VLSI Design Flow: RTL To GDS (NPTEL Course)

Lab Tutorial 11

Objective: To gain a hands-on experience on **Chip-planning (Floorplanning, Powerplanning) and Placement** using OpenROAD app.

OpenROAD: OpenROAD is an integrated chip physical design tool that takes a design from synthesized Verilog to routed layout.

1. **Yosys:** for performing Logic Synthesis.

2. Floorplanning through Detailed Routing: using **OpenROAD**

References:

1. Ajayi, Tutu, Vidya A. Chhabria, Mateus Fogaça, Soheil Hashemi, Abdelrahman Hosny, Andrew B. Kahng, Minsoo Kim et al. "Toward an open-source digital flow: First learnings from the openroad project." In Proceedings of the 56th Annual Design Automation Conference 2019, pp. 1-4. 2019.
2. Ajayi, Tutu, and David Blaauw. "OpenROAD: Toward a self-driving, open-source digital layout implementation tool chain." In Proceedings of Government Microcircuit Applications and Critical Technology Conference. 2019.

Website:

https://theopenroadproject.org/

OpenRoad Community:

https://gitter.im/The-OpenROAD-Project/community

OpenRoad Documentation:

To see the available options for any OpenROAD command, you can type the following command in the openroad shell:

openroad> **help [pattern]**

Example 1:
openroad> **help macr***
This will generate a comprehensive listing of all relevant commands and their corresponding options that match with the pattern "macr*"
Example 2:
openroad> **help macro_placement**
This will list all the options of the openroad command macro_placement.

However, if you want to see the description of any command and their options, you can refer to the link provided below. On the left-hand side of the page, distinct sections are organized based on various stages, each delineating specific commands pertinent to that stage:

https://openroad.readthedocs.io/en/latest/

<u>Activity</u>: *Take any chip planning tool that is available to you (I am using OpenROAD tool) and carry out the following activities. Load a Verilog netlist. You should give the timing constraints, technology library, and physical information (possibly using LEF files) as inputs.*

TASK 1: Initialize floorplan by assuming some reasonable value of die size and core size.
Perform IO pin placement.
Perform macro placement.
Perform tap cell and well tie insertion.

TASK 2: Perform Power Planning.

TASK 3: Perform global placement
Perform optimized IO pin placement.
Perform placement optimization and repairing: resizing and buffering.
Perform placement legalization.
Perform detailed placement.

Script used to execute activity:

gcd_nangate45.tcl
(Location: OpenROAD/test/gcd_nangate45.tcl)

Inputs to the OpenRoad Tool:

1. RTL Netlist: gcd_nangate45.v
(Location: OpenROAD/test/gcd_nangate45.v)
2. SDC file: gcd_nangate45.sdc
(Location: OpenROAD/test/gcd_nangate45.sdc)
3. Library file: Nangate45_typ.lib
(Location: OpenROAD/test/Nangate45/Nangate45_typ.lib)
4. LEF file
    A. Technology Lef: Nangate45_tech.lef
    (Location: OpenROAD/test/Nangate45/Nangate45_tech.lef)
    B. Standard Cell Lef: Nangate45_stdcell.lef
    (Location: OpenROAD/test/Nangate45/Nangate45_stdcell.lef)

NOTE:
a. Technology LEF file contains information about technology site, available layers for routing, layer's physical information (pitch, width, spacing, mask, direction etc.), DRC rules, VIA definition, Non-Default Rules (NDR).
b. Cell LEF file contains information about abstract view of macro and standard cells

1. gcd_nangate45.tcl

```
# gcd flow pipe cleaner
source "helpers.tcl"
source "flow_helpers.tcl"
source "Nangate45/Nangate45.vars"

set design "gcd"
set top_module "gcd"
set synth_verilog "gcd_nangate45.v"
set sdc_file "gcd_nangate45.sdc"
set die_area {0 0 100.13 100.8}
set core_area {10.07 11.2 90.25 91}

source -echo "flow.tcl”
# You can divide the flow.tcl file into smaller sub-files to systematically analyze the distinct
stages of the Physical Design process as demonstrated in the tutorial.
```

2. flow.tcl
(Location: OpenROAD/test/flow.tcl)

```
# Assumes flow_helpers.tcl has been read.
read_libraries
read_verilog $synth_verilog
link_design $top_module
read_sdc $sdc_file

utl::metric "IFP::ord_version" [ord::openroad_git_describe]
# Note that sta::network_instance_count is not valid after tapcells are added.
utl::metric "IFP::instance_count" [sta::network_instance_count]

initialize_floorplan -site $site \
  -die_area $die_area \
  -core_area $core_area

write_def gcd/post_floorplan.def
source $tracks_file

# remove buffers inserted by synthesis
remove_buffers

################################################################
# IO Placement (random)
place_pins -random -hor_layers $io_placer_hor_layer -ver_layers $io_placer_ver_layer

################################################################
# Macro Placement
```

```
if { [have_macros] } {
  global_placement -density $global_place_density
  macro_placement -halo $macro_place_halo -channel $macro_place_channel
}

write_def gcd/post_macro_placement.tcl
####################################################################
# Tapcell insertion
eval tapcell $tapcell_args

write_def gcd/post_tapcell.def
####################################################################
# Power distribution network insertion
source $pdn_cfg
pdngen

write_def gcd/post_pdn.def
####################################################################
# Global placement

foreach layer_adjustment $global_routing_layer_adjustments {
  lassign $layer_adjustment layer adjustment
  set_global_routing_layer_adjustment $layer $adjustment
}
set_routing_layers -signal $global_routing_layers \
  -clock $global_routing_clock_layers
set_macro_extension 2

global_placement -routability_driven -density $global_place_density \
  -pad_left $global_place_pad -pad_right $global_place_pad

# IO Placement
place_pins -hor_layers $io_placer_hor_layer -ver_layers $io_placer_ver_layer

# checkpoint
#set global_place_db [make_result_file ${design}_${platform}_global_place.db]
write_db gcd/gcd_nangate45_global_place_db
write_def post_global_placement.def

####################################################################
# Repair max slew/cap/fanout violations and normalize slews
source $layer_rc_file
set_wire_rc -signal -layer $wire_rc_layer
set_wire_rc -clock  -layer $wire_rc_layer_clk
set_dont_use $dont_use

estimate_parasitics -placement
```

```
repair_design -slew_margin $slew_margin -cap_margin $cap_margin

repair_tie_fanout -separation $tie_separation $tielo_port
repair_tie_fanout -separation $tie_separation $tiehi_port

#################################################################
set_placement_padding -global -left $detail_place_pad -right $detail_place_pad
detailed_placement

# post resize timing report (ideal clocks)
report_worst_slack -min -digits 3
report_worst_slack -max -digits 3
report_tns -digits 3
# Check slew repair
report_check_types -max_slew -max_capacitance -max_fanout -violators

utl::metric "RSZ::repair_design_buffer_count" [rsz::repair_design_buffer_count]
utl::metric "RSZ::max_slew_slack" [expr [sta::max_slew_check_slack_limit] * 100]
utl::metric "RSZ::max_fanout_slack" [expr [sta::max_fanout_check_slack_limit] * 100]
utl::metric "RSZ::max_capacitance_slack" [expr [sta::max_capacitance_check_slack_limit] *
100]

write_verilog post_detailed_placement.v
write_def post_detailed_placement.def
```