LOG_DIR)/5_1_fastroute.json 2>&1 | tee $(LOG_DIR)/5_1_fastroute.log

# STEP 2: Run detailed route
#
ifeq ($(USE_WXL),)
$(RESULTS_DIR)/5_2_route.odb: $(RESULTS_DIR)/5_1_grt.odb
else
$(RESULTS_DIR)/5_2_route.odb: $(RESULTS_DIR)/4_cts.odb
endif
    ($TIME_CMD) $(OPENROAD_CMD) $(SCRIPTS_DIR)/detail_route.tcl -metrics $(LOG_DIR)/5_2_TritonRoute.json 2>&1 | tee $(LOG_DIR)/5_2_TritonRoute.log

$(RESULTS_DIR)/5_route.odb: $(RESULTS_DIR)/5_2_route.odb
cp $< $@

$(RESULTS_DIR)/5_route.sdc: $(RESULTS_DIR)/4_cts.sdc
cp $< $0
cp $< $0

clean_route:
    rm -rf output*/ results*.out.dmp layer_*/*.mps
    rm -rf *.gdd *.log *.met *.sav *.res.dmp
    rm -rf $(RESULTS_DIR)/route_guide $(RESULTS_DIR)/output_guide.mod $(RESULTS_DIR)/updated_clks.sdc
    rm -rf $(RESULTS_DIR)/5_*.odb $(RESULTS_DIR)/5_route.sdc
    rm -f $(REPORTS_DIR)/5_*
    rm -f $(LOG_DIR)/5_*
    rm -f $(LOG_DIR)/5_*

klayout_tr_rpt: $(RESULTS_DIR)/5_route.def $(OBJECTS_DIR)/klayout.lyt
    $(call KLAYOUT_FOUND)
    $(K_LAYOUT_CMD) -rd in_dir=$(REPORTS_DIR)/5_route_drc.rpt \
        -rd in_def=$< \
        -rd tech_file=$(OBJECTS_DIR)/klayout.lyt \
```

Finishing:

執行 metal fill insertion，目的是把電源線接在一起，並且檢查其設計有無錯誤，以此提升晶片的良率。此階段會回報任務結束時間。

```
$(LOG_DIR)/6_report.log: $(RESULTS_DIR)/6_1_fill.odb $(RESULTS_DIR)/6_1_fill.sdc  
    ($TIME_CMD) $(OPENROAD_CMD) $(SCRIPTS_DIR)/final_report.tcl -metrics $(LOG_DIR)/6_report.json 2>&1 | tee $(LOG_DIR)/6_report.log  
$(RESULTS_DIR)/6_final.def: $(LOG_DIR)/6_report.log
```

▲把 fill.odb 接起來

```
$(WRAPPED_GDSOAS): $(OBJECTS_DIR)/klayout_wrap.lyt $(WRAPPED_LEFS)  
    $(call KLAYOUT_FOUND)  
    $($TIME_CMD) $(K_LAYOUT_CMD) -zz -rd design_name=$(basename $(notdir $@)) \  
        -rd in_def=$(OBJECTS_DIR)/def/$(notdir $(@:$STREAM_SYSTEM_EXT)=def)) \  
        -rd in_files="$(ADDITIONAL_GDSOAS)" \  
        -rd config_file=$(FILL_CONFIG) \  
        -rd seal_file="" \  
        -rd out_file=$@ \  
        -rd tech_file=$(OBJECTS_DIR)/klayout_wrap.lyt \  
        -rd layer_map=$(GDS_LAYER_MAP) \  
        -r $(UTILS_DIR)/def2stream.py) 2>&1 | tee $(LOG_DIR)/6_merge_$(basename $(notdir $@)).log
```

▲使用 Klayout 把 wrapped macros 合併起來

```
---  
GDS_MERGED_FILE = $(RESULTS_DIR)/6_1_merged.$(STREAM_SYSTEM_EXT)  
$(GDS_MERGED_FILE): $(RESULTS_DIR)/6_final.def $(OBJECTS_DIR)/klayout.lyt $(GDSOAS_FILES) $(WRAPPED_GDSOAS) $(SEAL_GDSOAS)  
    $(call KLAYOUT_FOUND)  
    $($TIME_CMD) $(STDBUF_CMD) $(K_LAYOUT_CMD) -zz -rd design_name=$(DESIGN_NAME) \  
        -rd in_def=$@ \  
        -rd in_files="$(GDSOAS_FILES) $(WRAPPED_GDSOAS)" \  
        -rd config_file=$(FILL_CONFIG) \  
        -rd seal_file="$(SEAL_GDSOAS)"' \  
        -rd out_file=$@ \  
        -rd tech_file=$(OBJECTS_DIR)/klayout.lyt \  
        -rd layer_map=$(GDS_LAYER_MAP) \  
        -r $(UTILS_DIR)/def2stream.py) 2>&1 | tee $(LOG_DIR)/6_1_merge.log
```

▲使用 Klayout 把 GDS 接起來

參考資料：

<https://vlsicad.ucsd.edu/Publications/Conferences/371/c371.pdf>

<https://shininglionking.blogspot.com/2013/05/2-ic.html>

[https://en.wikipedia.org/wiki/Placement_\(electronic_design_automation\)](https://en.wikipedia.org/wiki/Placement_(electronic_design_automation))

https://www.youtube.com/watch?v=mUgyantHBhg&ab_channel=Yi-

[YuLiu%28%E5%8A%89%E4%B8%80%E5%AE%87%29](#)