Cell-base Design Flow

Example:

Ternary-Coded-Binarized Neural Network

Author: Kuan-Yu Huang

Advisor: Tsung-Chu Huang

March 31, 2023

Table of Contents

Page

Table of Figure

Page

Table of Tables

Page

Overview

Cell-base design flow is a series of steps that helps IC designers to create an IC. The details of these steps are not covered here as it would take way too much time to explain. This document will only mention the basic content of common sense.

The process of digital IC design is a series of hierarchical decomposition steps.

Table 1 Cell-base Design Flow

Step	Requirements	Tool
Design Concept	Idea	Brain
RTL code	Verilog/VHDL/System Verilog	Text editor
RTL Simulation	Functional Correctness	VCS/NC-Verilog/ModelSim
Logic Synthesis	Meet constraints	Design Compiler
Pre-layout sim* (Gate-level sim)	Functional Correctness	VCS/NC-Verilog/ModelSim
APR	Meet constraints	ICC
Post-layout-sim**	Functional Correctness	VCS/NC-Verilog/ModelSim
Verification	DRC/LVS	Calibre
*Add .sdf file from DC ** Add .sdf file from ICC		

DEMO Video:

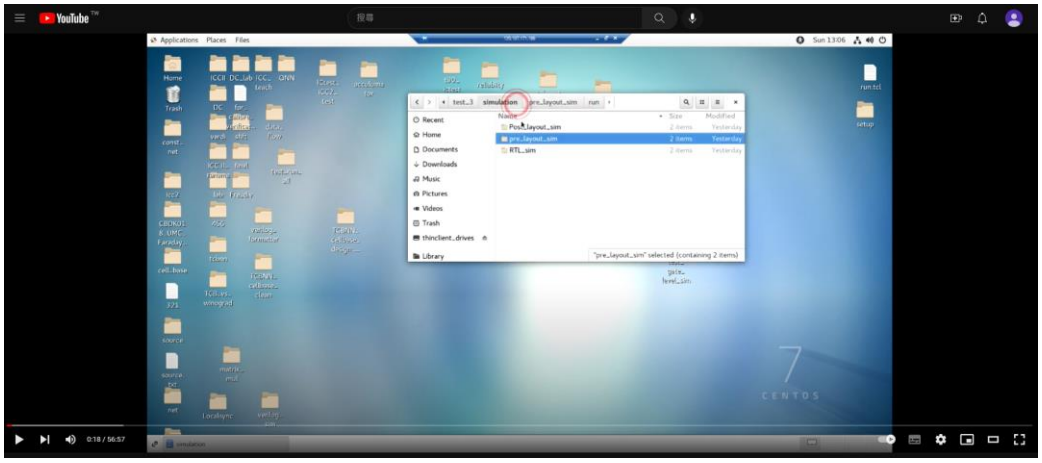


Figure 1 Video Demo

https://www.youtube.com/watch?v=_br-QIyvIVY

How to implement an idea using cell-base front-end design flow?

An idea could be a novel adder or an accelerator within a large design. When designing a circuit, things must be kept in mind. First of all, the designed circuit has to be synthesizable. Namely, it can be analyzed by Design Compiler. The common rule of synthesizable design is that the designer can visualize the circuit in mind.

```
always @ (posedge clk)
begin
  if (reset) begin
      a=0;
      ……..
  end
  else  begin
      a=c+d*b;
       …….
  end
end
```

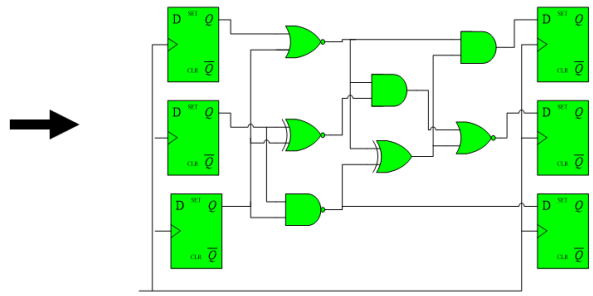


Figure 2 Logic Synthesis [1]

The other thing is to verify the functionality of this design with the simulators mentioned above, e.g., VCS, ModelSim.

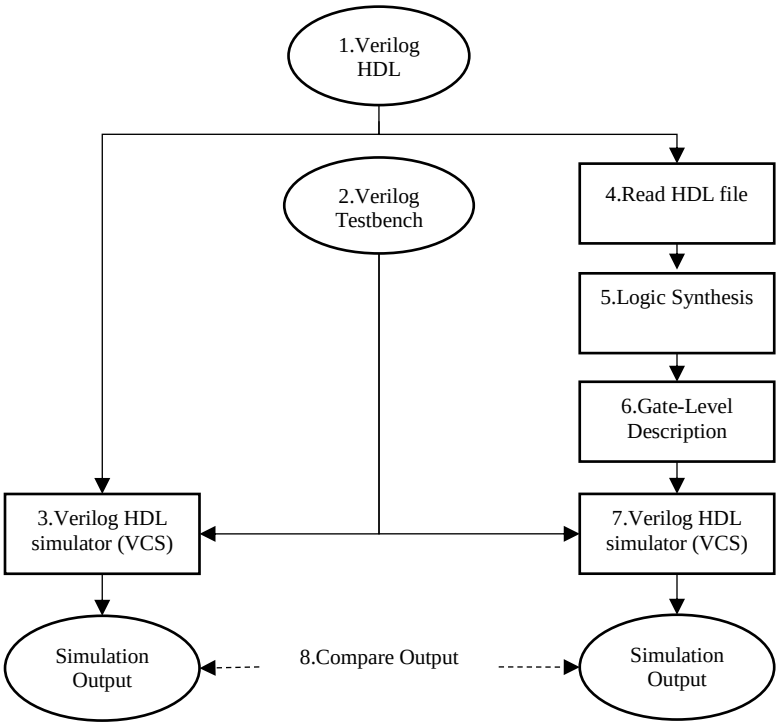


Figure 3 Front-end Flow

RTL code

Quote: *"Register Transfer Level" code is a smaller subset of the full range of HDL code. RTL describes circuits at a level similar to the design description on a schematic: flip-flops activated by fully-specified clocks, and combinatorial logic (ranging from simple gates to large multipliers) between the flip-flops.[3]*

Design a circuit with HDL language and Design compiler will synthesis your design with constraints.

RTL Simulation

Here are commands that help users to run RTL simulation using VCS in terminal window.

VCS command-RTL sim

Remember to source tools from tool table to use EDA tool.

Example:

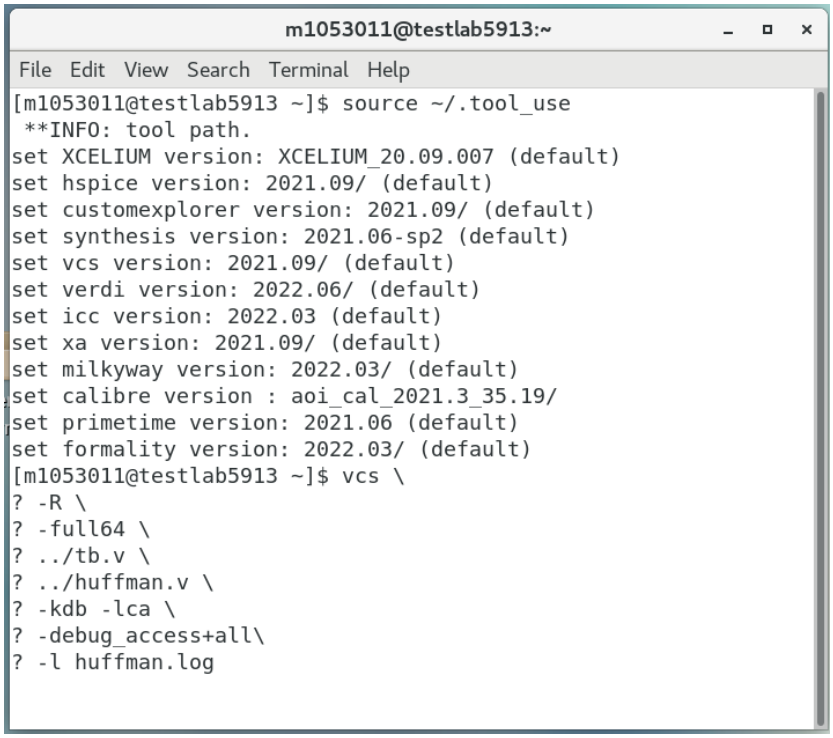


Figure 4 Command Example

Table 2 RTL Commadn

<table>
<tr><td>RTL sim</td><td></td></tr>
<tr><td>vcs \
../your_test_bench.v \
../your_design.v \</td><td>Include Verilog file</td></tr>
<tr><td rowspan="5">-R \
-full64 \
-kdb -lca \
-debug_access+all \
+v2k
- l output_log_file_name.log</td><td>-R: compile and simulation</td></tr>
<tr><td>-full64:64bit operating system</td></tr>
<tr><td>-kdb -lca: generate Knowledge Data Base at local file which can help improve compile time.</td></tr>
<tr><td>- debug_access+all:enable all debug function</td></tr>
<tr><td>+v2k: verilog version
-l:output log file</td></tr>
<tr><td>-xprop=tmerge -reprot=xprop \</td><td>Enable X propagation function can help trace unknow signal</td></tr>
<tr><td>+define+RTL_SIM</td><td>User define function in Verilog file</td></tr>
<tr><td colspan="2">Add the following text in testbench to define RTL_SIM

```
`ifdef RTL_SIM
begin
    $display("*****************RTL Simulation begins dump_waveform*****************\n\n");
    $fsdbDumpfile("/users/m1053011/Desktop/tcbnn/RTL_pre_simulation/rtl_sim_axis_top.fsdb");
    $fsdbDumpMDA;
    $fsdbDumpvars;
end
`endif
```

</td></tr>
<tr><td colspan="2">To use the following command, you must add “-kdb -lca” in vcs command to enable knowledge data base function. It will automatically read your source code and wave in Verdi window.</td></tr>
<tr><td>verdi -dbdir simv.daidir -ssf pre_sim_axis_top_gate_level.fsdb&</td><td>Open Verdi to view waveform and debug function.</td></tr>
</table>

X propagation:

Google Verdi x propagation[4][5].

Table 3 X propagation Command

X propagation in VCS
#vcs -xprop[=tmerge|xmerge|xprop_config_file] otehr_vcs_option # #tmerge:default,more realistic #xmerge:pessimistic or worst case merge # #-report=xprop # :Generate an Xprop instrucmentation reprort named xprop_config.report ,continue the simulation #-report=xprop+exit # :Generate report, and terminate simulation

VCS command-Gate-Level sim

Table 4 Gate-level Command

<table>
<tr><td>Gate-level sim</td><td></td></tr>
<tr><td colspan="2">Remember to add sdf_annotate in Testbench to simulate cell delay from DC:
Example:
$sdf_annotate("/users/m1053011/Desktop/tcbnn/dc/axis_top_ver1126_syn.sdf",
DUT1);</td></tr>
<tr><td>vcs /users/m1053011/Desktop/tcbnn/verilog/tb_for_combine.v\
/users/m1053011/Desktop/tcbnn/dc/axis_top_ver1126_syn.v \</td><td>Tb:testbench
Syn.v:gate level netlist file from DC</td></tr>
<tr><td>-v
/usr/cad/Library/CBDK018_UMC_Faraday_v1.0/CIC/Verilog/fsa0m_a_generic_core_30.lib.src
/usr/cad/Library/CBDK018_UMC_Faraday_v1.0/CIC/Verilog/fsa0m_a_t33_generic_io_30.lib. src \</td><td>U18 core library and IO library for Verilog simulation</td></tr>
<tr><td rowspan="3">-R -full64 +maxdelays -negdelay +neg_tchk -kdb -lca -l post_sim_gate_level.log -debug_access+all +v2k\</td><td>+maxdelays -negdelay +neg_tchk: timing check option, in this case, simulate max delay and negative delay and check negative timing check.</td></tr>
<tr><td>-kdb -lca: generate Knowledge Data Base which can help improve compile time.
- debug_access+all:enable all debug function</td></tr>
<tr><td>+v2k: verilog version</td></tr>
<tr><td>+define+POST_SIM</td><td>User define function in Verilog file</td></tr>
<tr><td colspan="2">

```
`ifdef POST_SIM
begin
    $display("****************POST_SIM Gate Level Simulation begins dump_waveform****************\n\n");
    $fsdbDumpfile("/users/m1053011/Desktop/tcbnn/gate_level_sim/post_sim/post_sim_axis_top_gate_level.fsdb");
    $fsdbDumpMDA;
    $fsdbDumpvars;

    $sdf_annotate("/users/m1053011/Desktop/tcbnn/dc/axis_top_ver1126_syn.sdf",
        DUT1);
end
`endif
```

</td></tr>
<tr><td colspan="2">verdi -dbdir simv.daidir -ssf post_sim_axis_top_gate_level.fsdb&</td></tr>
</table>

How to use Design Ware Library?

Example: DW_sqrt

In some cases, you may want to do complex calculations, for example, sin, cos or sqrt etc. It will take a lot of time to implement this function by yourself. Luckily, Synopsys provides Design Ware Library to everyone who has license to use pre-design IPs which help you develop your design easily.

In this example, DW_sqrt, a module to do square root, will be used to do RTL sim in VCS and logic synthesis in Design Compiler.

Document location:

/usr/cad/synopsys/synthesis/2021.06-sp2/dw/doc

How to use DW in verilog example:

/usr/cad/synopsys/synthesis/2021.06-sp2/dw/examples

Verilog simulation file:

/usr/cad/synopsys/synthesis/2021.06-sp2/dw/sim_ver

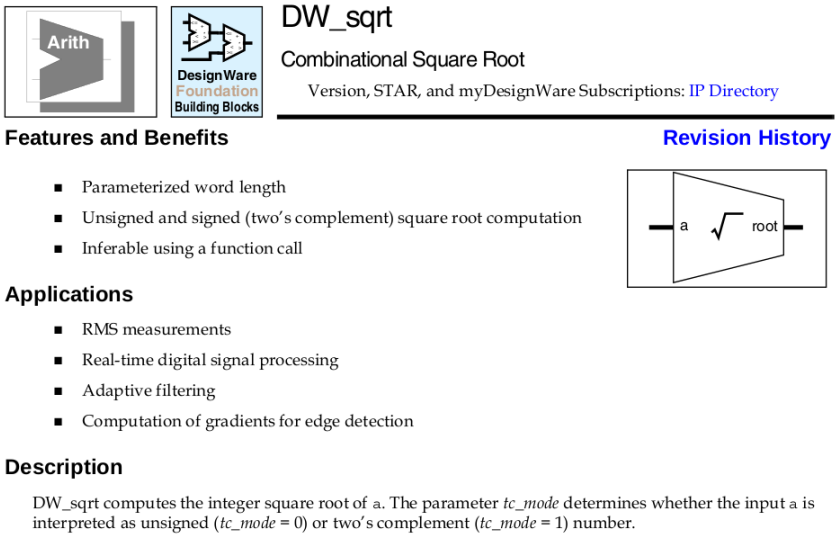


Figure 5 Design Ware Manual

Function description in doc

DW_sqrt_inst.v

usr > cad > synopsys > synthesis > 2021.06-sp2 > dw > examples > DW_sqrt_inst.v > {} DW_sqrt_inst

```
module DW_sqrt_inst (radicand, square_root);
  parameter radicand_width = 8;
  parameter tc_mode        = 0;

  input  [radicand_width-1 : 0]       radicand;
  output [(radicand_width+1)/2-1 : 0] square_root;
  // Please add +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator
  // command line (for simulation).

  // instance of DW_sqrt
  DW_sqrt #(radicand_width, tc_mode)
    U1 (.a(radicand), .root(square_root));
endmodule

```

Figure 6 Example in Verilog

```
wire [10:0] square_root;
DW_sqrt #(20, 0) DW_sqrt_U1 (
    .a(input_value),
    .root(square_root)
);
```

Figure 7 Implement it in your code

```
#DesignWare Library
vcs \
-R \
+v2k \
-full64 \
../geo_tb.v \
../geofence.v \
-debug_access+all \
-kdb -lca \
-l geo.log \
-y /usr/cad/synopsys/synthesis/2021.06-sp2/dw/sim_ver/ +libext+.v \
+incdir+/usr/cad/synopsys/synthesis/2021.06-sp2/dw/sim_ver/+ \
+define+RTL_SIM
```

Figure 8 Command Example

Add -y … and +incdir… in your VCS command.

.synopsys_dc.setup

users > m1053011 > Desktop > verilog_sim > ic_contest > 2021_grad_cell > 2021_grad_cell > dc > .synopsys_dc.setup

```
set company "CIC"
set designer "Student"
set search_path      ". /users/m1053011/Desktop/verilog_sim/ic_contest/CBDK_IC_Contest_v2.5/SynopsysDC/db/ \
/usr/cad/synopsys/synthesis/2021.06-sp2/dw/sim_ver $search_path"
set target_library   "slow.db"
set link_library     "* $target_library dw_foundation.sldb"
set symbol_library   "generic.sdb"
set synthetic_library "dw_foundation.sldb"
```

Figure 9 .synopsys_dc.setup

Moreover, add” /usr/cad/synopsys/synthesis/2021.06-sp2/dw/sim_ver” to search_path in .synopsys_dc.setup

Waveform Result

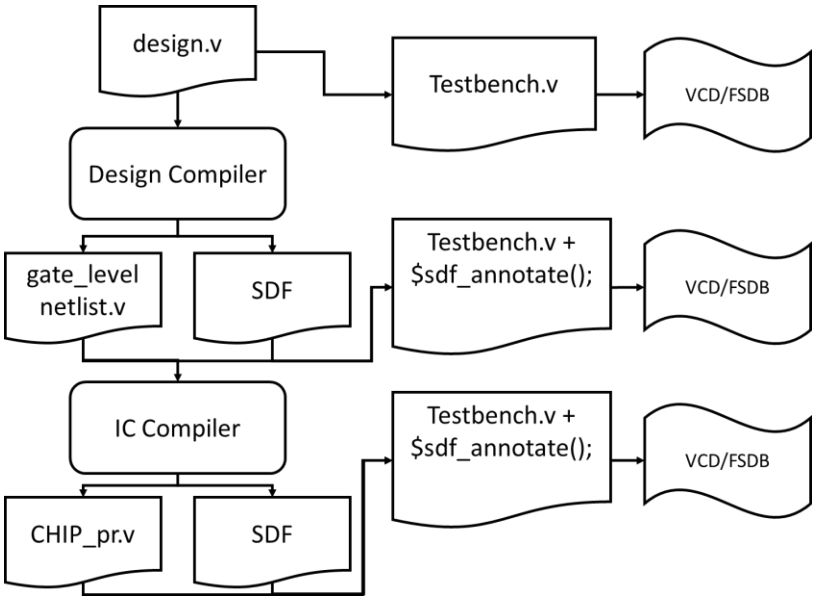
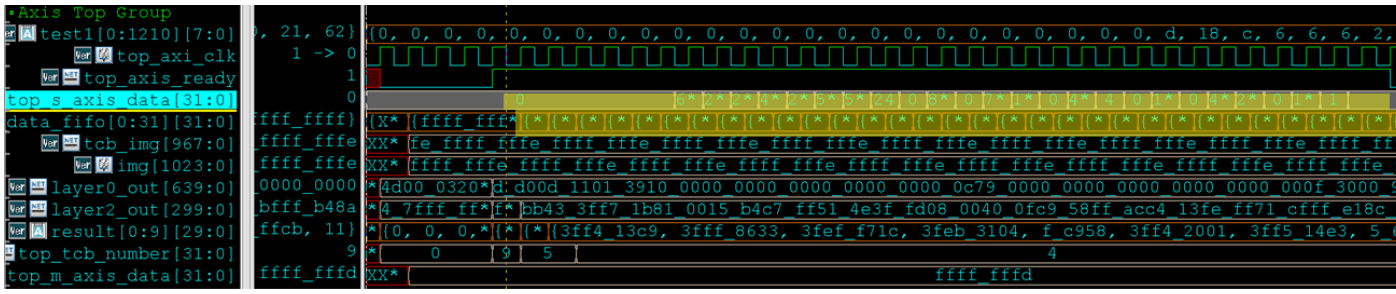
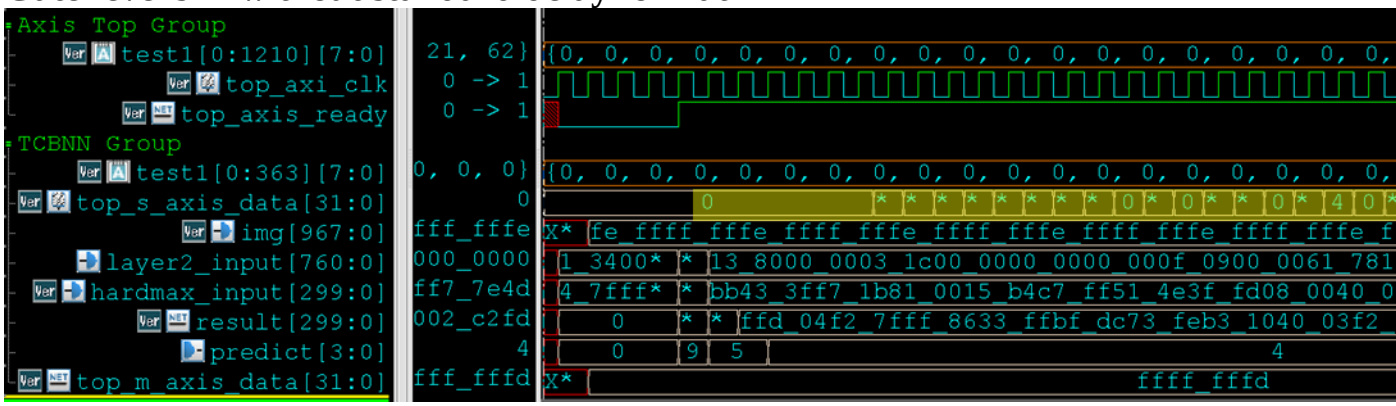


Figure 10 Waveform Result Flow

RTL sim



Gate-level sim without standard delay format



Gate -level sim with standard delay format

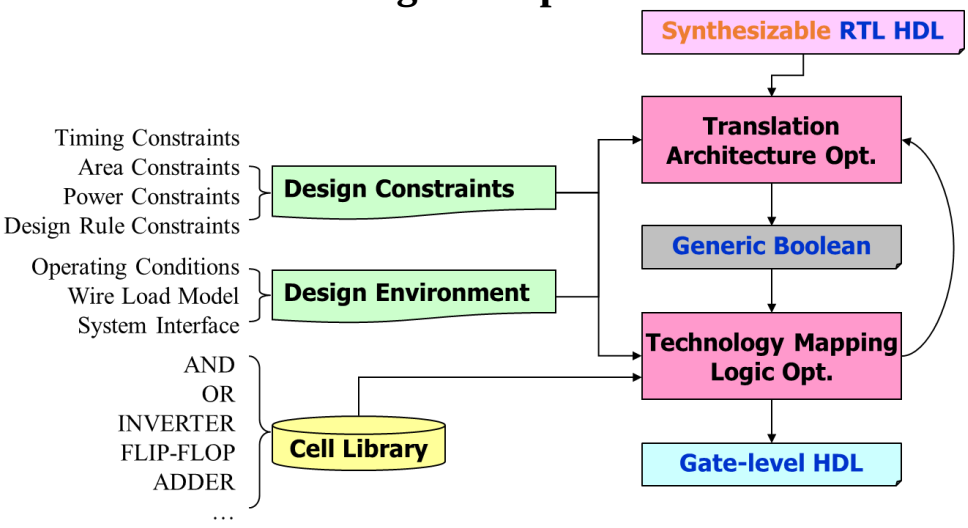


Figure 11 Logic Synthesis [1]

The key idea in Design Compiler is logic synthesis, seen in Figure 2. Design Compiler will map RTL (Register Transfer Level) HDL (Hardware Description Language) according to the design constraints and environment.

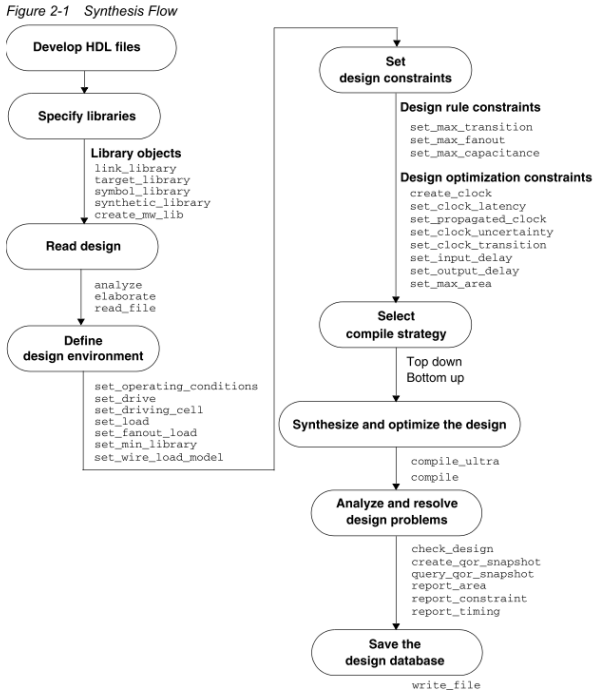


Figure 12 Synthesis Flow[2]

Design Setup

In. synopsys_de.setup, we will set the path where the logic synthesis library locates.

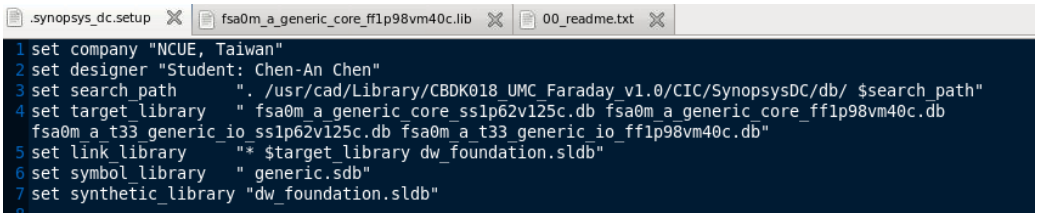


Figure 13.synopsys_dc.setup

.synopsys_dc.setup

1.設定作者,學校

```
set company "NCUE, Taiwan"
set designer "Student: xxx"
```

2.設定 DC 搜尋路徑

```
set search_path ".
/usr/cad/Library/U18/CBDK018_UMC_Faraday_v1.0/CIC/SynopsysDC/db/
$search_path"
```

3.設定製程用的 library

```
set target_library  " fsa0m_a_generic_core_ss1p62v125c.db
fsa0m_a_generic_core_ff1p98vm40c.db fsa0m_a_t33_generic_io_ss1p62v125c.db
fsa0m_a_t33_generic_io_ff1p98vm40c.db"
```

4. dw_foundation.sldb 是今天所使用的 std cell library

```
set link_library    "* $target_library dw_foundation.sldb"
```

5.顯示在 schematic 介面的邏輯圖案 library

```
set symbol_library  " generic.sdb"
```

6.合成用的 library

```
set synthetic_library "dw_foundation.sldb"
```

7.個人偏好設定

```
set verilogout_no_tri true
set hdlin_enable_presto_for_vhdl "TRUE"
set sh_enable_line_editing true
history keep 100
alias h history
```

Define design environment

- Beware the defaults are not realistic conditions
  - Input drive is not infinite
  - Capacitive loading is usually not zero
  - Consider process, voltage, temperature (**PVT**) variation
- The operating environment affects the components selected from target library and timing through your design
- The real world environment you define describes the conditions that the circuit will operate within

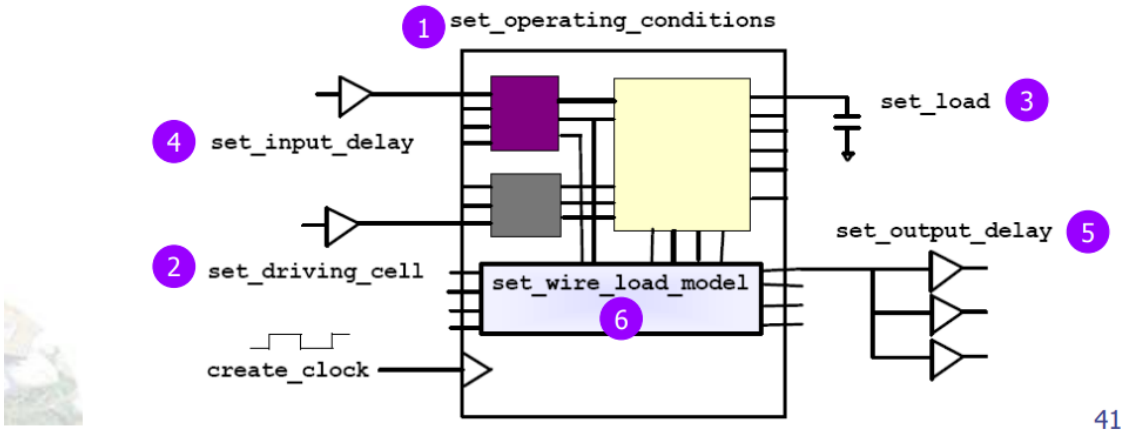


Figure 14 Design Environment

1. Attributes > Operating Environment > Operating Condition ” 設定 Operating Condition

```
Unix% set_operating_conditions -min_library fsa0m_a_generic_core_ff1p98vm40c -min BCCOM  -max_library fsa0m_a_generic_core_ss1p62v125c -max WCCOM
```

2. 設定 Input drive strength 與 output load

```
Unix% set_driving_cell -library fsa0m_a_t33_generic_io_ss1p62v125c -lib_cell XMC -pin {O} [all_inputs]
```

Set design constraints

Figure 8-2 Major Design Optimization Constraints

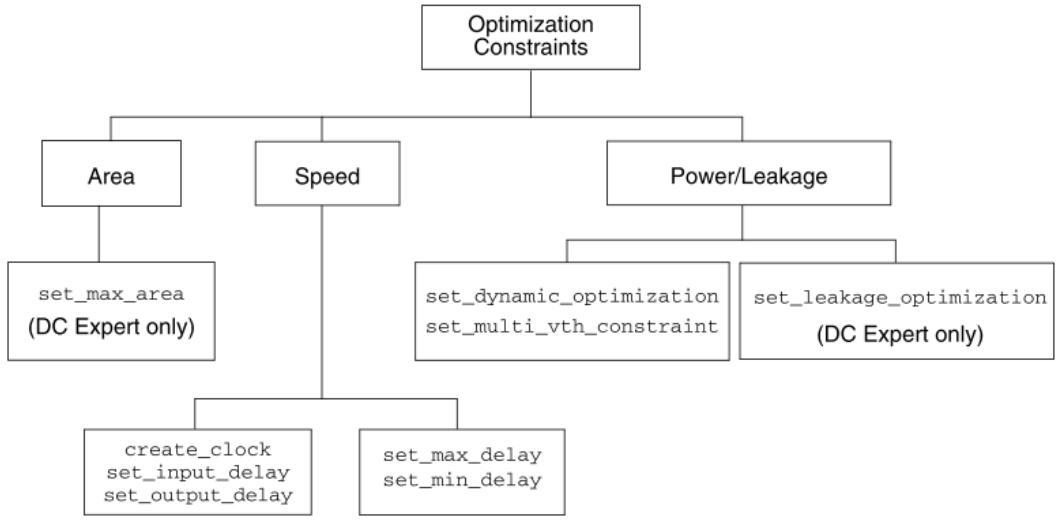


Figure 15 Optimization Constraints[2]

Timing

3. 設定 1.Period & 2.Waveform

```
Unix% create_clock -period 10  [get_ports clk]  -waveform {0 5}
Unix% set_dont_touch_network  [get_clocks clk]
Unix% set_fix_hold  [get_clocks clk]
```

4. 設定 3.Clock skew

```
Unix% set_clock_uncertainty  0.1  [get_clocks clk]
```

5. 設定 4&5. Clock/Source Latency

```
Unix% set_clock_latency  0.5  [get_clocks clk]
```

6. 設定 6&7. Input/Clock Transition

```
Unix% set_input_transition  0.5  [all_inputs]
Unix% set_clock_transition  0.1  [all_clocks]
```

Area

7. 設定欲合成的面積目標與 Design Rule Constraint

```
Unix% set_max_area 0
Unix% set_max_capacitance 0.5 [all_inputs]
Unix% set_max_transition 0.5 [all_inputs]
```

Other Constraints

8. 設定 Ideal network

```
Unix% set_ideal_network  [get_ports clk]
Unix% set_ideal_network  [get_ports rst]
```

9. 設定 fanout

```
Unix% set_max_fanout 6 [all_inputs]
```

10. 設定 High Fan-out Net 與呼叫同一名稱的子電路複製一份並重新命名

```
Unix% set high_fanout_net_threshold 0
Unix% uniquify
```

11. 處理 assign 語法

```
Unix% set_fix_multiple_port_nets -all -buffer_constants [get_designs *]
```

12. 設定禁止 DC 處理 Pad

```
Unix% set_dont_touch [get_cell ipad_*]
Unix% set_dont_touch [get_cell opad_*]
```

Compile & Synthesize and optimize the design

13. 以-boundary_optimization 進行合成

```
Unix% compile -boundary_optimization
Or compile_ultra
```

Analyze and resolve design problem

14. 檢查 Timing 是否符合 Constraint

```
Unix% check_timing
```

Save the design database

15. 設定等一下匯出檔案的規則

```
Unix% set bus_inference_style {%s[%d]}
Unix% set bus_naming_style {%s[%d]}
Unix% set hdlout_internal_busses true
Unix% change_names -hierarchy -rule verilog
Unix% define_name_rules name_rule -allowed {a-z A-Z 0-9 _} -max_length 255 -type cell
Unix% define_name_rules name_rule -allowed {a-z A-Z 0-9 _[]} -max_length 255 -type net
Unix% define_name_rules name_rule -map {{"\\*cell\\*" "cell"}}
Unix% define_name_rules name_rule -case_insensitive
Unix% change_names -hierarchy -rules name_rule
```

16. 匯出檔案

```
Unix% write -format verilog -hierarchy -output "chip_syn.v"
Unix% write -format ddc   -hierarchy -output "chip_syn.ddc"
Unix% write_sdc chip_syn.sdc
Unix% write_sdf -version 1.0 -context verilog -load_delay net chip_syn.sdf
```

17. Report 面積、功耗、Timing(Setup/Hold) 以及 Quality of Results file

```
Unix% report_area > area.log
Unix% report_timing > timing.log
Unix% report_power > power.log
Unix% report_qor   >  CHIP_syn.qor
```

How to get precise Power Report?

In general, the default power report is not accurate at all. To get realistic power report using prime time is the best choice. However, Design Compiler can also help user to get power result by reading saif (Switching Activity Interchange Format) file to estimate power consumption.

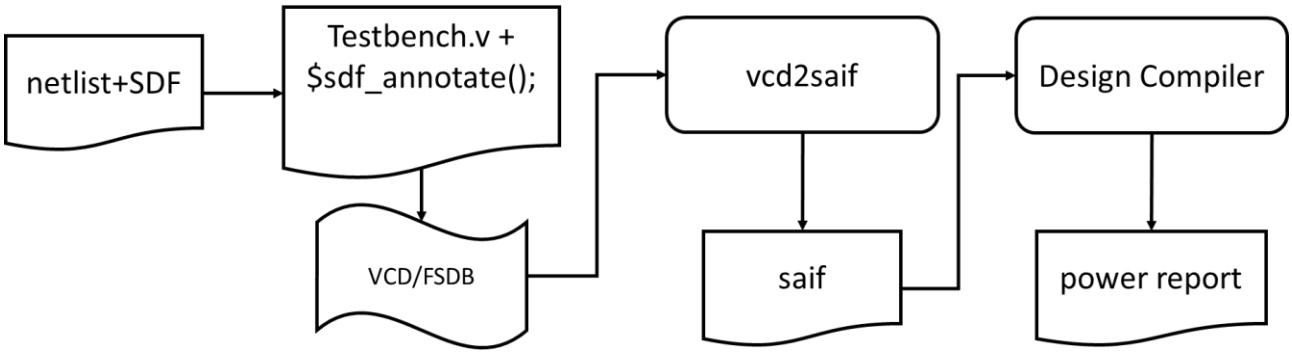


Figure 16 DC power report flow

First, get the gate-level netlist and sdf to do gate-level simulation.

1. Write out gate-level netlist

Tool	Format	Command
Design_vision	.v	write -format Verilog -hierarchy -output chip_syn.v

2. Get SDF

Tool	Format	Command
Design_vision	.sdf	write_sdf -version 1.0 -load_delay net chip_syn.sdf

3. Add $sdf_annotate in testbench
   $sdf_annotate("the sdf file name", the top module instance name);

```
$sdf_annotate("/users/m1053011/Desktop/tcbnn/icc_ultra_ver/run/CHIP_pr.sdf", DUT1);
```

4. Simulate using VCS or NC-Verilog to dump vcd or fsdb file

Tool	Format	Command
NC Verilog	.v	Ncverilog tb.v chip_syn.v -v yout_sumulation_model.v +access+r
VCS	.v	vcs -R tb.v chip_syn.v -v yout_simulation_model.v

```
vcs \
your_testbench.v\
your_gate_level_netlist.v \
-v /usr/cad/Library/CBDK018_UMC_Faraday_v1.0/CIC/Verilog/fsa0m_a_generic_core_30.lib.src \
/usr/cad/Library/CBDK018_UMC_Faraday_v1.0/CIC/Verilog/fsa0m_a_t33_generic_io_30.lib.src \
-R \
-full64 +maxdelays \
-negdelay +neg_tchk \
-kdb -lca \
-debug_access+all +v2k\
-l post_sim_gate_level.log \
+define+DUMP_VCD+POST_SIM
```

Figure 17 Run Gate-sim

```
`ifdef POST_SIM
        begin
            $display(
                "*****************POST_SIM Gate Level Simulation begins dump_waveform**********
            $fsdbDumpfile("post_sim_axis_top_gate_level.fsdb");
            $fsdbDumpMDA;
            $fsdbDumpvars;

            $sdf_annotate("/users/m1053011/Desktop/tcbnn/dc/axis_top_ver1126_syn.sdf", DUT1);
        end
`endif
```

Figure 18Testbench dump fsdf file

```
`ifdef DUMP_VCD
        begin
            $display("*****************Dumping VCD waveform****************\n\n");
            $dumpfile("axis_top_ver1126_syn.vcd");
            $dumpvars(0, DUT1);
        end
`endif
```

Figure 19 testbench dumps vcd file

5. Get saif (Switching Activity Interchange Format) from vcd file

```
Unix%          vcd2saif -input your_design.vcd -output your_design.saif
```

```
[m1053011@testlab5913 gate_level_sim]$ vcd2saif -input axis_top_ver1126_syn.vcd  -output axis_top_ver1126_syn.saif
VCD to SAIF translator version S-2021.06 Synopsys, Inc.
direct mapping all VCD instances
processing header of VCD file: axis_top_ver1126_syn.vcd
processing value changes of VCD file: axis_top_ver1126_syn.vcd
generating backward SAIF file: axis_top_ver1126_syn.saif
[m1053011@testlab5913 gate_level_sim]$
```

Figure 20 v2saif result

6. Get more precise power analysis (the best option is to use prime time) from Design Compiler

No saif Power Result: 36mW.

```
Global Operating Voltage = 1.62
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =  10.9812 mW   (31%)
  Net Switching Power  =  24.7261 mW   (69%)
                          ---------
Total Dynamic Power    =  35.7073 mW  (100%)

Cell Leakage Power     = 387.1038 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register           5.4552           0.6716        1.5324e+07             6.1422  (  17.02%)
sequential     5.4218e-04       8.4394e-04        7.7553e+03         1.3939e-03  (   0.00%)
combinational      5.5255          24.0529        3.7177e+08            29.9499  (  82.98%)
------------------------------------------------------------------------------------------------
Total            10.9812 mW       24.7254 mW     3.8710e+08 pW        36.0934 mW
```

Figure 21 No saif Power Result

Tool	Command
Design_vision	read_saif -input axis_top_ver1126_syn.saif -instance tb_for_comebine/DUT1

```
design_vision> read_saif -input axis_top_ver1126_syn.saif -instance_name tb_for_combine/DUT1
Warning: There are 48 objects not found during annotation. (PWR-452)
1
```

Report_power with SAIF power report

```
                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  At
------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register           5.8151           0.9260        1.6362e+07             6.7574  (   8.58%)
sequential         0.0000           0.0000            0.0000             0.0000  (   0.00%)
combinational     13.3379          58.2791        3.4849e+08            71.9676  (  91.42%)
------------------------------------------------------------------------------------------------
Total             19.1529 mW       59.2050 mW     3.6485e+08 pW        78.7250 mW
1
design_vision>
```

Total power is 78mW. It increased 42mW from original power result.

Area

There are two ways of reporting the cost of area.

1. Number of the cell from Design Compiler report
2. Gate count by basic gate element (NAND)

Number of the cell from Design Compiler report

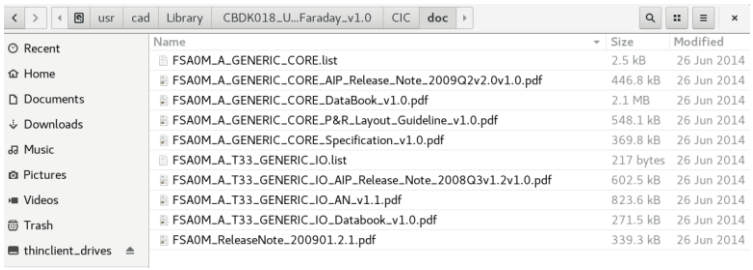
```
Number of combinational cells:         193961
Number of sequential cells:              3863
Number of macros/black boxes:               0
Number of buf/inv:                      34531
Number of references:                      26

Combinational area:            7894063.536078
Buf/Inv area:                   293537.459295
Noncombinational area:          231766.408932
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:               8125829.945010
Total area:                 undefined
1
```

Gate count by basic gate element (NAND)

First of all, it is essential to know the location of the U18 document.

Document Location:

/usr/cad/Library/CBDK018_UMC_Faraday_v1.0/CIC/doc



1.1 General Description

The FSA0M_A library is a 0.18 µm standard cell library tailored by using UMC's 0.18 µm Mixmode/RFCMOS process. It is optimized for the applications requiring ultra high density and low power consumption while keeping high performance.

The 9-track (5.04 µm) cell height has the industry's smallest cell layout area (110k gates/mm^2). The optimized drive strength of the library provides an extensive library database that is easy to manage and use. These rich and complex cells help you to shorten the time spent in your design by employing the most up-to-date synthesis tools.

Table 1-3. Physical Attributes of Core Cell

General Physical Attribute	Description	Characteristic
Core cell height	-	5.04 µm
Vertical routing track	-	9 tracks
Vertical routing grid	-	0.56 µm
Horizontal routing grid	-	0.62 µm

Figure 22 Physical Attributes of Core Cell (1)

Table 1-5. Physical Specification of Core Cell

Vertical grid	0.56 μm
Horizontal grid	0.62 μm
Top metal grid	0.88 μm
Cell height	5.04 μm (9 vertical grids)
Cell power/ground rail width for the follow-pins routes (M2)	0.76 μm
Layout resolution	0.01 μm (0.001 μm for the bend gate)
Substrate/Well contact	Each cell has at least 1 substrate/well contact.

Figure 23 Physical Attributes of Core Cell (2)

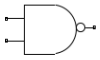
NAND

ND2 *FARADAY STANDARD CELL LIBRARY FSA0M_A*

Group Name : ND2

Function : 2 Input NAND

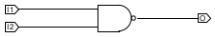
Symbol



Truth Table

I1	I2	O
1	1	0
OTHERS		1

Schematic



Pin Order O I1 I2

Input Capacitance (pf) & Maximum Loading (pf) & Power Consumption (uW/MHz)

Version	Input Capacitance		Maximum Loading	Power Consumption
	I1	I2	O	O
ND2S	0.0036	0.0041	0.2111	0.009
ND2	0.0044	0.0050	0.2853	0.011
ND2P	0.0106	0.0121	0.6107	0.026
ND2T	0.0179	0.0200	0.9792	0.041
ND2F	0.0234	0.0257	1.251	0.054

Figure 24 NAND Cell (1)

AC Characteristics (Temp=25.0°C Core Voltage=1.8V Process=Nominal Delay Unit=ns)

Version	Cell Unit	Output Load											
ND2S	3	Path	0.0020 pf		0.0053 pf		0.0141 pf		0.0374 pf		0.0994 pf		0.2640 pf	
			tplh	tphl	tplh	tphl	tplh	tphl	tplh	tphl	tplh	tphl	tplh	tphl
		I1-O	0.053	0.032	0.070	0.045	0.109	0.073	0.208	0.138	0.466	0.301	1.149	0.735
		I2-O	0.065	0.035	0.081	0.046	0.119	0.073	0.217	0.136	0.475	0.300	1.158	0.734
ND2	3	Path	0.0020 pf		0.0058 pf		0.0166 pf		0.0478 pf		0.1375 pf		0.3960 pf	
			tplh	tphl	tplh	tphl	tplh	tphl	tplh	tphl	tplh	tphl	tplh	tphl
		I1-O	0.047	0.034	0.062	0.047	0.099	0.078	0.197	0.157	0.474	0.374	1.272	0.999
		I2-O	0.057	0.036	0.071	0.048	0.107	0.078	0.204	0.155	0.482	0.373	1.279	0.998

Figure 25 NAND Cell (2)

We can get the size of NAND gate is:

$$\text{NAND GATE SIZE} = 5.04 \times (3 \times 0.62) = 9.3744\ um^2$$

In the area report from Design Compiler, the area of the design is 8125829 um^2. The gate count of the design is :

$$\text{GATE COUNTs} = \frac{8125829}{9.3744} = 866810 \text{ gates}$$

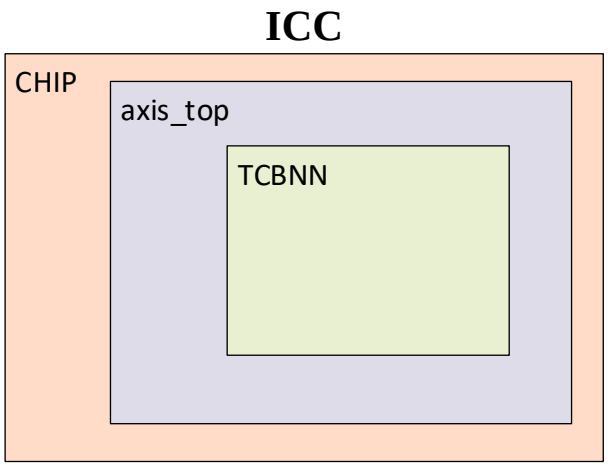


Figure 26 Design Hierarchy

Design Flow Overview

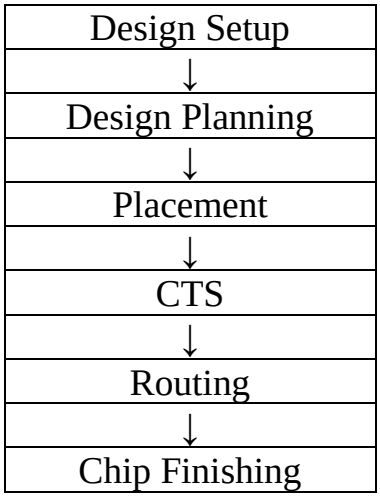


Figure 27 APR Overview

Above figure is the overview of cell-base back-end flow.

The files before APR

Create ‘Physical Only” I/O Cells

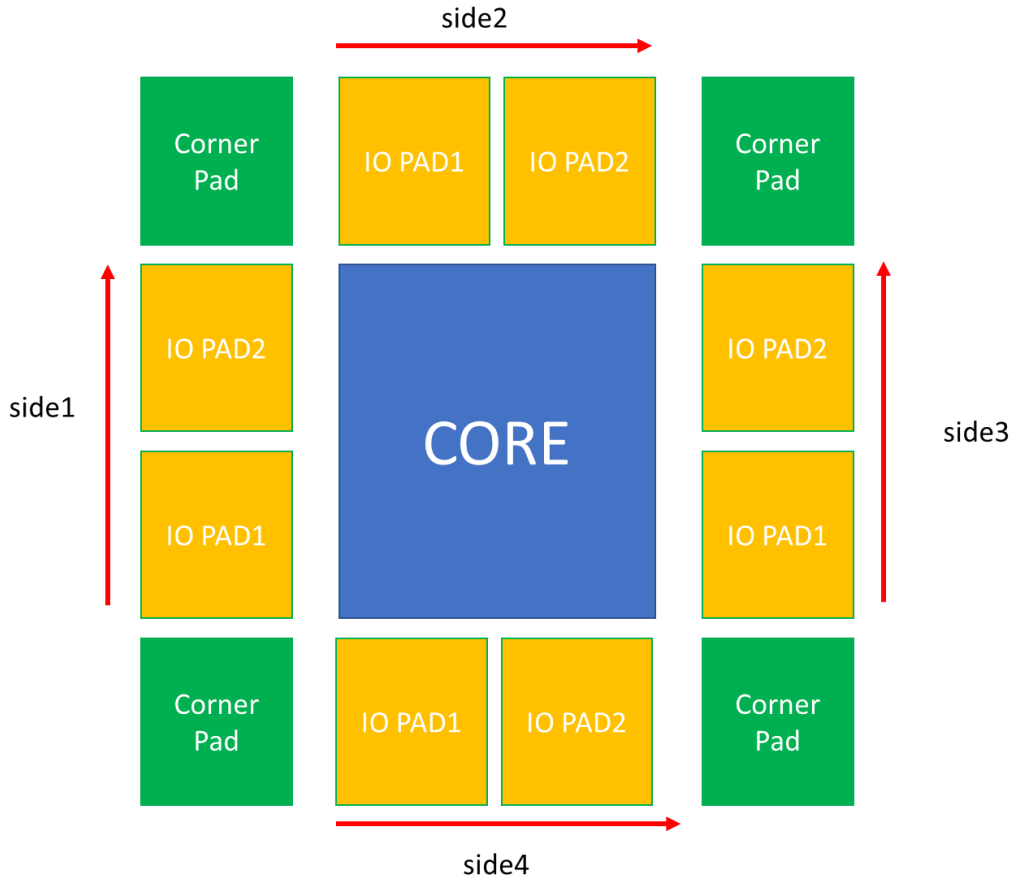


Figure 28 IO PAD Order

Physical cells like input output port are not part of design in Verilog file. We need to define these physical cells in CHIP.v to decide the port order in layout.

CHIP.v contains the IO ports information. The design below is pad limit. Therefore, the IO port cells are XMC and YA2GSC.

users > m1053011 > Desktop > tcbnn > verilog > CHIP.v > {} CHIP

```v
module CHIP(
input s_axis_last ,
output m_axis_valid ,
input s_axis_valid ,
input axi_clk ,
output m_axis_last ,
input axi_reset_n ,
output s_axis_ready ,
input [31:0] s_axis_data ,
output [31:0] m_axis_data ,
input m_axis_ready
);
wire i_s_axis_last;
wire i_m_axis_valid;
wire i_s_axis_valid;
wire i_axi_clk;
wire i_m_axis_last;
wire i_axi_reset_n;
wire i_s_axis_ready;
wire [31:0] i_s_axis_data;
wire [31:0] i_m_axis_data;
wire i_m_axis_ready;
axis_top_ver1126 DUT (.s_axis_last(i_s_axis_last),.m_axis_valid(i_m_axis_valid)

XMC ipad_in0(.O(i_s_axis_last), .I(s_axis_last),.PU(n_Logic0_), .PD(n_Logic0_),

XMC ipad_in1(.O(i_s_axis_valid), .I(s_axis_valid),.PU(n_Logic0_), .PD(n_Logic0_
XMC ipad_in2(.O(i_axi_clk), .I(axi_clk),.PU(n_Logic0_), .PD(n_Logic0_), .SMT(n_

XMC ipad_in3(.O(i_axi_reset_n), .I(axi_reset_n),.PU(n_Logic0_), .PD(n_Logic0_),
```

Figure 29 CHIP.v

create_phy_cell.tcl

users > m1053011 > Desktop > tcbnn > icc > scripts > create_phy_cell.tcl

```
create_cell {cornerLL cornerLR cornerUL cornerUR}
create_cell {core_vcc1 core_vcc2} VCCKC
create_cell {core_gnd1 core_gnd2} GNDKC
create_cell {io_vcc1 io_vcc2} VCC3IOC
create_cell {io_gnd1 io_gnd2} GNDIOC
```

XMC	Programmable 3.3V LVTTL input buffer, pad limit
XMD	Programmable 3.3V LVTTL input buffer, core limit
YA2GSC	Programmable 2 ~ 16mA 3.3V LVTTL output buffer, pad limit
YA2GSD	Programmable 2 ~ 16mA 3.3V LVTTL output buffer, core limit
ZMA2GSC	Programmable 2 ~ 16mA 3.3V LVTTL bi-direction buffer, pad limit
ZMA2GSD	Programmable 2 ~ 16mA 3.3V LVTTL bi-direction buffer, core limit
VCC3IOC	Power supply for 3.3V input buffers and output buffers, pad limit
VCC3IOD	Power supply for 3.3V input buffers and output buffers, core limit
VCCKC	Power pad for internal core cell, pad limit
VCCKD	Power pad for internal core cell, core limit

Figure 30 Physical Cells

Core and IO pad power

Recommend website:

皓宇的筆記: https://timsnote.wordpress.com/2017/08/09/pad-selection/

All documents can be found at /usr/cad/Library/CBDK018_UMC_Faraday_v1.0/CIC/doc

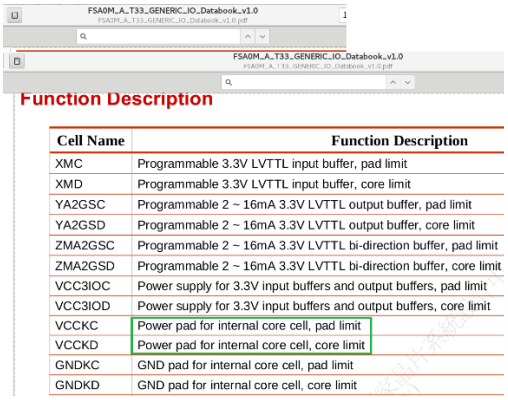
There are two things on chip consume power. The core on the chip, and the Output pads around the core.

Core power:

DC Power report: 78.725mW

The working voltage in U18 is 1.8v. According to the power formula $I = \frac{P}{V}$, the current we need is $\frac{78.725mW}{1.8} = 43.736\text{mA}$.

Read the U18 manual in doc file, a core pad provides 34.1 mA.



Function Description

Cell Name	Function Description
XMC	Programmable 3.3V LVTTL input buffer, pad limit
XMD	Programmable 3.3V LVTTL input buffer, core limit
YA2GSC	Programmable 2 ~ 16mA 3.3V LVTTL output buffer, pad limit
YA2GSD	Programmable 2 ~ 16mA 3.3V LVTTL output buffer, core limit
ZMA2GSC	Programmable 2 ~ 16mA 3.3V LVTTL bi-direction buffer, pad limit
ZMA2GSD	Programmable 2 ~ 16mA 3.3V LVTTL bi-direction buffer, core limit
VCC3IOC	Power supply for 3.3V input buffers and output buffers, pad limit
VCC3IOD	Power supply for 3.3V input buffers and output buffers, core limit
VCCKC	Power pad for internal core cell, pad limit
VCCKD	Power pad for internal core cell, core limit
GNDKC	GND pad for internal core cell, pad limit
GNDKD	GND pad for internal core cell, core limit

Table 3-2. Current-Carrying Capacity of Power/Ground Pad

Pad Name of Metal Layer	Vertical Metal Laye
	M1 + M2
VCCKC	34.1
VCCKD	55.6
VCC3IOC	34.1
VCC3IOD	55.6
GNDKC	26.1
GNDKD	46.1
GNDIOC	26.1
GNDIOD	46.1

Figure 31 Core power in doc

Example: T90	U18
P=100mw V=1v $P = IV$ $I=\frac{P}{V}$ =100 mA T90: one IO port provides 52.394mw 100 /52.394 = 1… => 2 It needs 2 core power cells	P=78.725mw V=1.8 $I = \frac{78.725}{1.8} = 43.736$ mA U18: IO provides 34.1 mA 43.736/34.1 =1.28 => 2 The design needs 2 core power cells.

IO pad power:

The number of IO pads is decided by the SSO (Simultaneous Switching Output).

4.1 SSO Noise Effects

The important SSO noise effects include the SSO push out and false toggle.

SSO Push Out

Figure 4-1 shows the extent of the propagation delay caused by the SSO noise during simultaneously switching of several drivers, which is an important concern in the synchronous circuits. The maximum delay is dependent on the maximum allowable simultaneously switched output buffers. Different numbers of simultaneously switched output buffers have different delay variations.

Figure 32 SSO (1)

4.2 Recommended I/O to Power/Ground Ratio

SPICE simulation is used to simulate the simultaneously switched drivers. Table 4-1 and Table 4- show the recommended value of I/O to P/G ratio by considering only the SSO noise buffer effect (SSO push out and false toggle). The SSO noise source is the only consideration, other noise sources such as the crosstalk, coupling, reflection, etc., are not considered in these tables.

Figure 33 SSO (2)

Each output port has its own drive Strength, which determines its output frequency. If the clock frequency in the chip is more than 100M (10ns), the required drive strength is around 8mA. *(the accurate value is not certain in U18, anyway, pick a much large drive strength can guarantee the output drive)*. The greater the drive strength, the higher the output frequency can be reached.

Table 4-2. Recommended I/O to P/G ratio vs. L_p (Loose Criteria) (For 3.3 V Generic I/O Cells)

Parasitic Inductance (L_p)		Drive Strength			
		4 mA	8 mA	12 mA	16 mA
Fast Slew	2 nH	> 25	> 25	24	20
	4 nH	> 25	17	13	10
	8 nH	21	9	6	5
	10 nH	17	7	5	4
	12 nH	14	6	4	3
Slow Slew	2 nH	> 25	> 25	> 25	> 25
	4 nH	> 25	20	16	13
	8 nH	23	10	8	7
	10 nH	18	8	6	5
	12 nH	16	7	5	4

Figure 34 I/O to P/G ratio

$$9:1 = IO\ pad : IO\ Power$$

The 9 means 1 IO power can provide for 9 IO pads full current when 9 IO pads consume power simultaneously at 100M.

In the circuit, only the output ports consume power. For example, in TCBNN, the are 40 output ports.

$$\frac{40}{9} = 4.44 \cong 5$$

TCBNN needs 5 IO power pads to provide current.

Drive Strength of output port

Group Name : YA2GS

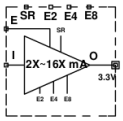
Function : Programmable Output Buffer, 3.3V

Cell List :

Pad Limited	Core Limited
YA2GSC	YA2GSD

Note: (1) Support Timing Model Only

Symbol



Truth Table

I	E	O
0	1	0
1	1	1
X	0	Z

Programmable Features

Output Driving Capability

E8	E4	E2	Driving @ 3.3V
0	0	0	2mA
0	0	1	4mA
0	1	0	6mA
0	1	1	8mA
1	0	0	10mA
1	0	1	12mA
1	1	0	14mA
1	1	1	16mA

Output Slew Rate

SR	Slew Rate
0	Fsat
1	Slow

Figure 35 Output pad info (1)

Input Capacitance (pf) & Maximum Loading (pf) & Power Consumption (uW/MHz)

Version	Input Capacitance						Maximum Loading	Power Consumption
	I	E	E2	E4	E8	SR		O
YA2GSC	0.0222	0.0246	0.0180	0.0223	0.0166	0.0183		20.99
YA2GSD	0.0228	0.0241	0.0149	0.0147	0.0140	0.0198		20.59

Figure 36 Output pad info (2)

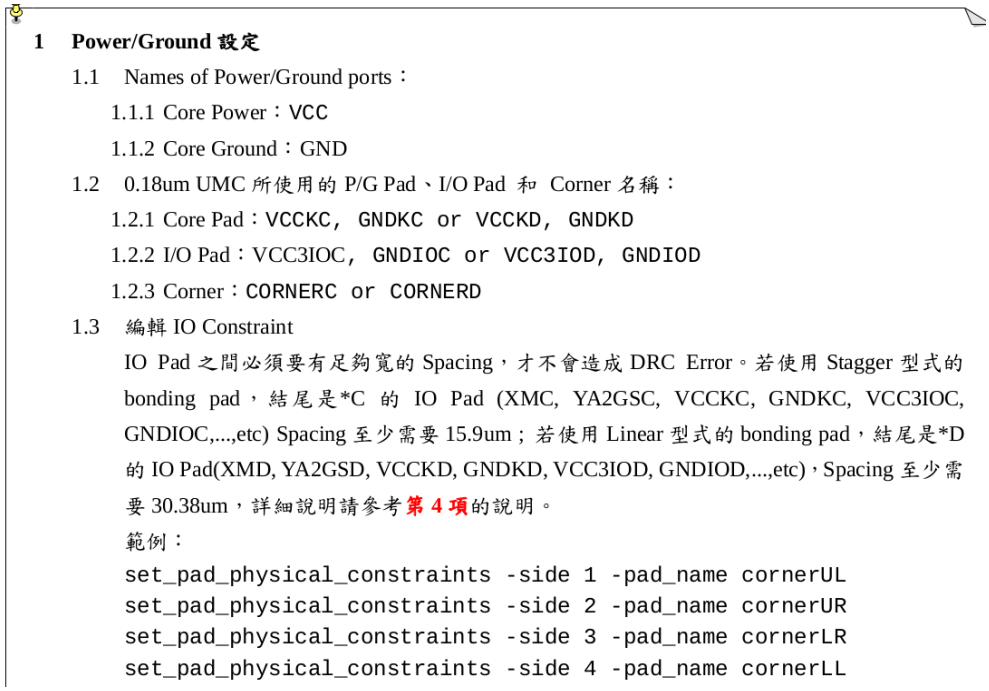
Table 5 Pad info

PAD limit	
YA2GSC opad_zero0(.SR(n_Logic0_), .O(A[0]), .I(i_A[0]), .E(n_Logic1_), .E2(n_Logic1_), .E4(n_Logic1_), .E8(n_Logic1_));	
.SR = 0	Set slew rate to fast
.O	Output port

.I	Input port
.E	High z or not
E2,E4,E8 = 1,1,1	Set driving factor to highest 16

IO.tdf (Top Design File)
Readme location:

/usr/cad/Library/CBDK018_UMC_Faraday_v1.0/CIC/ICC/DTS-151016-00-000.pdf



1 Power/Ground 設定

1.1 Names of Power/Ground ports：

1.1.1 Core Power：VCC

1.1.2 Core Ground：GND

1.2 0.18um UMC 所使用的 P/G Pad、I/O Pad 和 Corner 名稱：

1.2.1 Core Pad：VCCKC, GNDKC or VCCKD, GNDKD

1.2.2 I/O Pad：VCC3IOC, GNDIOC or VCC3IOD, GNDIOD

1.2.3 Corner：CORNERC or CORNERD

1.3 編輯 IO Constraint

IO Pad 之間必須要有足夠寬的 Spacing，才不會造成 DRC Error。若使用 Stagger 型式的 bonding pad，結尾是*C 的 IO Pad (XMC, YA2GSC, VCCKC, GNDKC, VCC3IOC, GNDIOC,...,etc) Spacing 至少需要 15.9um；若使用 Linear 型式的 bonding pad，結尾是*D 的 IO Pad(XMD, YA2GSD, VCCKD, GNDKD, VCC3IOD, GNDIOD,...,etc)，Spacing 至少需要 30.38um，詳細說明請參考**第 4 項**的說明。

範例：

```
set_pad_physical_constraints -side 1 -pad_name cornerUL
set_pad_physical_constraints -side 2 -pad_name cornerUR
set_pad_physical_constraints -side 3 -pad_name cornerLR
set_pad_physical_constraints -side 4 -pad_name cornerLL
```

Figure 37 IO.tdf setting

Minimum IO space: U18:16 um

Design setup

The logical library that provides timing and functionality for all standard cells and hard macros must be specified in ICC. To layout the standard cells without violating Design Rule, a physical library that contains information about the design rules for each layer must be specified as well.

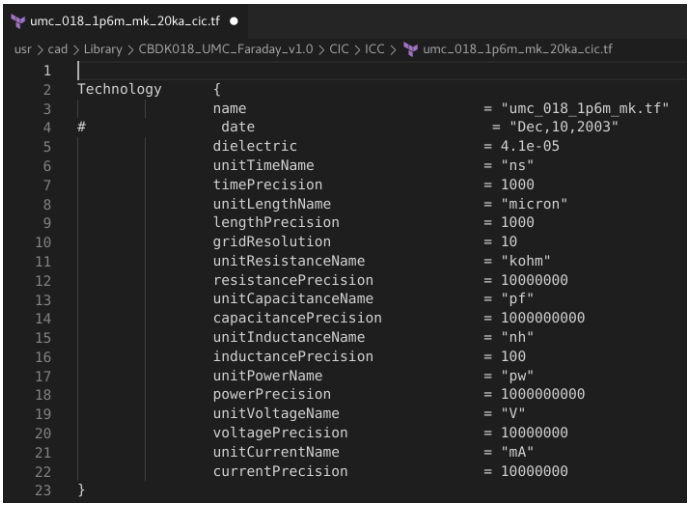


Figure 38 U18 Physical Library

In the design setup phase, ICC will put all the standard cell on the screen and it can be seen on the cell view. Figure 39

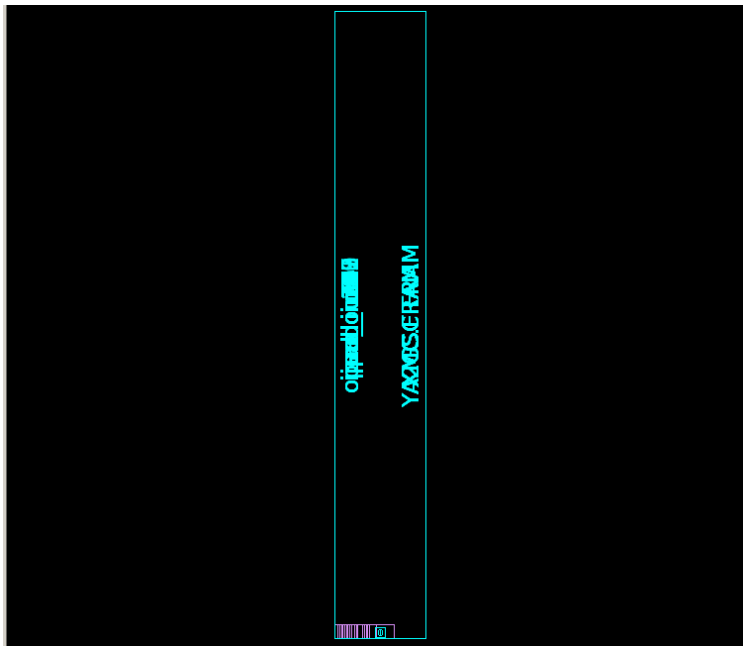


Figure 39 Design Setup Cell view

ICC calculates interconnect capacitance and resistances using geometry and the TLU+ look-up table. The parasitic effects including parasitic capacitances, parasitic

resistances and parasitic inductances are in the TLUPlus file which enables accurate RC extraction results in ICC.

Design Planning

In the design planning phase, the course floorplan must be initialized. The top-level IO cells are listed according to the order defined in CHIP.v. The standard cells are puts on the right-hand side in Figure 40.

To avoid the congestion problem and DRC violation in routing phase, the core utilization should be defined appropriately. The experimental results show that the acceptable core utilization of TCBNN is 0.1 which means 10% of core area is occupied by standard cells, 90% of core area is saved for the routing.

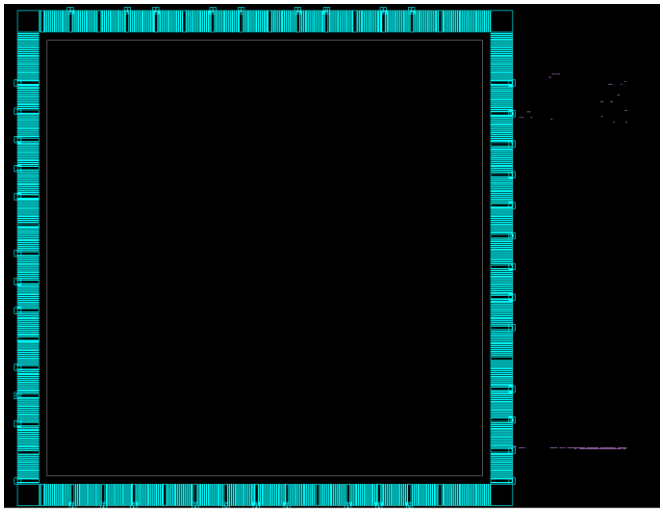


Figure 40 Design Planning-create floorplan

CHIP.v

users > m1053011 > Desktop > tcbnn > verilog > CHIP.v > {} CHIP

```
module CHIP(
input s_axis_last ,
output m_axis_valid ,
input s_axis_valid ,
input axi_clk ,
output m_axis_last ,
input axi_reset_n ,
output s_axis_ready ,
input [31:0] s_axis_data ,
output [31:0] m_axis_data ,
input m_axis_ready
);
wire i_s_axis_last;
wire i_m_axis_valid;
wire i_s_axis_valid;
wire i_axi_clk;
wire i_m_axis_last;
wire i_axi_reset_n;
wire i_s_axis_ready;
wire [31:0] i_s_axis_data;
wire [31:0] i_m_axis_data;
wire i_m_axis_ready;
axis_top_ver1126 DUT (.s_axis_last(i_s_axis_last),.m_axis_valid(i_m_axis_valid),.s_axis_valid(i_s_ax

XMC ipad_in0(.O(i_s_axis_last), .I(s_axis_last),.PU(n_Logic0_), .PD(n_Logic0_), .SMT(n_Logic0_));

XMC ipad_in1(.O(i_s_axis_valid), .I(s_axis_valid),.PU(n_Logic0_), .PD(n_Logic0_), .SMT(n_Logic0_));
XMC ipad_in2(.O(i_axi_clk), .I(axi_clk),.PU(n_Logic0_), .PD(n_Logic0_), .SMT(n_Logic0_));

XMC ipad_in3(.O(i_axi_reset_n), .I(axi_reset_n),.PU(n_Logic0_), .PD(n_Logic0_), .SMT(n_Logic0_));
```

Figure 41 CHIP.v contains the physical cells including IO pads, corner pad.

After create_fp_placement, the standard cells which are synthesized by Design Compiler in CHIP_syn.v are roughly placed on the chip.

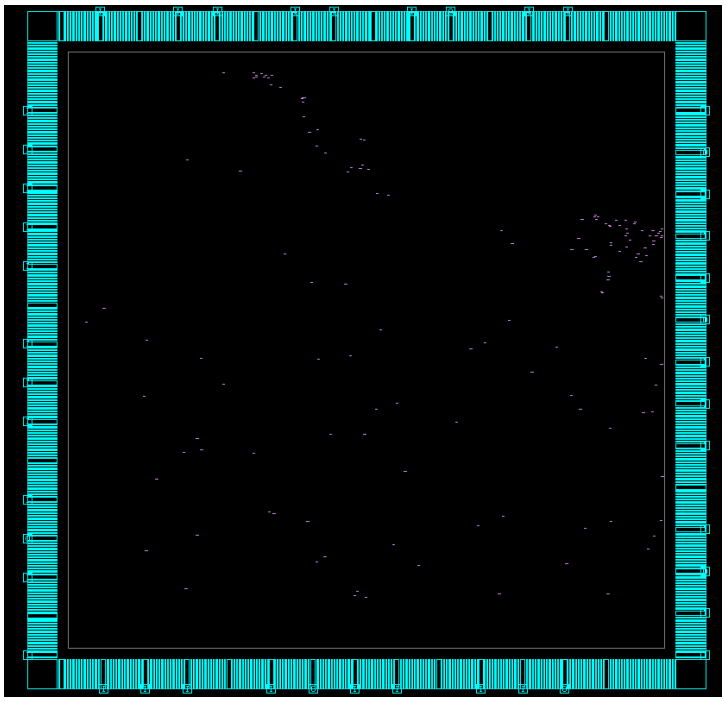


Figure 42 Design Planning-create_fp_placement

In the design planning phase, the power network is defined by the power report from Design Compiler.

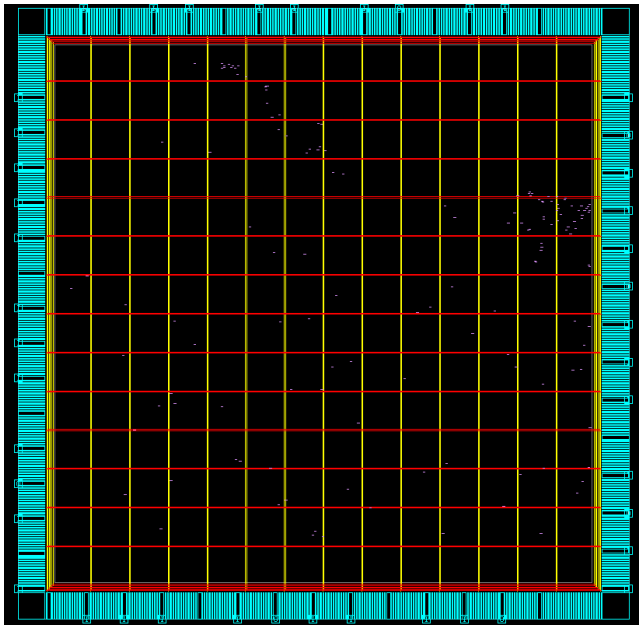


Figure 43 Design Planning-Power Network Synthesis

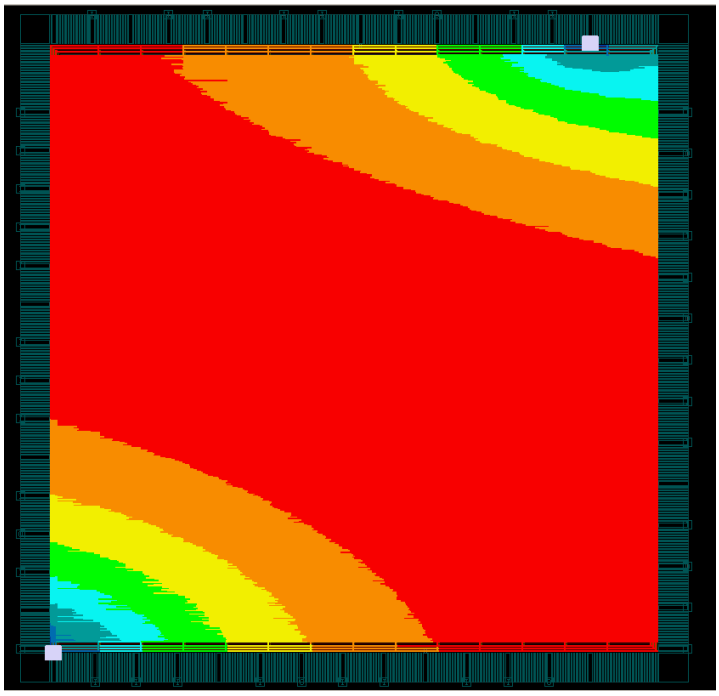
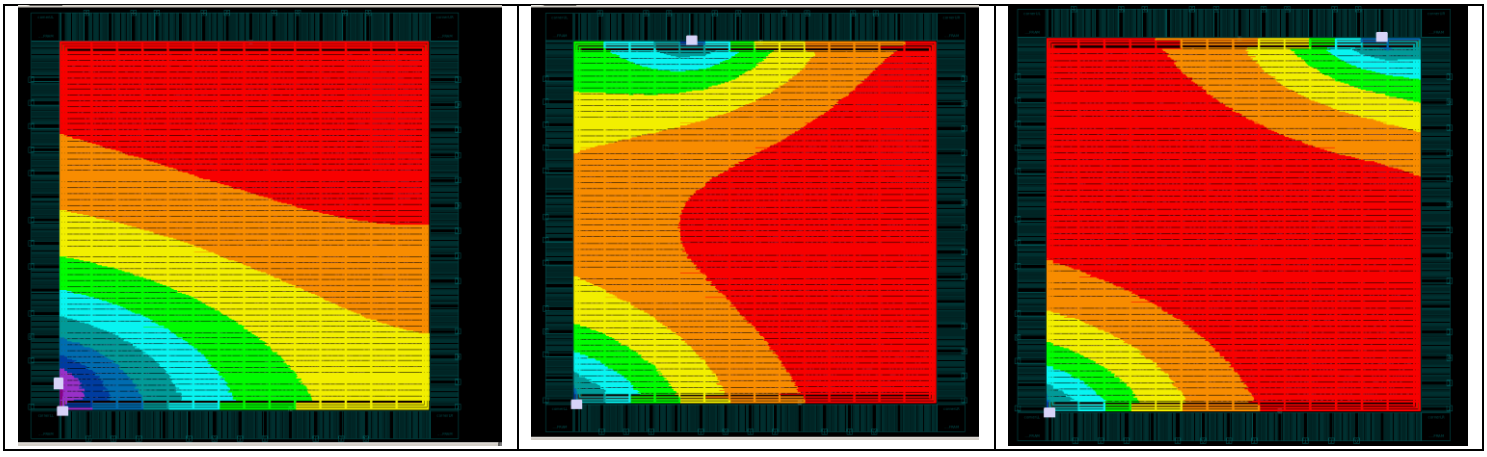


Figure 44 Design Planning-PNS Analysis

The power network must be pre-routed to analyze the voltage drops between IO ports and standard cells. According to [TSRI], the max voltage drop cannot exceed 10% of the working voltage. In this thesis, the working voltage of U18 is 1.8V. Namely, the max voltage drop cannot be larger than 0.18V. In this design, the max IR drops is around 45mV.



Try to move the power pad and the power analysis will be different.

```
INFO: 52 pre-routes used for checking; 0 redundant shapes removed
[end initializing data for legality checker]
Error: report_congestion failed because some cells do not have legal placement. (PSYN-972)
0
```

If PYSN-972 happened:

legalize_placement

check_zrt_routability report following message:

```
>>> Port at [3603.060,2811.790..3607.360,2842.070] metal1, metal2, metal3, metal4, metal5, metal6 is blocked. Inst=o
>>> Port at [3603.060,2811.790..3607.360,2842.070] metal6 is blocked. Inst=  Master=CHIP Port=m_axis_data[1] Net=m_a
>>> Port at [2829.330,3600.340..2859.610,3604.640] metal1, metal2, metal3, metal4, metal5, metal6 is blocked. Inst=i
>>> Port at [2829.330,3600.340..2859.610,3604.640] metal6 is blocked. Inst=  Master=CHIP Port=s_axis_data[20] Net=s_

>>>>>> The design cannot be cleanly routed because it has 86 completely blocked ports
```

It can be ignored.

Placement

In placement phase, the ICC will spread apart cells that contribute to high congestion to prevent routing problem. For most designs, it is better to deal with congestion problems before fixing timing issues. Since timing issues can be fixed by adding more buffers to resolve timing delay, congestion problems are often restricted by the space and lead to impossible to complete routing.

ICC will automatically buffer all high-fanout nets except clock and constant nets. Clock optimization will be done in the Clock Tree Synthesis (CTS) phase.

<table>
<tr><td rowspan="4">Setup</td><td>Start with a Good Design Setup</td><td>check_physical_constraints</td></tr>
<tr><td>Placement Setup</td><td>Define placement blockages</td></tr>
<tr><td>DFT Setup</td><td>Skip</td></tr>
<tr><td>Power Setup</td><td>Perform leakage and dynamic optimization:<table><tr><td>clock gating</td><td>Design clock gating</td></tr><tr><td>leakage power optimization</td><td>High/Low V_{th} cells only available at T90/T40/T29 process</td></tr><tr><td>dynamic power optimization</td><td>read_saif…</td></tr></table></td></tr>
<tr><td rowspan="2">Execution</td><td>Tie Connection</td><td>add_tie.tcl</td></tr>
<tr><td>Placement & Optimization</td><td>place_opt
place_opt -power</td></tr>
<tr><td>Analysis</td><td>Improve Congestion/ Timing</td><td>Analysis congestion maps and report</td></tr>
</table>

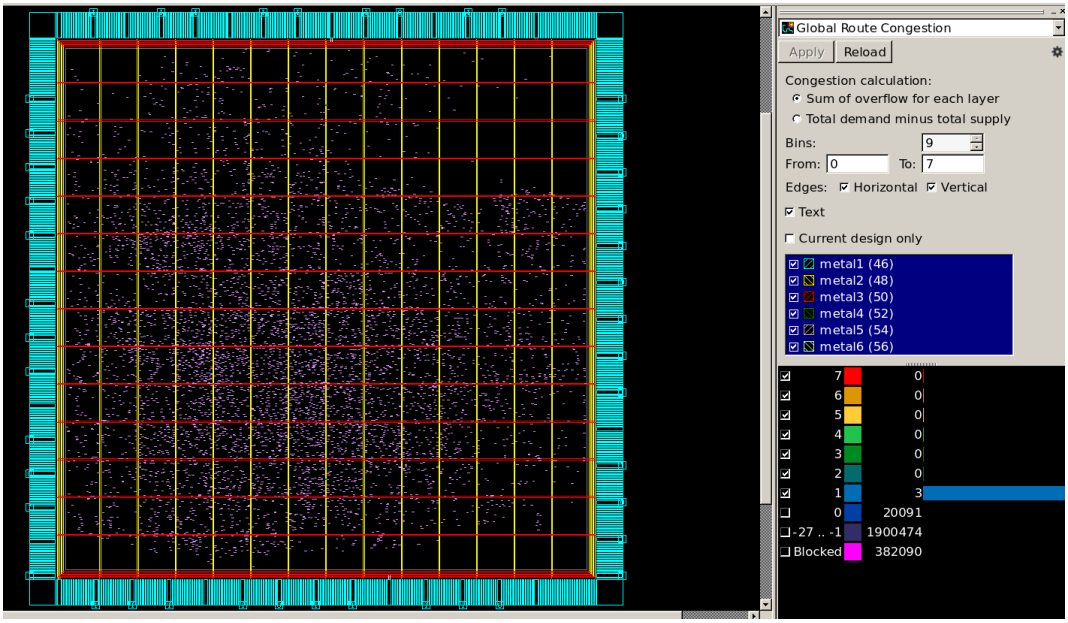


Figure 45 Placement-Global Route Congestion

CTS

For the reason that clock signal is one of the most important signals in digital design, clock tree will be synthesized before all other signals. In CTS phase, ICC will create a buffer tree to minimize skew, latency and transition goals. Because CTS may move non-clock cells to less ideal location which causes congestion or transition and capacitance violations, it is essential to check design constraints after CTS.

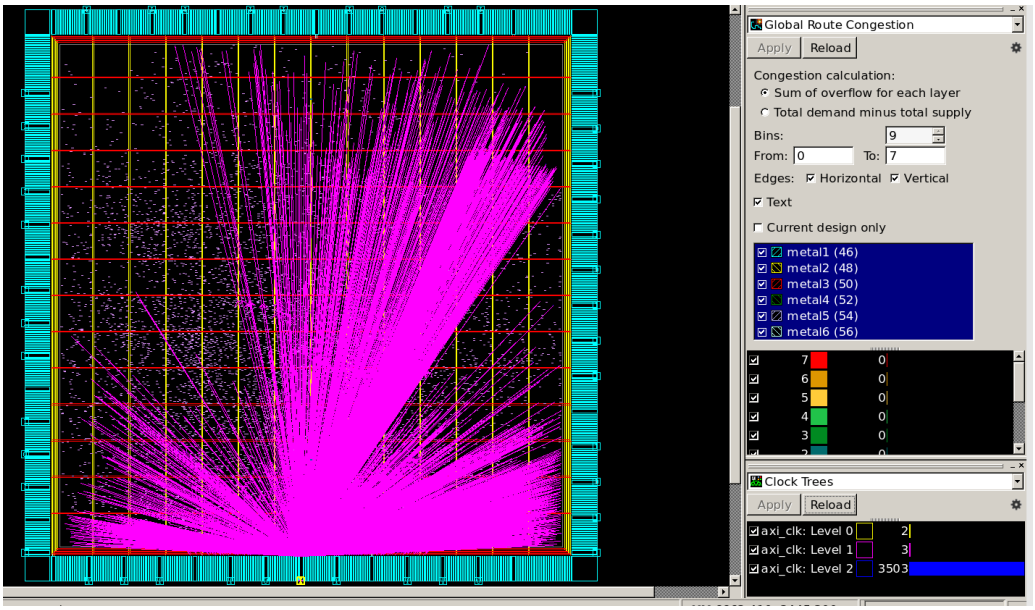


Figure 46 CTS-Before CTS

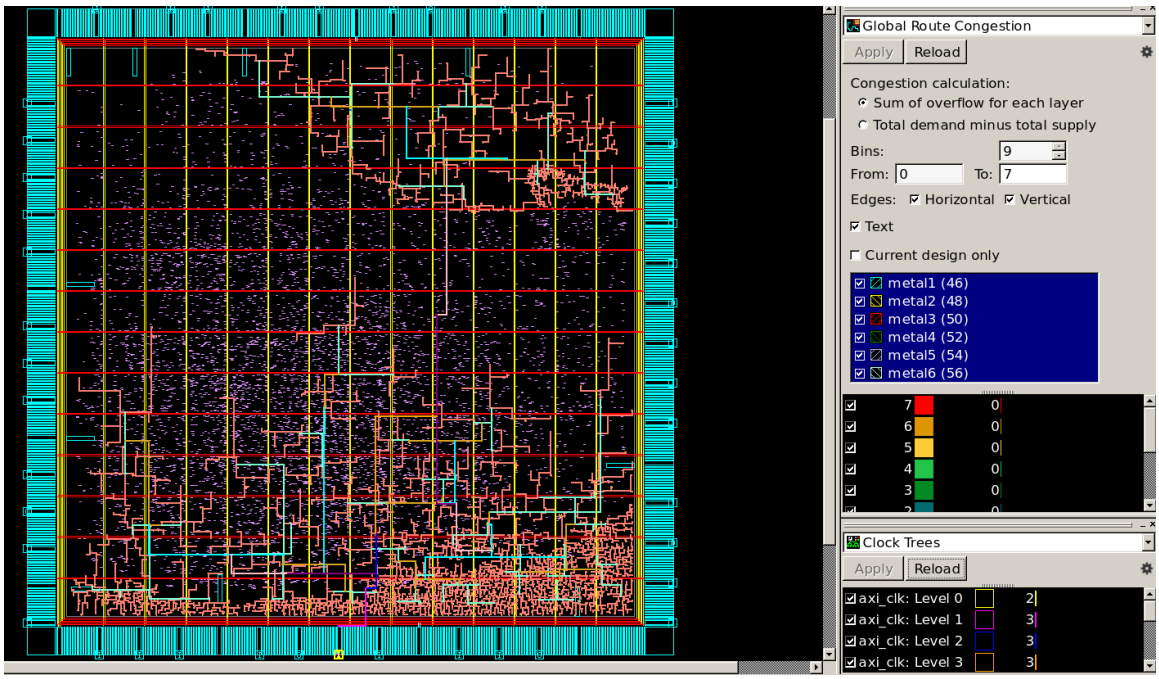


Figure 47 CTS-After CTS

```
****************************************
Report : clock tree
Design : CHIP
Version: T-2022.03
Date   : Sun Apr  2 14:55:38 2023
****************************************

Information: Float pin scale factor for the 'max' operating condition of scenario 'default' is set to 1.000 (CTS-375
)

======================= Clock Tree Summary ========================
Clock                Sinks     CTBuffers ClkCells  Skew      LongestPath TotalDRC   BufferArea
-------------------------------------------------------------------------------------------------
axi_clk              3503      131       132       0.1003    2.5839      0            4546.5830
1
icc_shell>
```

Figure 48CTS-Clock Tree Summary

Route

Placement & CTS	
Routing Set	set_route_zrt_common_options \ -post_detail_route_redundant_via_insertion high \ -concurrent_redundant_via_mode insert_at_high_cost \ -concurrent_redundant_via_effort_level high
Route Clock net	route_zrt_group -all_clock_nets ;
Route Signal Nets	route_zrt_auto
Optimize	verify_zrt_route route_zrt_detail -incremental true \ -initial_drc_from_input true
Design for Manufacturing	

Figure 49 General Flow for Routing

Routing creates physical connection to all clock and signal pins through metal interconnects. All routes' paths must meet setup and hold timing and other design constraints.

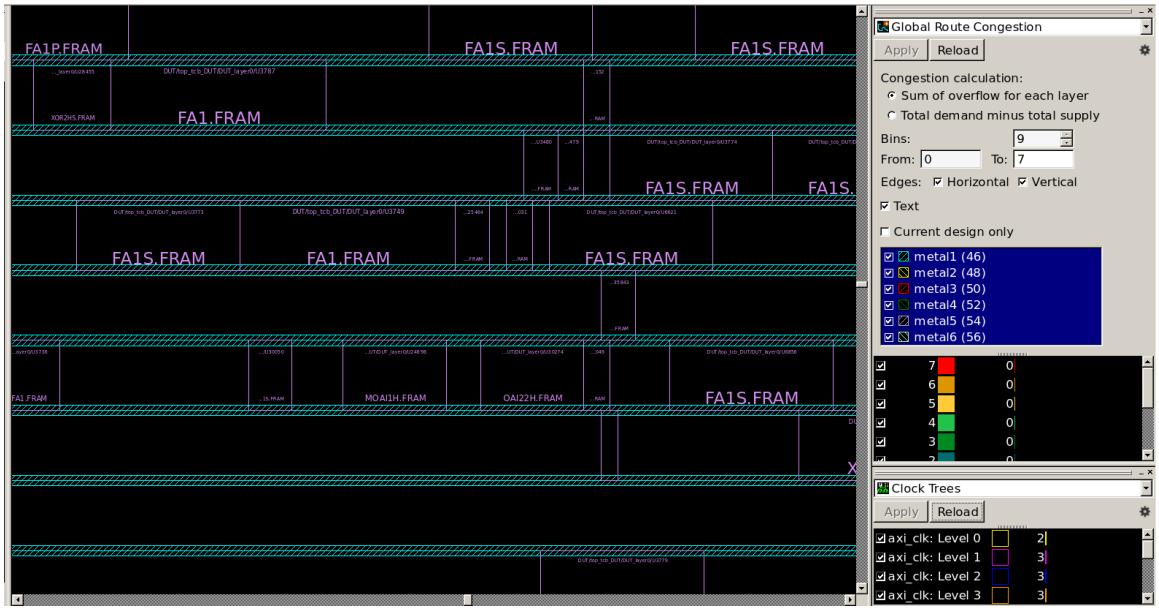


Figure 50 Route-Standard cell Before route

Metal traces (routes) are built along and centered upon routing tracks based on a grid. Each metal layer has its own grid and preferred routing direction. ICC performs global routing, tack routing and detail routing.

Global routing assign nets to specific metal layers and global routing cells (Gcells). Global routing tries to avoid congested Gcells including power/ground net, blockage while minimizing detours.

Track assignment assigns each net to a specific track and lays down the actual metal traces. When ICC performs track assignment, it attempts to make long, straight trace and reduce the number of via. Track assignment does not check or follow physical DRC rules (min-spacing).

Detail route try to clear DRC violations left by track assignment stage using a fixed size Sbox.

After global routing, track assignment and detail routing, all clock signal nets will be completely routed and should meet all timing and all DRC requirements.

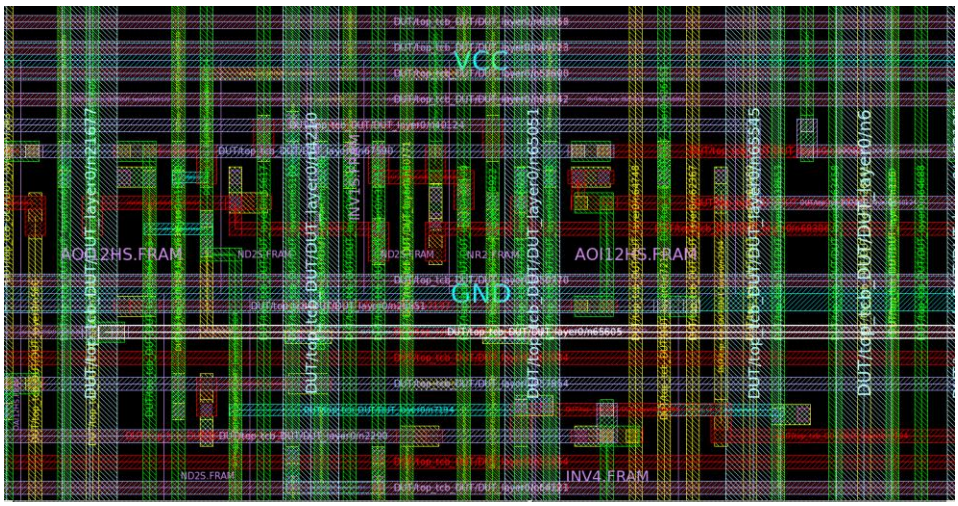
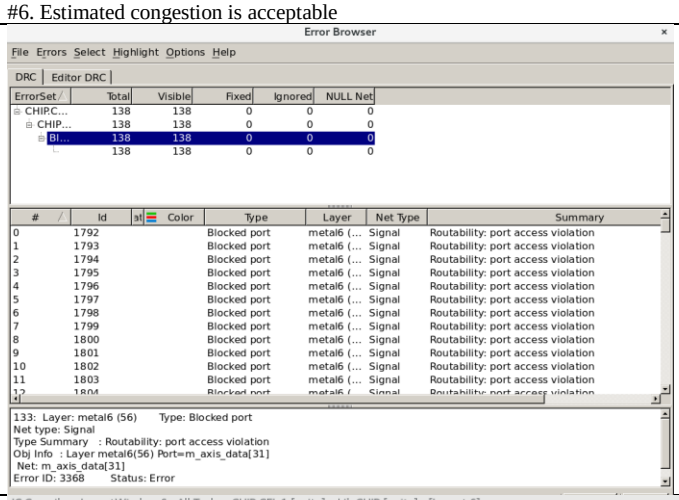
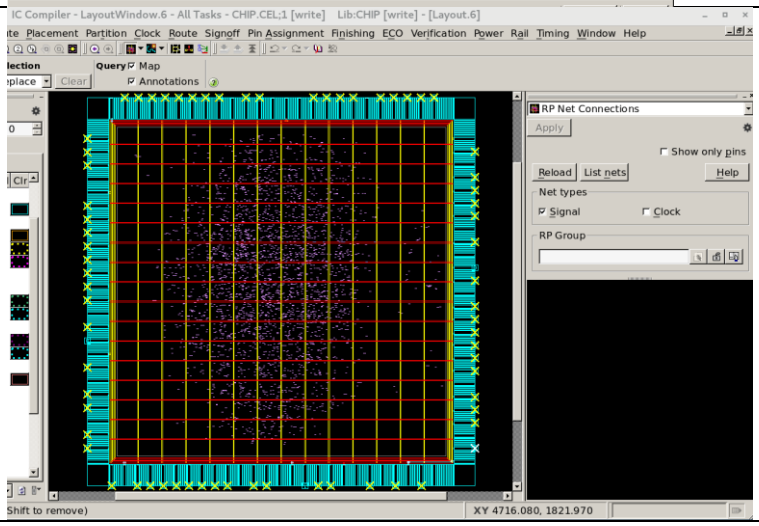
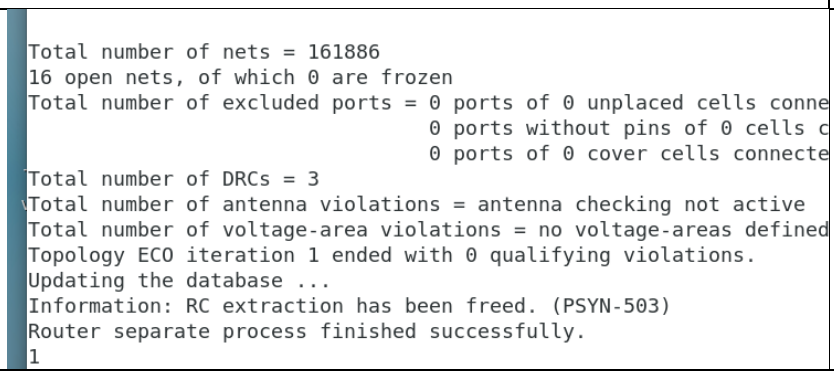


Figure 51 Route-Standard cell After route

```
Verify Summary:

Total number of nets = 93176, 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

Router separate process finished successfully.
1
```

Figure 52 Route-Verify zroute result

report_constraint -all	#Before Route, the following requirements must be met: #1. P/G net prerouted #2. placement is completed #3. CTS is completed #4. Estimated timing is acceptable ~0 ~ -0.01 #5. Estimated max cap/transition no violation #6. Estimated congestion is acceptable	
Check_routability		
set_route_zrt_common_options -post_detail_route_redundant_via_insertion high -concurrent_redundant_via_mode insert_at_high_cost -concurrent_redundant_via_effort_level high route_zrt_group -all_clock_nets		
verify_zrt_route route_zrt_detail -incremental true -initial_drc_from_input true -max_number_iterations 1000 ;# fix DRC problem		

DRC and open net must be 0.	Total number of nets = 161886 16 open nets, of which 0 are frozen Total number of excluded ports = 0 ports of 0 unplaced cells 0 ports without pins of 0 ce 0 ports of 0 cover cells cor Total number of DRCs = 1 Total number of antenna violations = antenna checking not act Total number of voltage-area violations = no voltage-areas de Topology ECO iteration 1 ended with 0 qualifying violations. Updating the database ... Information: RC extraction has been freed. (PSYN-503) Router separate process finished successfully. 1 icc_shell> Warning: GRC mapmode data is updated. (GUI-998)
OK	Verify Summary: Total number of nets = 84492, 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 0 ports without pins of 0 ce 0 ports of 0 cover cells cor Total number of DRCs = 0 Total number of antenna violations = no antenna rules defined Total number of voltage-area violations = no voltage-areas de Total number of tie to rail violations = not checked Total number of tie to rail directly violations = not checked 1
OK	Total number of nets = 98782 0 open nets, of which 0 are frozen Total number of excluded ports = 0 ports of 0 unplaced cells ce 0 ports without pins of 0 cell 0 ports of 0 cover cells conne Total number of DRCs = 0 Total number of antenna violations = antenna checking not activ Total number of voltage-area violations = no voltage-areas defi Topology ECO iteration 1 ended with 0 qualifying violations. Updating the database ... Information: RC extraction has been freed. (PSYN-503) 1 icc_shell>

DRC Problem

If the worst case happened, the DRC problems cannot be resolved by verify_zrt_route and route_zrt_detail. Here are three ways to deal with DRC problems.

```
#=========================================
# [1]
# If still have DRC problems, repeat the following command
#=========================================
# verify_zrt_route > [format %s%s%s $REPORT_PATH $design_name "_route_drc.rpt"];
# route_zrt_detail -incremental true -initial_drc_from_input true  -max_number_iterations 1000 ;# fix DRC problem

#=========================================
# [2]
# If DRC violation can not be solved after using route_zrt_detail
# Remove the detail route and re-route detail route
# route_opt will do GR,DR,search%repai,logic and placenebt opt
# =========================================

# remove_route_by_type -signal_detail_route ;#delete routing
# route_opt -effort high ; #re-route
# verify_zrt_route
# route_zrt_detail -incremental true -initial_drc_from_input true  -max_number_iterations 1000 ;# fix DRC problem

#=========================================
# [3]
# re-route and blockages
#=========================================
# create_routing_blockage -layers {metal2Blockage}  -bbox {{014.930 998.580} {1015.770 999.600}}


# remove_route_by_type -signal_detail_route -nets DUT/top_tcb_DUT/DUT_layer0/sub_1_root_sub_1_root_add_157_221_DIFF_13_

# remove_route_by_type -signal_detail_route -nets  DUT/top_tcb_DUT/DUT_layer0/sub_61_root_add_0_root_add_206/carry[13]

# route_eco -nets {DUT/top_tcb_DUT/DUT_layer0/sub_1_root_sub_1_root_add_157_221_DIFF_13_ \
# DUT/top_tcb_DUT/DUT_layer0/sub_61_root_add_0_root_add_206/carry[13]}

# RT-302
# route_eco -auto -search_repair_loop 5\
#          -utilize_dangling_wires -scope local\
#          -reroute modified_nets_first_then_others

# remove_routing_blockage {RB_11266049}
```

Figure 53 Route-DRC tcl

1. Repeat verify_zrt_route and route_zrt_detail.
   Set the max iteration to max 1000. And let ICC to fix DRC problem. Remember to use verify_zrt_route before route_zrt_detail.
2. Remove the detail route to re-route.
   Remove detail route by
   remove_route_by_type -signal_detail_route
   and if there are any blockages in the design, remove it.
   Run route opt and set the effort the high.
   route_opt -effort high
3. Re-route and set blockages.
   If the routing problem cannot be resolved by re-route and route_zrt_detail, you can only fix it by yourself by setting blockage on the layout. Open the error browser window seen in Figure 54 in ICC and choose detail route. Find out where the DRC problems are and set a blockage on the location. For example, there is min-spacing violation between two wires. Delete two wires and set a blockage on this location. Re-route detail route. After routing, remember to remove the blockage on the layout.

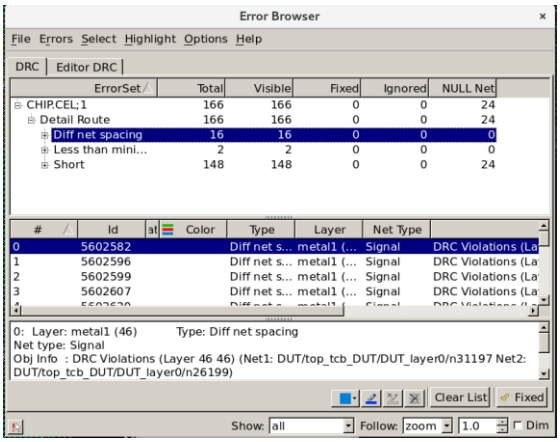


Figure 54 Route- Error Browser window

4. If the DRC problems cannot be solved by the above methods, go back to placement to deal with congestion or go back to the design planning and change the core utilization. Try to leave more space for routing.

Placement -effort high

Design Setup			
↓			
Design Planning	←	←	←
↓			↑
Placement	←	←	↑
↓		↑	↑
CTS		↑	↑
↓		↑	↑
Routing	→	↑	↑
↓			
Chip Finishing			

Figure 55 Repeat flow

Chip Finishing

Design For manufacture (DFM) performs key chip finishing and design for manufacturing steps required after the signal routing is complete.

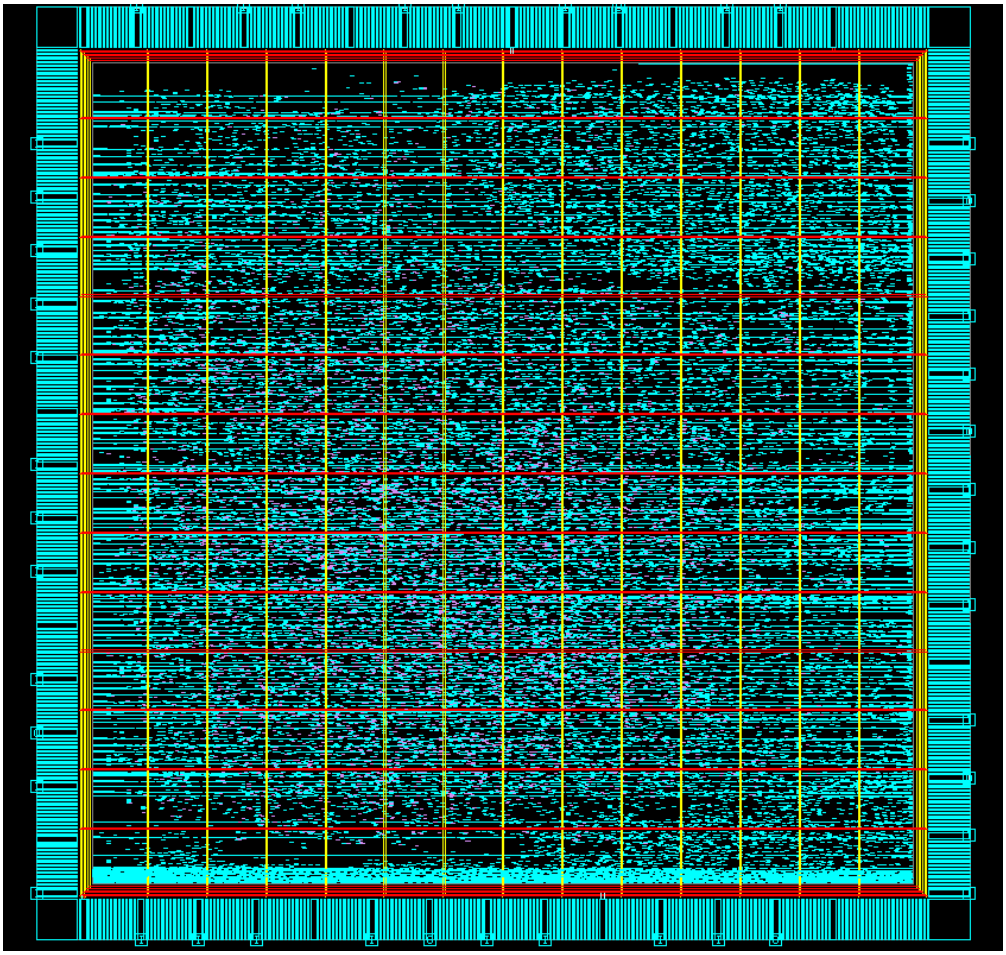


Figure 56 DFM-Add core filler

Fix antenna violations, add redundant vias, insert standard cell filler, insert metal filler, DRC, LVS check.

Add IO text
Choose m3text.

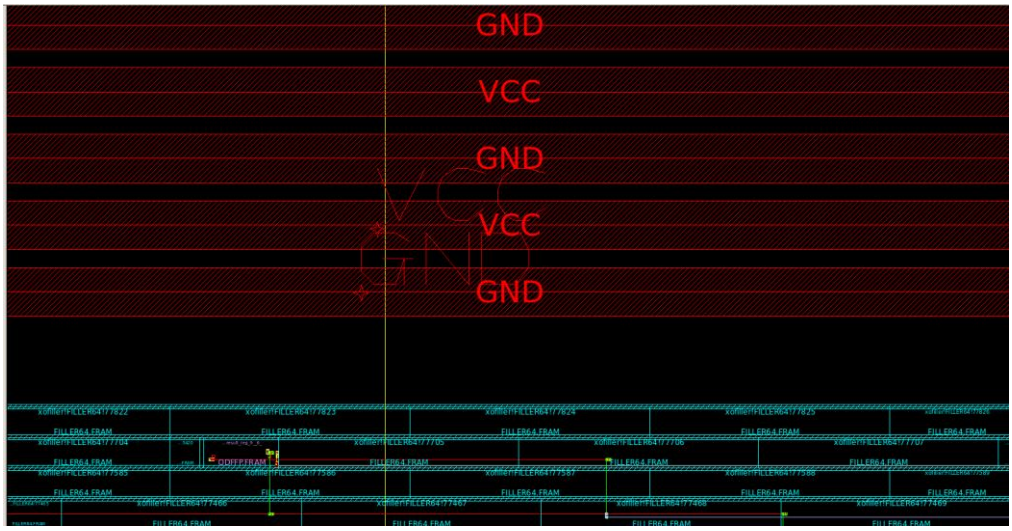


Figure 57 DFM-Add GND VCC on the power ring.

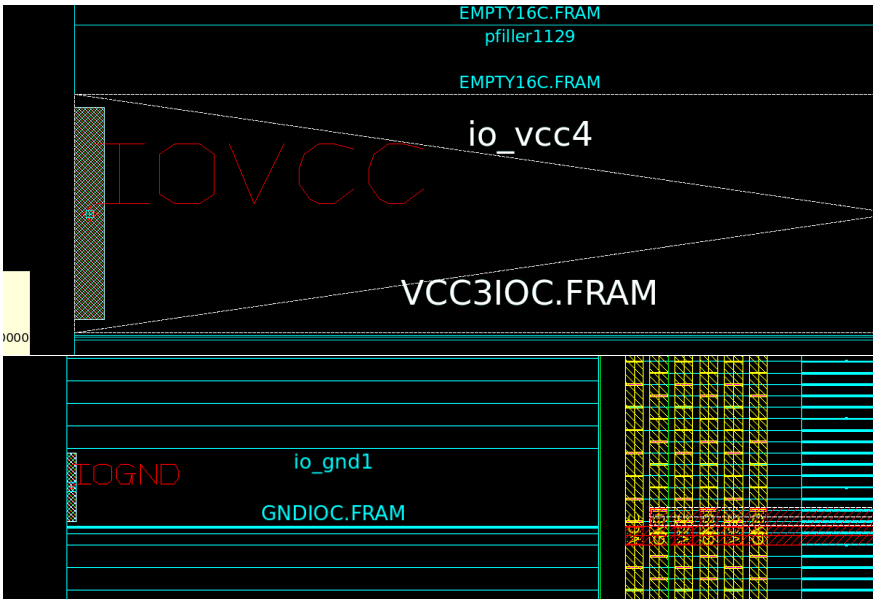


Figure 58 DFM-Add VCC/GND Pin

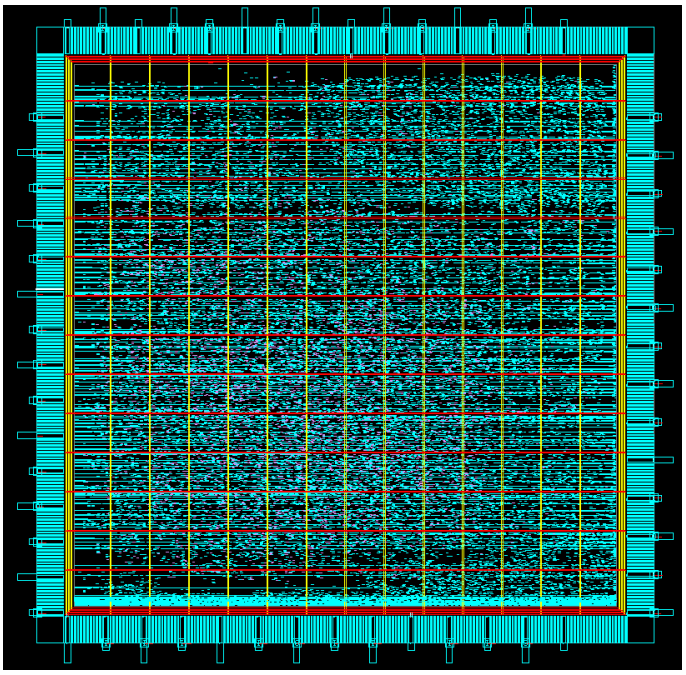


Figure 59 DFM-Add Bond pad

CHIP size

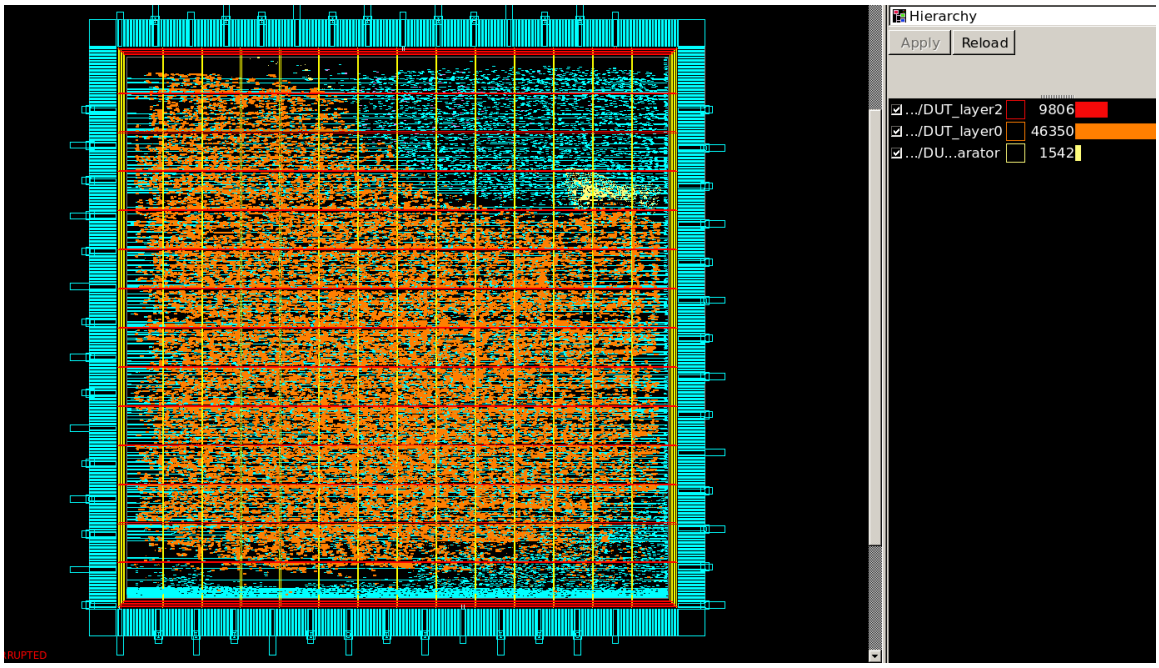


Figure 60 CHIP Hierarchy

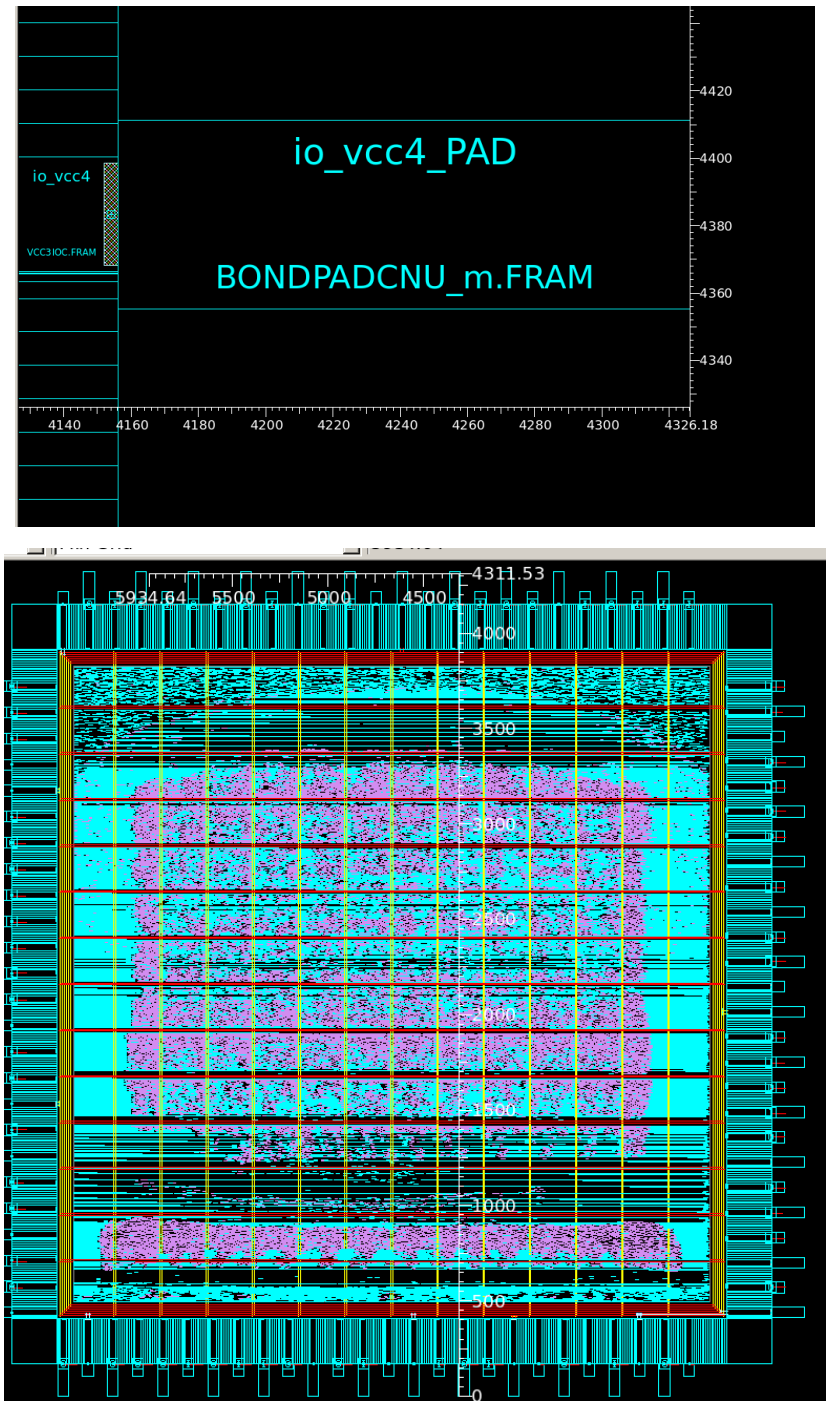


Figure 61 DFM-Chip size

4311.53*4326.18 um^2

Verify (DRC LVS)

After writing out the gds and netlist file, it is essential to run DRC and LVS.

DRC	calibre -drc -hier G-DF-Mixed_Mode_RFCMOS18-1.8v_3.3v-1P6M-MMC-Calibre-DRC-2.14_P1
	calibre -rve BaseRule.db
V2LVS	v2lvs -v CHIP_pr.v -l umc18_core_lvs.v -s umc18_core_lvs.spi -l umc18_io_lvs.v -sumc18_io_lvs.spi -lumc18_pad_lvs.v -s umc18_pad_lvs.spi -o source.spi -s1 VCC -s0 GND
LVS	calibre -lvs -spice layout.spi -hier -auto G-DF-MIXED_MODE_RFCMOS18-1.8V_3.3V-1P6M-MMC_CALIBRE-LVS-2.1-P8_m.txt
	calibre -rve svdb

Figure 62 Verify-command

```
504
505   SOURCE PATH      "./source.spi"
506   SOURCE PRIMARY   "CHIP"
507   SOURCE SYSTEM    SPICE
508
509   LAYOUT PATH      "./CHIP_lab.gds"
510   LAYOUT PRIMARY   "CHIP"
511   LAYOUT SYSTEM    GDSII
512
```

Figure 63 Verify-Edit LVS file

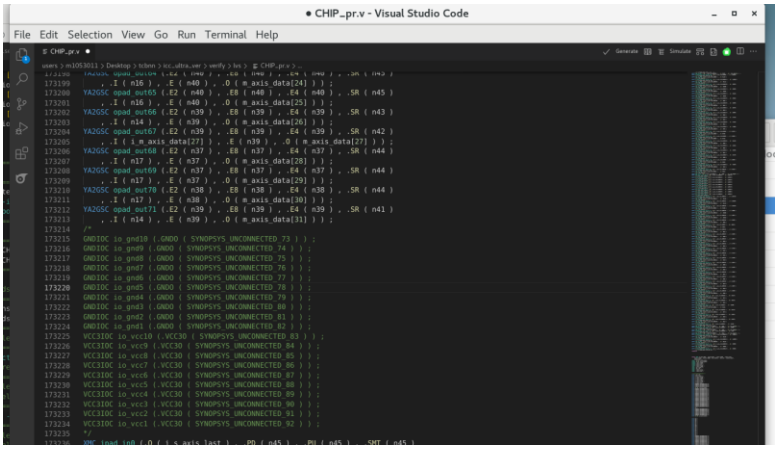


Figure 64 Verify-Comment the GNDIOC and VCC3IOC in CHIP_pr.v

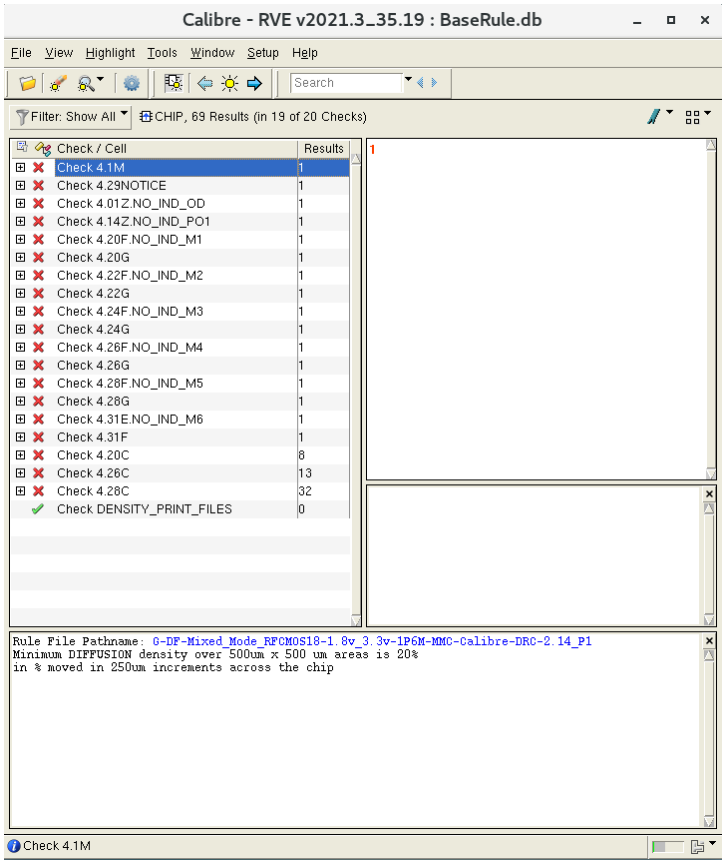


Figure 65 Verify-DRC Result

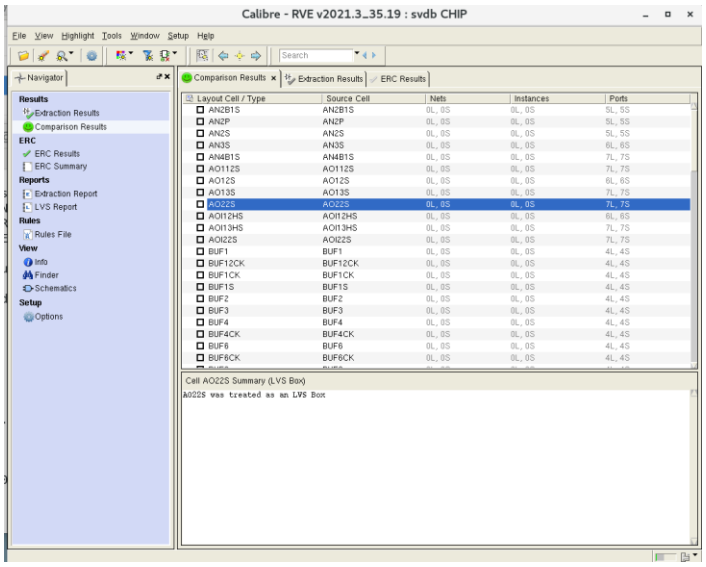


Figure 66 Verify-LVS Result (1)

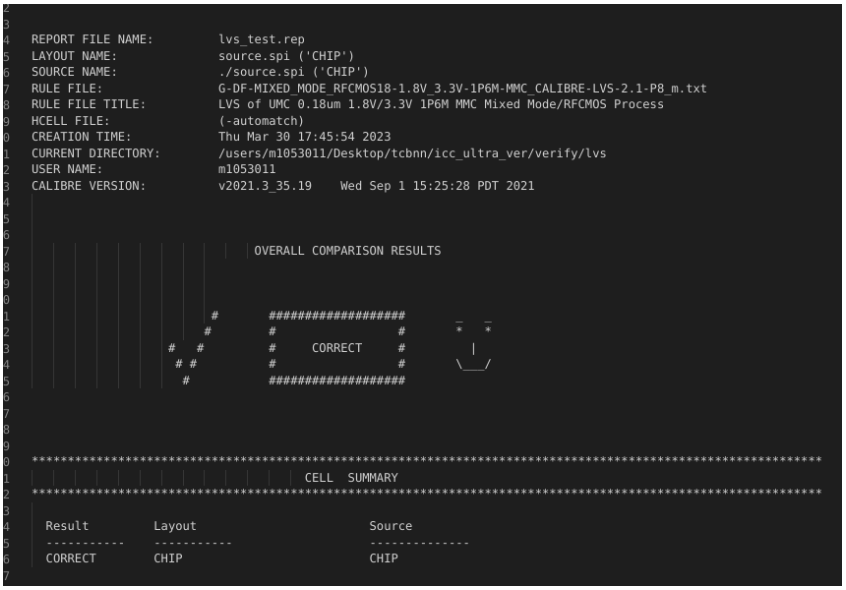
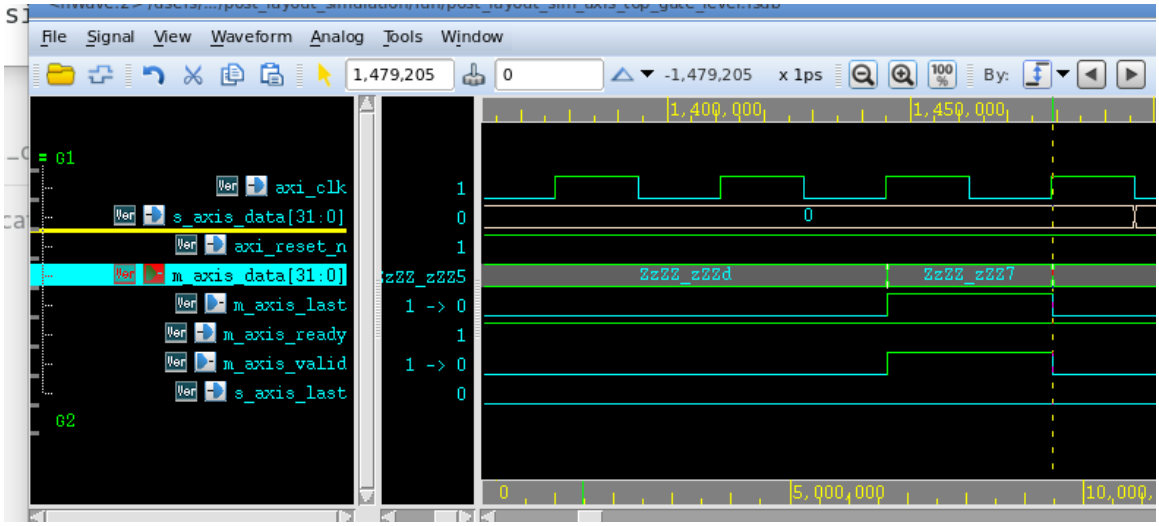
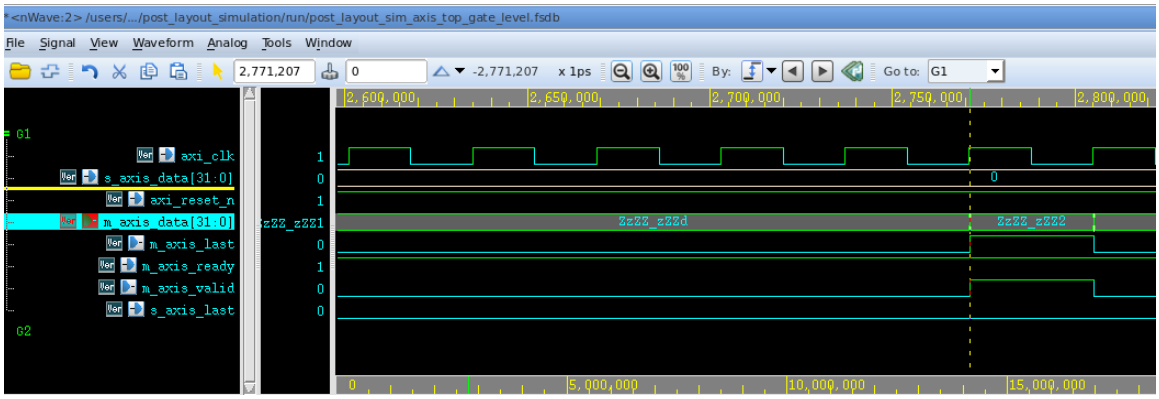
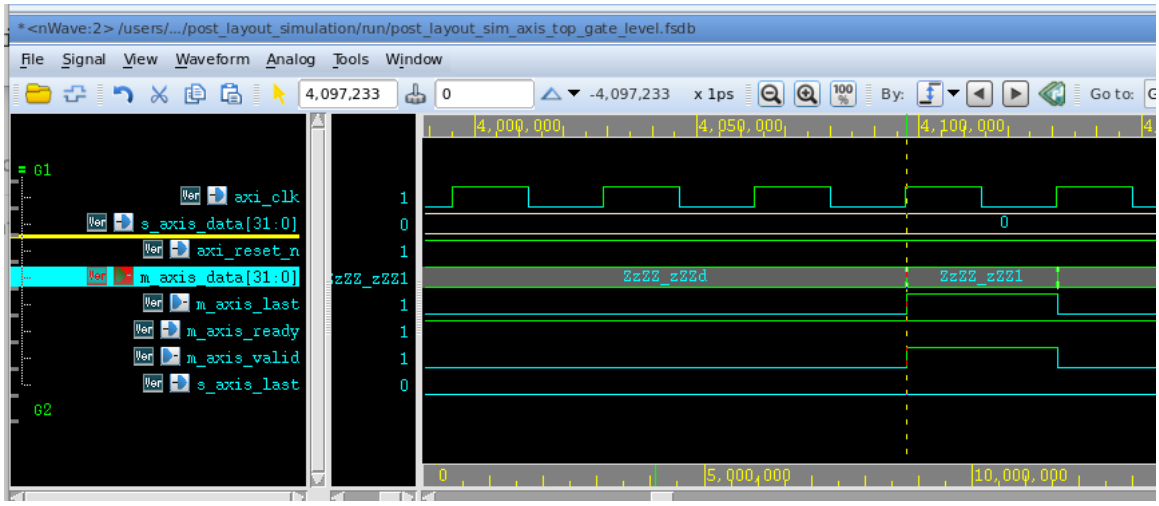


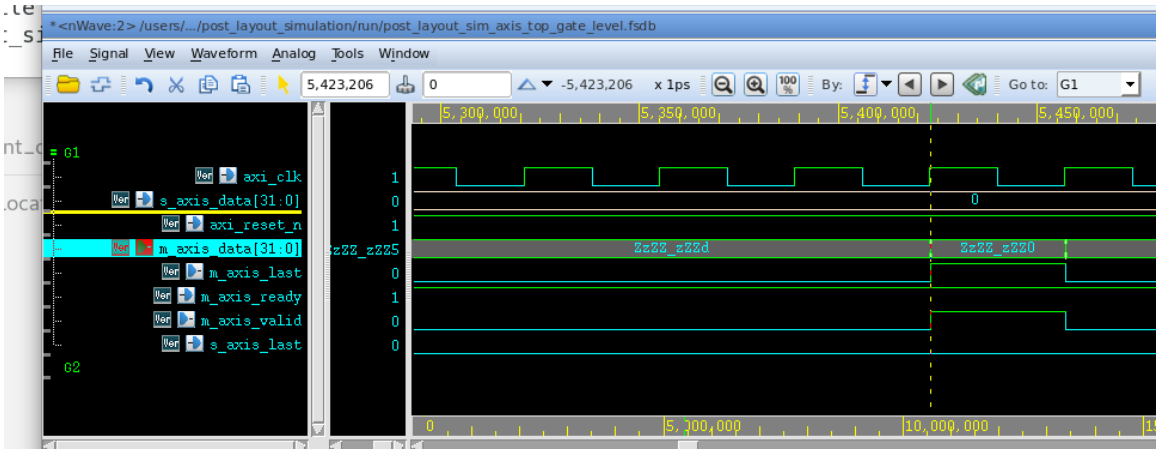
Figure 67 Verify-LVS Result (2)

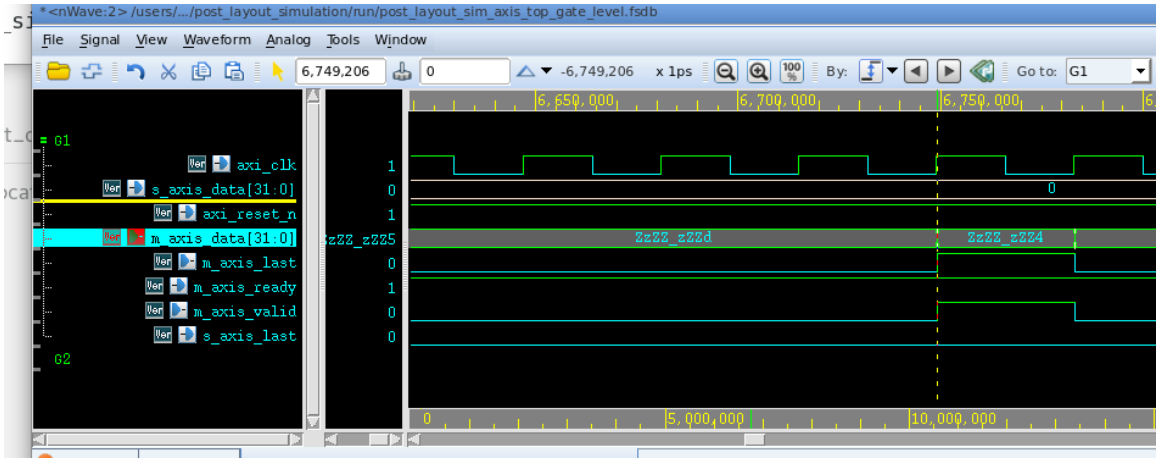
Post-layout simulation

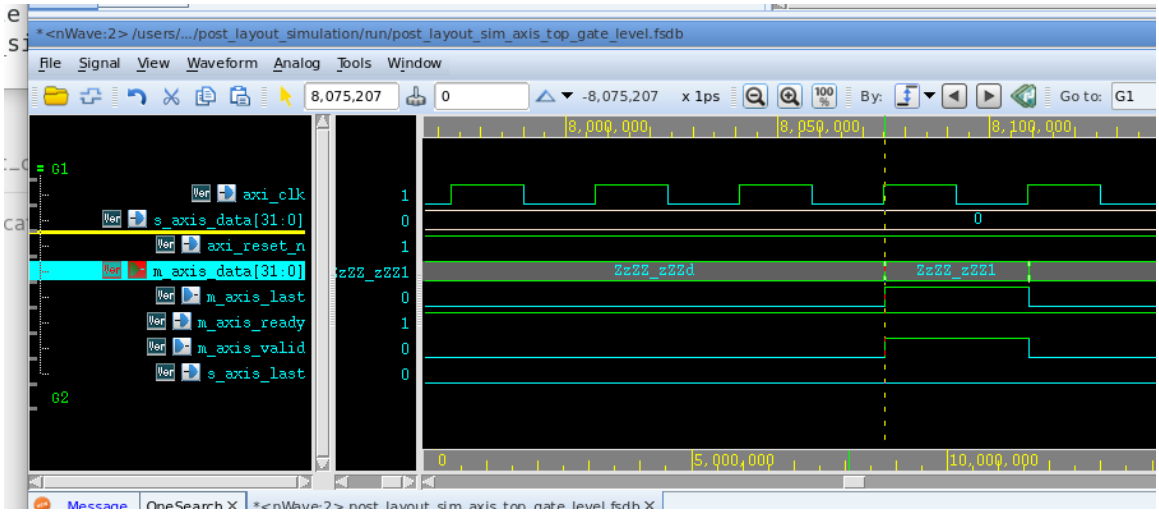


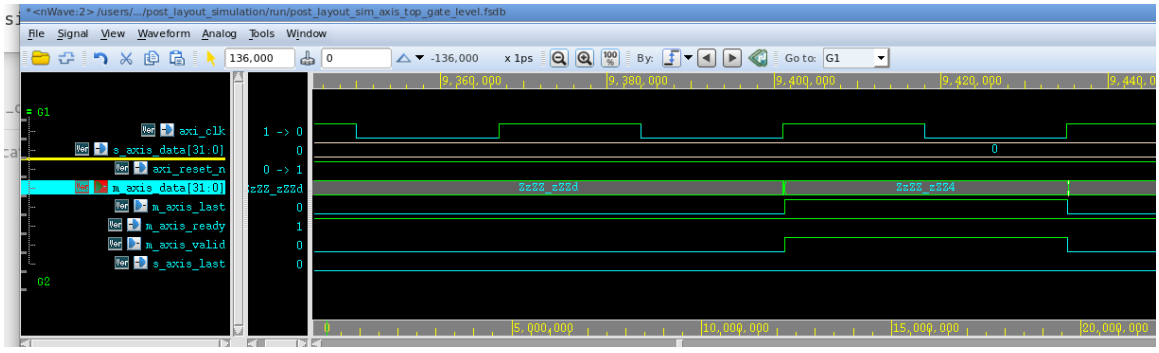

*<nWave:2> /users/.../post_layout_simulation/run/post_layout_sim_axis_top_gate_level.fsdb
File Signal View Waveform Analog Tools Window
2,771,207
-2,771,207
x 1ps
By:
Go to:
G1
axi_clk
s_axis_data[31:0]
axi_reset_n
m_axis_data[31:0]
m_axis_last
m_axis_ready
m_axis_valid
s_axis_last
G2
2,600,000
2,650,000
2,700,000
2,750,000
2,800,000
5,000,000
10,000,000
15,000,000

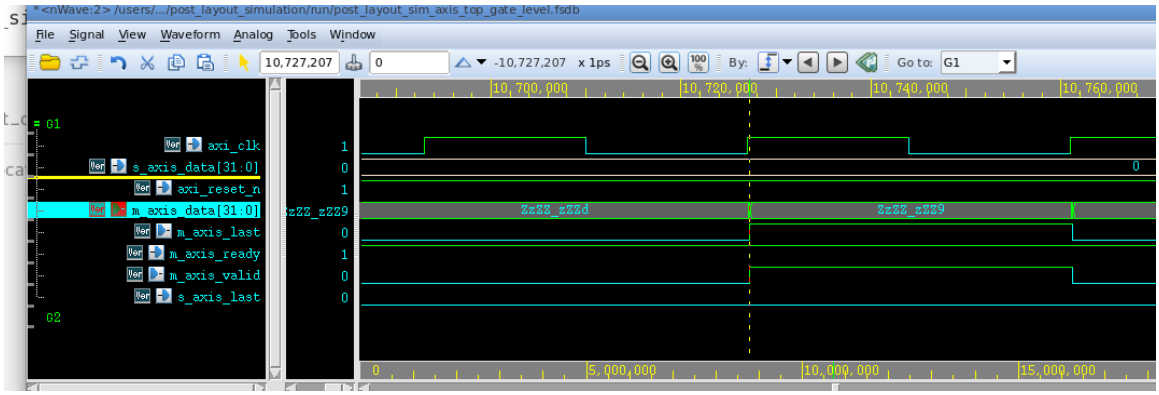

*<nWave:2> /users/.../post_layout_simulation/run/post_layout_sim_axis_top_gate_level.fsdb
File Signal View Waveform Analog Tools Window
4,097,233
-4,097,233
x 1ps
By:
Go to:
G1
axi_clk
s_axis_data[31:0]
axi_reset_n
m_axis_data[31:0]
m_axis_last
m_axis_ready
m_axis_valid
s_axis_last
G2
4,000,000
4,050,000
4,100,000
5,000,000
10,000,000

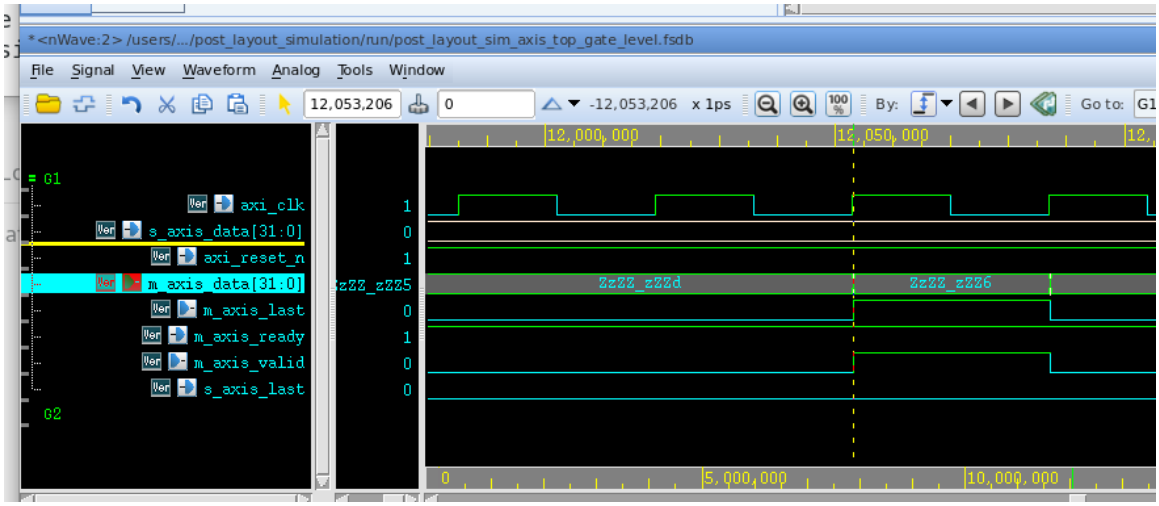

*<nWave:2> /users/.../post_layout_simulation/run/post_layout_sim_axis_top_gate_level.fsdb
File Signal View Waveform Analog Tools Window
5,423,206
-5,423,206
x 1ps
By:
Go to:
G1
axi_clk
s_axis_data[31:0]
axi_reset_n
m_axis_data[31:0]
m_axis_last
m_axis_ready
m_axis_valid
s_axis_last
G2
5,300,000
5,350,000
5,400,000
5,450,000
5,300,000
10,000,000

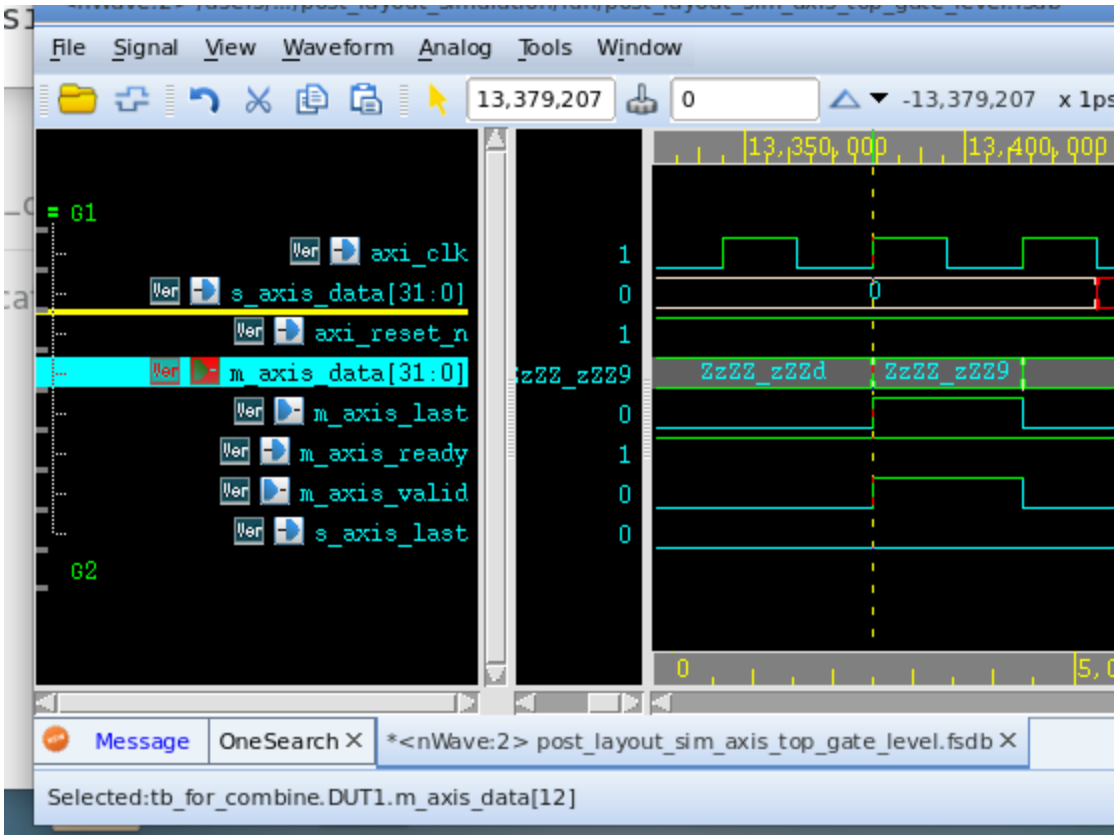

*<nWave:2> /users/.../post_layout_simulation/run/post_layout_sim_axis_top_gate_level.fsdb
File Signal View Waveform Analog Tools Window
6,749,206
-6,749,206
x 1ps
Go to: G1
G1
axi_clk
s_axis_data[31:0]
axi_reset_n
m_axis_data[31:0]
m_axis_last
m_axis_ready
m_axis_valid
s_axis_last
G2
6,650,000
6,700,000
6,750,000
ZzZZ_zZZd
ZzZZ_zZZ4
5,000,000
10,000,000


*<nWave:2> /users/.../post_layout_simulation/run/post_layout_sim_axis_top_gate_level.fsdb
File Signal View Waveform Analog Tools Window
8,075,207
-8,075,207
x 1ps
Go to: G1
G1
axi_clk
s_axis_data[31:0]
axi_reset_n
m_axis_data[31:0]
m_axis_last
m_axis_ready
m_axis_valid
s_axis_last
G2
8,000,000
8,050,000
8,100,000
ZzZZ_zZZd
ZzZZ_zZZ1
5,000,000
10,000,000
Message
OneSearch


*<nWave:2> /users/.../post_layout_simulation/run/post_layout_sim_axis_top_gate_level.fsdb
File Signal View Waveform Analog Tools Window
136,000
-136,000
x 1ps
Go to: G1
G1
axi_clk
s_axis_data[31:0]
axi_reset_n
m_axis_data[31:0]
m_axis_last
m_axis_ready
m_axis_valid
s_axis_last
G2
9,360,000
9,380,000
9,400,000
9,420,000
ZzZZ_zZZd
ZzZZ_zZZ4
5,000,000
10,000,000
15,000,000
20,000,000


*<nWave:2> /users/.../post_layout_simulation/run/post_layout_sim_axis_top_gate_level.fsdb
File Signal View Waveform Analog Tools Window
10,727,207
-10,727,207 x 1ps
By:
Go to: G1
10,700,000
10,720,000
10,740,000
10,760,000
G1
axi_clk
s_axis_data[31:0]
axi_reset_n
m_axis_data[31:0]
m_axis_last
m_axis_ready
m_axis_valid
s_axis_last
G2
1
0
1
zZZ_zZZ9
0
1
0
0
ZzZZ_zZZd
ZzZZ_zZZ9
0
5,000,000
10,000,000
15,000,000


*<nWave:2> /users/.../post_layout_simulation/run/post_layout_sim_axis_top_gate_level.fsdb
File Signal View Waveform Analog Tools Window
12,053,206
-12,053,206 x 1ps
By:
Go to: G1
12,000,000
12,050,000
G1
axi_clk
s_axis_data[31:0]
axi_reset_n
m_axis_data[31:0]
m_axis_last
m_axis_ready
m_axis_valid
s_axis_last
G2
1
0
1
zZZ_zZZ5
0
1
0
0
ZzZZ_zZZd
ZzZZ_zZZ6
0
5,000,000
10,000,000

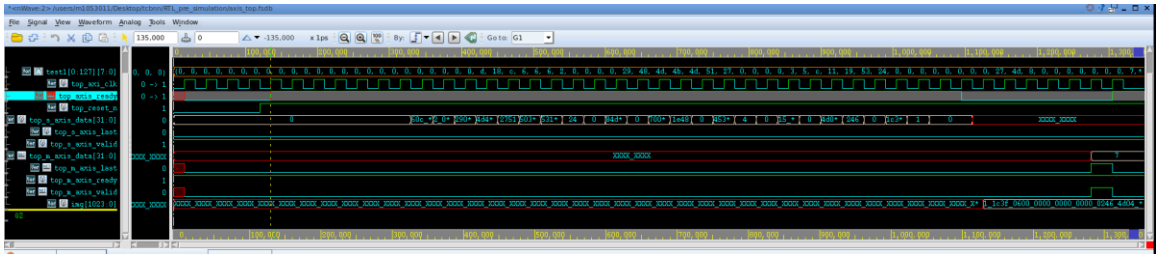

File Signal View Waveform Analog Tools Window
13,379,207
0
-13,379,207
x 1ps
13,350,000
13,400,000
G1
axi_clk
s_axis_data[31:0]
axi_reset_n
m_axis_data[31:0]
m_axis_last
m_axis_ready
m_axis_valid
s_axis_last
G2
1
0
1
zZZ_zZZ9
0
1
0
0
zzzz_zzzd
zzzz_zzz9
Message
OneSearch
*<nWave:2> post_layout_sim_axis_top_gate_level.fsdb
Selected:tb_for_combine.DUT1.m_axis_data[12]

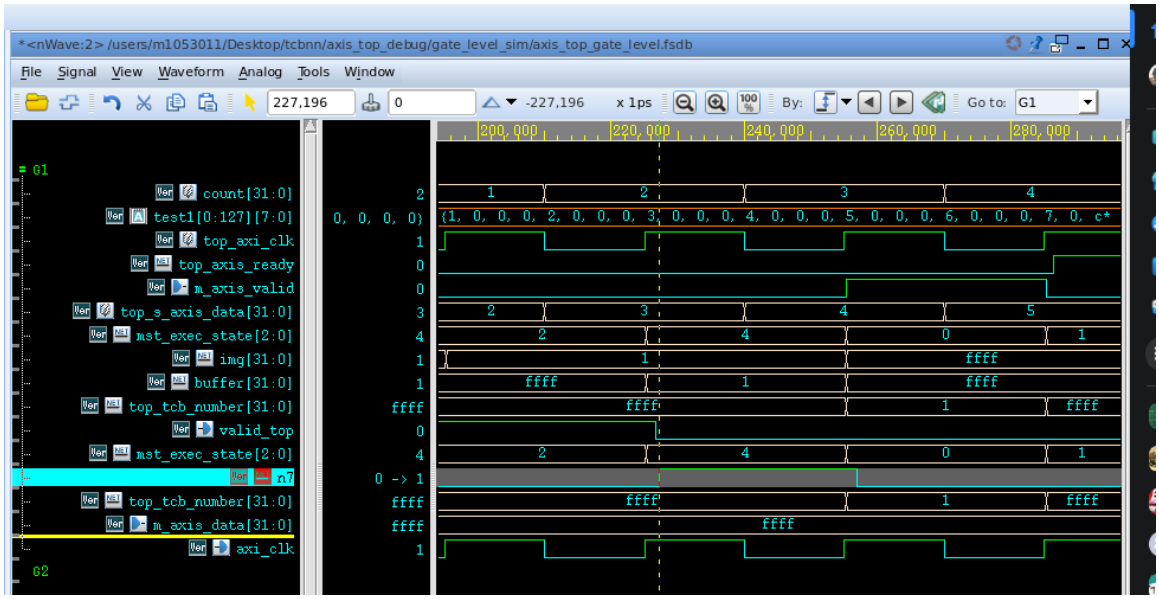
BUG

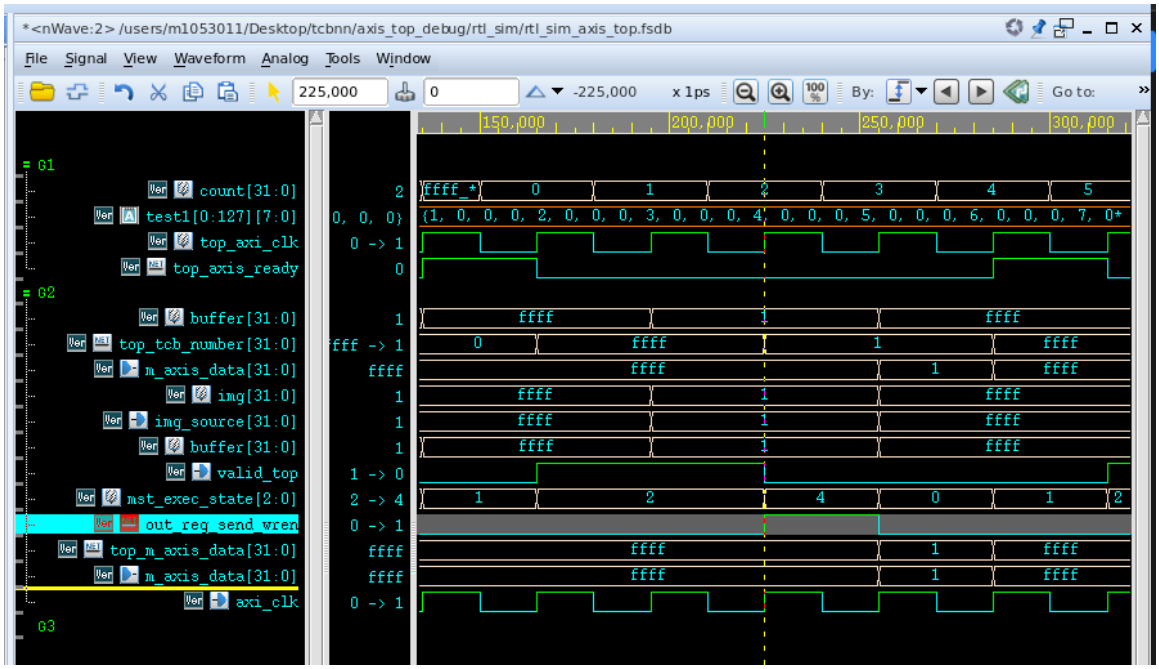
Front-end

Gate-Level_sim

Top_axis_ready Wrong

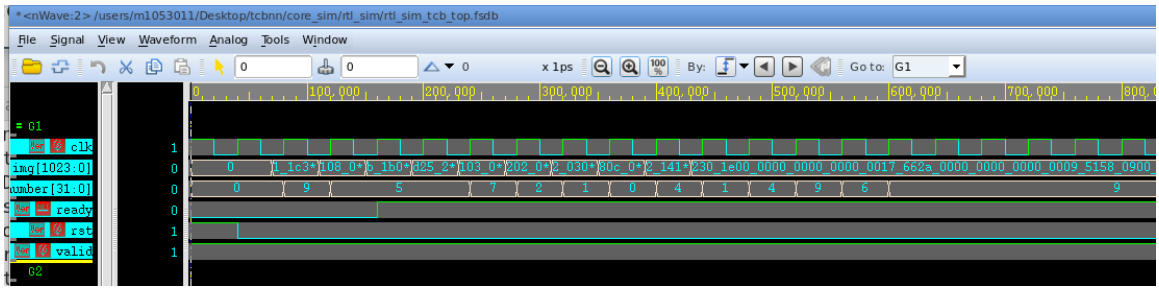




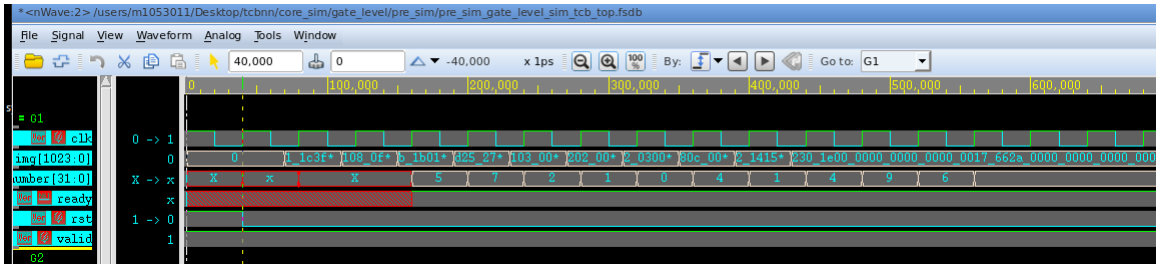


Only core simulation

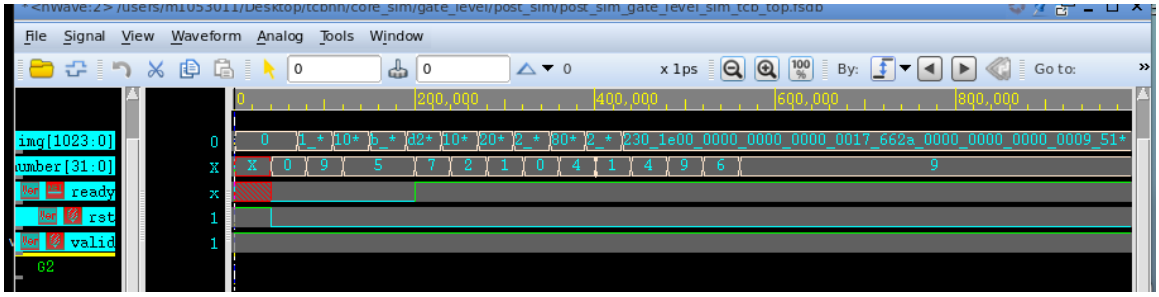
RTL



pre

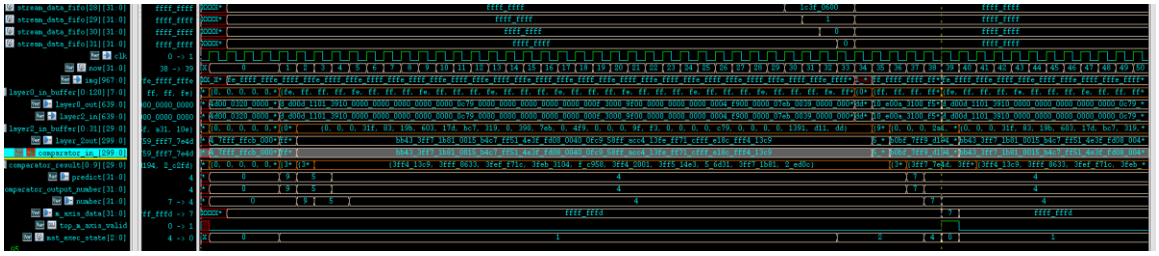


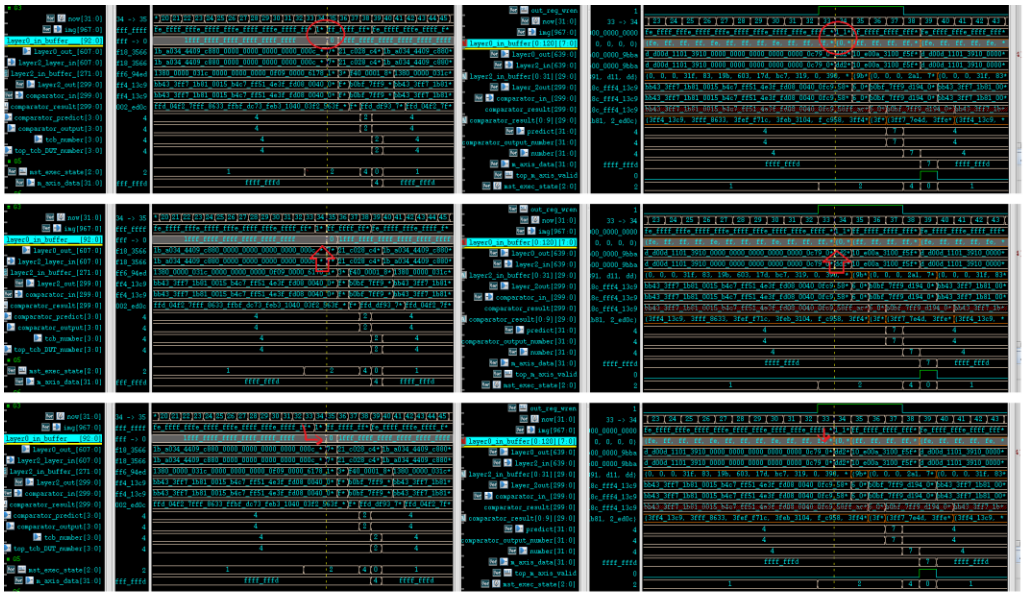
Post_sim



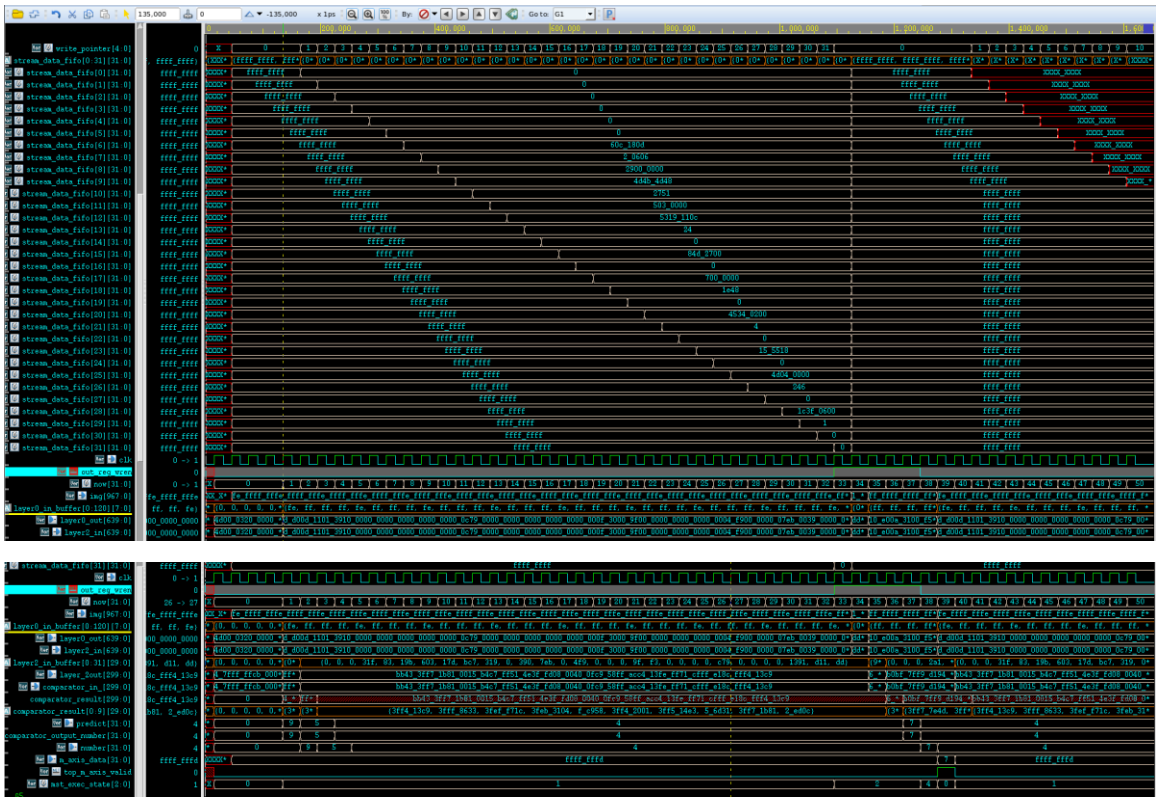
BUG

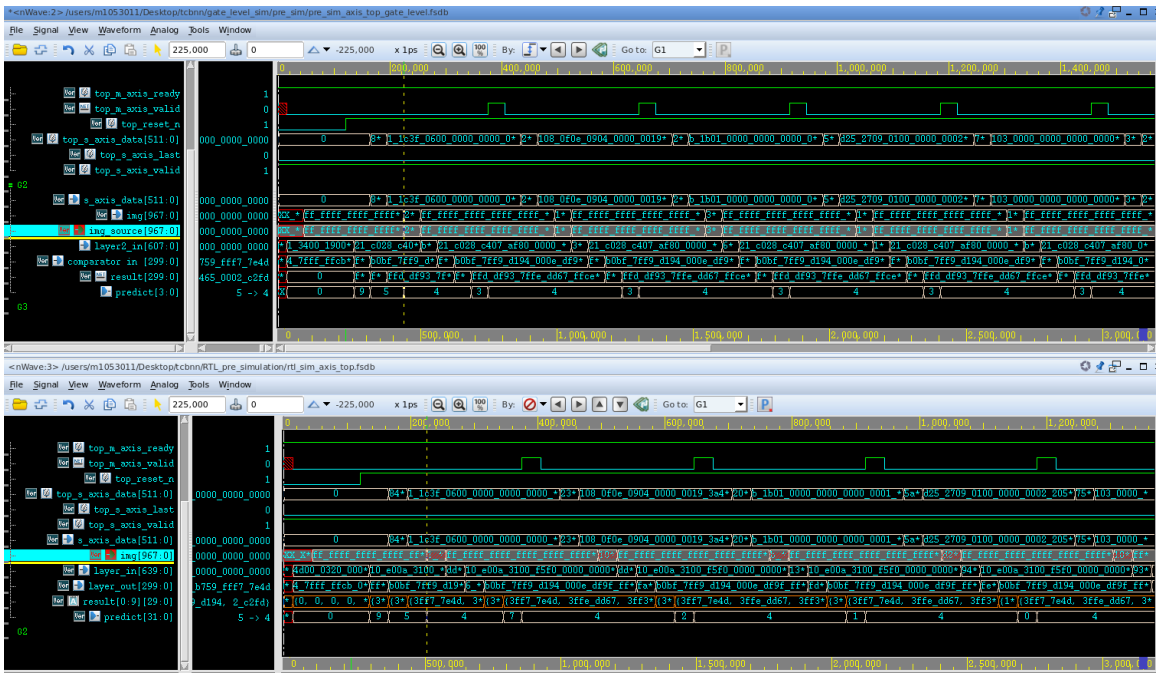
RTL dataflow



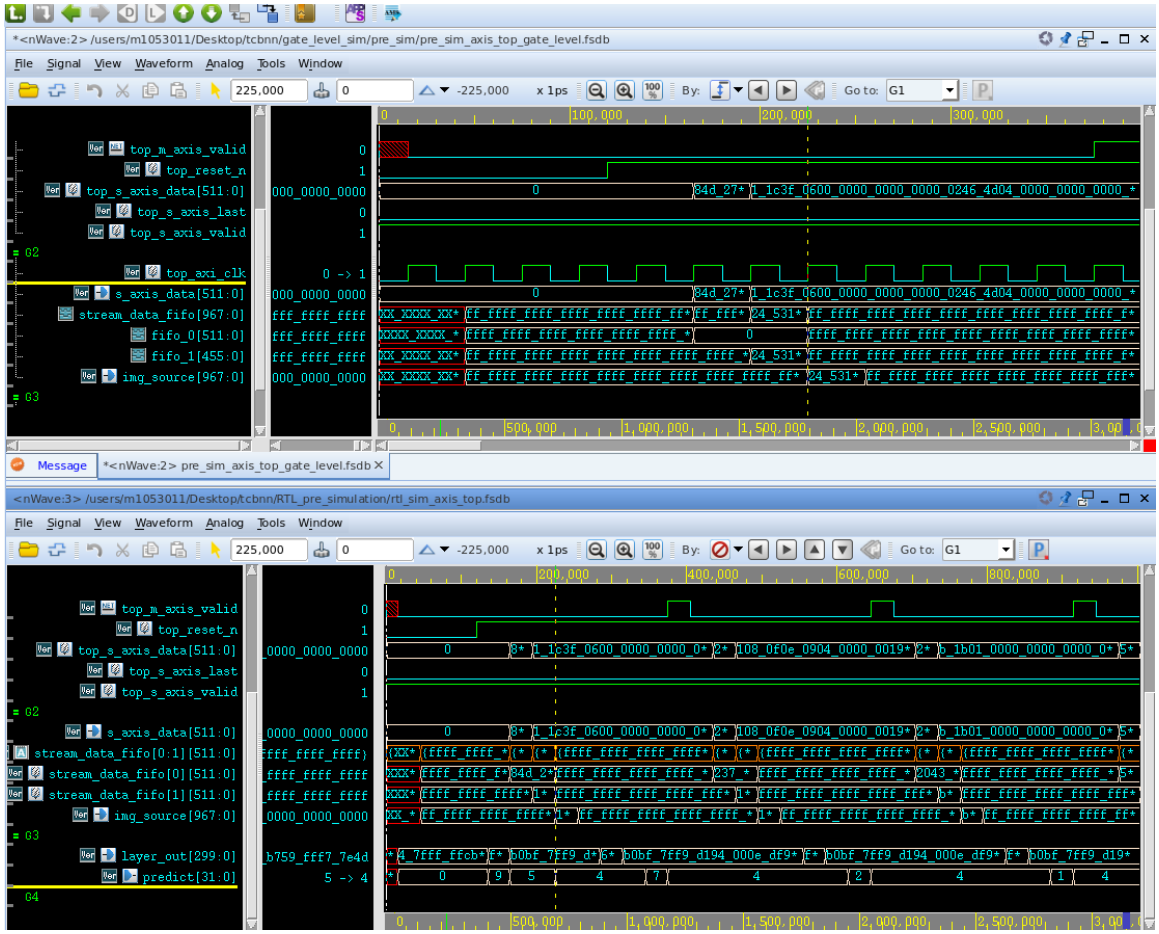


Data Flow Diagram



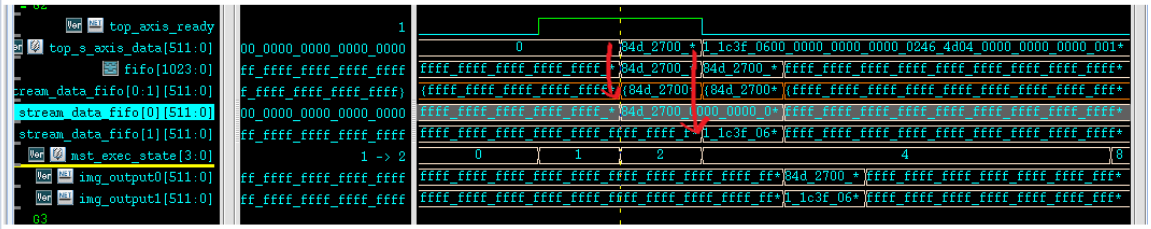


Data Shift 512 bit

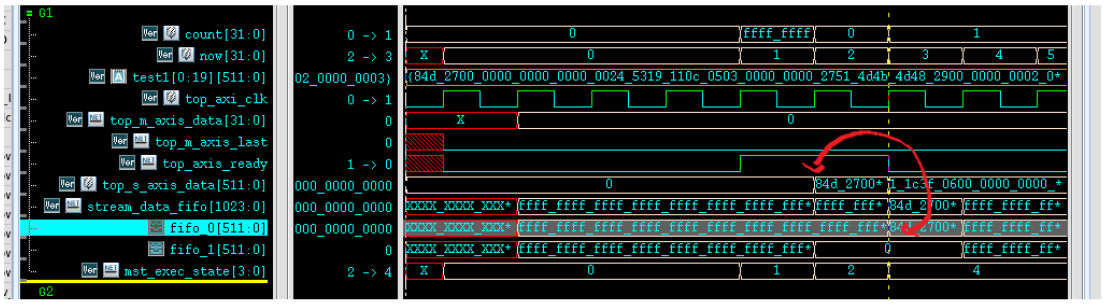


```
1: False
0: True
 11c3f0600000000000000002464d040000000000000015551800000000000000044534020000000000000001e4807000000000000000
84d27000000000000000024531911 0c0503000000002751 4d4b4d4829000000000020606060c180d000000000000000000000000000000
0000000000000000000
 245319110c0503000000027514d4b4d4829000000000020606060c180d000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000
[m1053011@testlab5913 tcbnn]$
```

RTL sim

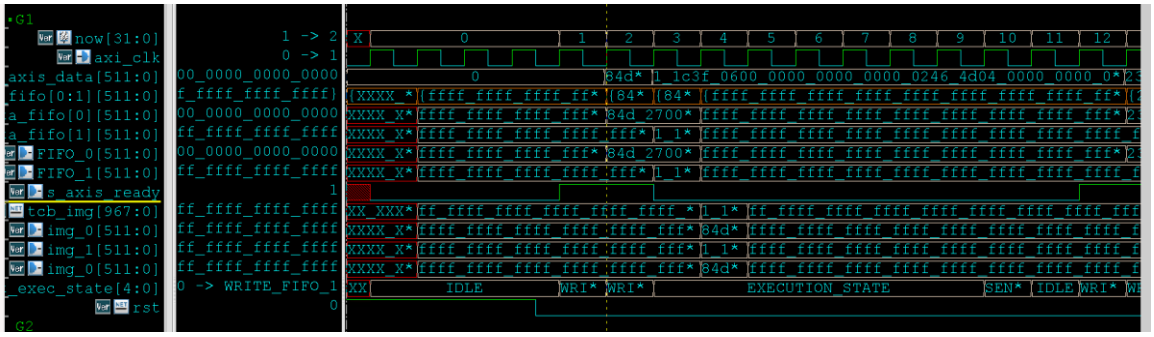


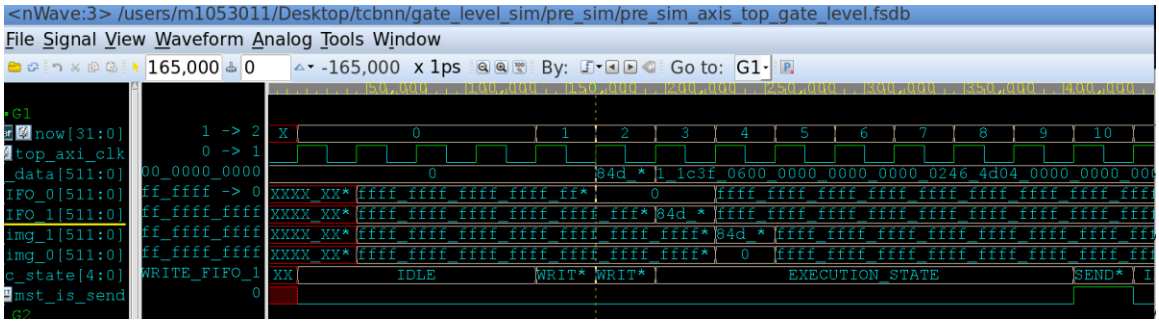
PRE sim




```verilog
//INPUT BUFFER PHASE
reg  [(DATA_WIDTH*16)-1:0] stream_data_fifo [0 : NUMBER_OF_INPUT_WORDS-1];
// FIFO write enable generation
assign fifo_wren = s_axis_valid & s_axis_ready;
// Streaming input data is stored in FIFO
integer i;
always @( posedge axi_clk )
    begin
        if (fifo_wren)
            begin
                stream_data_fifo [(mst_exec_state-1)] <= s_axis_data;
            end
        else
            for(i=0;i<NUMBER_OF_INPUT_WORDS;i=i+1)
                begin
                    stream_data_fifo[i] <= {512{1'b1}};
                end
    end
```

RTL


G1
now[31:0]
axi_clk
axis_data[511:0]
fifo[0:1][511:0]
a_fifo[0][511:0]
a_fifo[1][511:0]
FIFO_0[511:0]
FIFO_1[511:0]
s_axis_ready
tcb_img[967:0]
img_0[511:0]
img_1[511:0]
img_0[511:0]
_exec_state[4:0]
rst
G2
IDLE
WRI*
EXECUTION_STATE
SEN*


<nWave:3> /users/m1053011/Desktop/tcbnn/gate_level_sim/pre_sim/pre_sim_axis_top_gate_level.fsdb
File Signal View Waveform Analog Tools Window
165,000
-165,000
x 1ps
By:
Go to:
G1
now[31:0]
top_axi_clk
_data[511:0]
IFO_0[511:0]
IFO_1[511:0]
img_1[511:0]
img_0[511:0]
c_state[4:0]
mst_is_send
G2
IDLE
WRIT*
EXECUTION_STATE
SEND*

Reference

[1] CIC2011_brief_introduction_of_cell_base_design.ppt

[2] Design_compiler_user_guide_2019_03

[3] https://www.sciencedirect.com/topics/computer-science/register-transfer-level#:~:text=outputting%20a%20netlist.-,3),)%20between%20the%20flip%2Dflops.

[4] https://www.techdesignforums.com/practice/technique/catch-x-propagation-issues-rtl/

[5] https://blog.csdn.net/Holden_Liu/article/details/117065321