Prompts

What is PDNGEN?

What is the essence of PDNGEN?

What is PDNGEN?

What does the -switch_cell argument in PDN do while doing power switch insertion?

How does the -switch_cell argument function in PDNGEN during power switch integration?

What does the -switch_cell argument in PDN do while doing power switch insertion?

What does the par module do?

What functionality does the par module provide?

What does the par module do?

Is TritonRoute the foundation of drt?

Does TritonRoute serve as the underlying technology for drt?

Is TritonRoute the foundation of drt?

What is the structure of OpenDB?

Can you describe the architecture of OpenDB?

What is the structure of OpenDB?

How is FastRoute better than previous routing frameworks?

In what ways is FastRoute superior to its predecessors in routing frameworks?

How is FastRoute better than previous routing frameworks?

What does RTLMP do?

What tasks does RTLMP accomplish?

What does RTLMP do?

What was the need to develop Hier-RTLMP?

What prompted the creation of Hier-RTLMP?

What was the need to develop Hier-RTLMP?

How is Hier-RTLMP different from RTL-MP?

In what aspects does Hier-RTLMP diverge from RTL-MP?

How is Hier-RTLMP different from RTL-MP?

How does FastRoute work?

Can you explain the operating principle of FastRoute?

How does FastRoute work?

Why is RSMT used more?

Why is RSMT favored over other methods?

Why is RSMT used more?

Tab completion does work in console mode but not in GUI mode, why?

Why does tab completion function in console mode but fail in GUI mode?

Tab completion does work in console mode but not in GUI mode, why?

What do these Debug output mean?
[INFO MPL-0024] [Multilevel Autoclustering] Creating clustered netlist.
[INFO MPL-0039] [Coarse Shaping] Determining shape functions for clusters.
[INFO MPL-0028] [Hierarchical Macro Placement] Placing clusters and macros.
[INFO MPL-0037] Updated location of 95 macros
Delete buffers for RTLMP flow...
[INFO RSZ-0026] Removed 0 buffers.

What are the units of the -pad_right and -pad_left arguments of the global_placement function

What are the measurement units for the -pad_right and -pad_left arguments in the global_placement function?

What are the units of the -pad_right and -pad_left arguments of the global_placement function

Does space padding influence the design utilization? I feel like it shouldn't.

Does padding affect design utilization, in your opinion?

Does space padding influence the design utilization? I feel like it shouldn't.

Wouldn't manually adding pads in the global_placement function reduce the maximum design utilization possible for a design? Especially since OpenROAD wants to be automatic, GPL will stop everything if it sees a DU >100% even if it could achieve a reasonable DU with equivalent padding

Could manually adding pads in the global_placement function affect the maximum design utilization?

Wouldn't manually adding pads in the global_placement function reduce the maximum design utilization possible for a design? Especially since OpenROAD wants to be automatic, GPL will stop everything if it sees a DU >100% even if it could achieve a reasonable DU with equivalent padding

Why did I encounter an issue with CORE_UTILIZATION when trying to aim for a DU of 70% on the ASAP7 PDK which should be possible and reasonable for any design?

Why did I face a CORE_UTILIZATION issue when targeting a DU of 70% with the ASAP7 PDK?

Why did I encounter an issue with CORE_UTILIZATION when trying to aim for a DU of 70% on the ASAP7 PDK which should be possible and reasonable for any design?

If I have a design for which the detailed placement fails for a few instances, if I relax my timing constraints will it result in a successful placement?

If detailed placement fails for certain instances in my design, would loosening timing constraints help achieve placement?

If I have a design for which the detailed placement fails for a few instances, if I relax my timing constraints will it result in a successful placement?
The timing optimizations done by the synthesis tool are "discarded" as the buffers are removed?

Are the timing adjustments made by the synthesis tool lost when buffers are eliminated?

The timing optimizations done by the synthesis tool are "discarded" as the buffers are removed?
Is there no need to re-synthesize with different timing constraints with Yosys? Or does it also use different/bigger non-buffer cells that also need to be resized?

Is re-synthesizing with altered timing constraints unnecessary with Yosys, or does it also resize non-buffer cells?

Is there no need to re-synthesize with different timing constraints with Yosys? Or does it also use different/bigger non-buffer cells that also need to be resized?
What does this warning mean?
[WARNING ODB-0208] VIA: duplicate VIA (via5_6_120_288_1_2_58_322)

Upon attempting to read an invalid ODB file, OpenROAD doesn't attempt to validate it and instead crashes with a cryptic message of Error: read_db.tcl, 1 ios_base::clear: unspecified iostream_category error. Why is this happening?

Why does OpenROAD crash without attempting to validate an invalid ODB file?

Upon attempting to read an invalid ODB file, OpenROAD doesn't attempt to validate it and instead crashes with a cryptic message of Error: read_db.tcl, 1 ios_base::clear: unspecified iostream_category error. Why is this happening?

What is IR Drop Analysis?

Can you explain IR Drop Analysis and its purposes?

What is IR Drop Analysis?

What are the features of IR Drop analysis?

What characteristics define IR Drop analysis?

What are the features of IR Drop analysis?

What is Flute3?

What distinguishes Flute3?

What is Flute3?

What is OpenDB?

What is the role of OpenDB?

What is OpenDB?

What is Automatic Code Generator used for?

What is the purpose behind the Automatic Code Generator?

What is Automatic Code Generator used for?

What commands are used to read and write design data in OpenROAD?

What support does OpenROAD have for Abstract LEF?

How does OpenROAD support Abstract LEF?

What support does OpenROAD have for Abstract LEF?

What does the global_connect command do?

What function does the global_connect command serve?

What does the global_connect command do?

What does the clear_global_connect command do?

What is the effect of using the clear_global_connect command?

What does the clear_global_connect command do?

What does the report_global_connect command do?

What information does the report_global_connect command provide?

What does the report_global_connect command do?

What does the report_cell_usage command do?

How does the report_cell_usage command assist users?

What does the report_cell_usage command do?

How does OpenROAD compute the die area when using the core_utilization argument in the initialize_floorplan?

How does OpenROAD determine die area with the core_utilization argument during floorplan initialization?

How does OpenROAD compute the die area when using the core_utilization argument in the initialize_floorplan?

I would like to know if there is any way to write the log output from OpenROAD into a file (using a report_ type command)?

Is there a method to direct OpenROAD's log output to a file?

I would like to know if there is any way to write the log output from OpenROAD into a file (using a report_ type command)?

What is the minimum number of metal layers OpenROAD can route in?

What is the minimum metal layer count that OpenROAD can handle for routing?

What is the minimum number of metal layers OpenROAD can route in?

Can OpenROAD work with multi-VT cells (HVT, LVT, SVT) and swap between them in a single run?

Can OpenROAD manage cells with different threshold voltages within a single

Can OpenROAD work with multi-VT cells (HVT, LVT, SVT) and swap between them in a single run?

Can OpenROAD work with Multi-Mode-Multi-Corner Files (mmmc) ?

Is OpenROAD compatible with Multi-Mode-Multi-Corner Files?

Can OpenROAD work with Multi-Mode-Multi-Corner Files (mmmc) ?

Is SystemVerilog support limited to the constructs that Yosys supports?

Are there limitations to SystemVerilog support based on Yosys compatibility?

Is SystemVerilog support limited to the constructs that Yosys supports?

What is the job of Pin Placer?

What is the function of the Pin Placer tool?

What is the job of Pin Placer?

What does Antenna Rule Checker do?

How does the Antenna Rule Checker contribute to design integrity?

What does Antenna Rule Checker do?

What is Clock Tree Synthesis in OpenROAD?

What entails Clock Tree Synthesis in OpenROAD?

What is Clock Tree Synthesis in OpenROAD?

Tell me about Detailed Placement in OpenROAD?

What does Detailed Placement involve in OpenROAD?

Tell me about Detailed Placement in OpenROAD?

Describe the Restructure module in OpenROAD?

Can you outline the Restructure module's role in OpenROAD?

Describe the Restructure module in OpenROAD?

What is RePlAce in OpenROAD?

What is RePlAce and its function in OpenROAD?

What is RePlAce in OpenROAD?

What is Hierarchical Macro Placement/ Hier-RTLMP?

How is Hierarchical Macro Placement defined within OpenROAD?

What is Hierarchical Macro Placement/ Hier-RTLMP?

Describe Parallax Static Timing Analyzer or OpenSTA?

What capabilities does the Parallax Static Timing Analyzer offer?

Describe Parallax Static Timing Analyzer or OpenSTA?

What file formats are supported by Parallax Static Timing Analyzer or OpenSTA?

Which file formats does Parallax Static Timing Analyzer accept?

What file formats are supported by Parallax Static Timing Analyzer or OpenSTA?
`auto_place_pins pin_layer` places pins on a single layer but It should be able to place vertical pins (sides of the die) and horizontal pins (top and bottom of the die) in separate layers, why is it not able to do this?

Why can't auto_place_pins pin_layer place pins on separate layers for vertical and horizontal orientations?

`auto_place_pins pin_layer` places pins on a single layer but It should be able to place vertical pins (sides of the die) and horizontal pins (top and bottom of the die) in separate layers, why is it not able to do this?

What is DFT?

Can you elaborate on DFT?

What is DFT?

Tell me about the parts of DFT insertion?

Could you describe the components involved in DFT insertio

Tell me about the parts of DFT insertion?

What is Read UPF Utility?

What functionality does the Read UPF Utility offer?

What is Read UPF Utility?

What is Metal fill?

What is the purpose of Metal fill in OpenROAD?

What is Metal fill?

Explain Chip-level Connections in OpenROAD?

How are chip-level connections managed in OpenROAD?

Explain Chip-level Connections in OpenROAD?

Brief me on the parasitics extraction module?

Can you brief me on the parasitics extraction module?

Brief me on the parasitics extraction module?

What are the Gate Resizer commands?

What are the functionalities of the Gate Resizer commands?

What are the Gate Resizer commands?

What is macro placement?

What is macro placement and its significance in OpenROAD?

What is macro placement?

What is global routing?

Can you explain global routing and its importance?

What is global routing?

Elaborate on FastRoute?

How does FastRoute enhance routing processes?

Elaborate on FastRoute?

Brief me on OpenROAD Flow?

What is OpenROAD Flow and its benefits?

Brief me on OpenROAD Flow?

I'm trying to port ORFS to a TSMC process. During the detailed route step, I get a lot of DRT-0073 errors. Utilization is set to 20% at the beginning of the P&R flow and it is a pretty simple design. Give me some tips about what to look into?
I have tried to run the pin_access command directly but the -verbose parameter does not seem to work as I don't get any more information

I'm encountering DRT-0073 errors during detailed routing in ORFS; any advice?

I'm trying to port ORFS to a TSMC process. During the detailed route step, I get a lot of DRT-0073 errors. Utilization is set to 20% at the beginning of the P&R flow and it is a pretty simple design. Give me some tips about what to look into?
I have tried to run the pin_access command directly but the -verbose parameter does not seem to work as I don't get any more information

Is there any way I can use just the RTL-MP2 stand-alone on either an RTL design or a gate netlist? The issue is, that we use proprietary tools so do not have an OpenDB database for our designs.

Is it possible to use RTL-MP2 standalone for RTL designs or gate netlists without an OpenDB database?

Is there any way I can use just the RTL-MP2 stand-alone on either an RTL design or a gate netlist? The issue is, that we use proprietary tools so do not have an OpenDB database for our designs.

How do I check DRC?

How do I perform a DRC check in OpenROAD?

How do I check DRC?

What does the argument -floorplan_initialize do in read_def?

What does the -floorplan_initialize argument achieve in read_def?

What does the argument -floorplan_initialize do in read_def?

What does the argument -skip_pin_swap & -skip_gate_cloning do in repair_timing?

What are the functions of -skip_pin_swap & -skip_gate_cloning in repair_timing?

What does the argument -skip_pin_swap & -skip_gate_cloning do in repair_timing?

What does the detailed_placement command do in OpenROAD?

What is the purpose of the detailed_placement command?

What does the detailed_placement command do in OpenROAD?

What does the argument -max_displacement disp|{disp_x disp_y} do in detailed_placement
Command?

How does the -max_displacement argument affect the detailed_placement command?

What does the argument -max_displacement disp|{disp_x disp_y} do in detailed_placement
Command?

What does the argument -disallow_one_site_gaps do in the detailed_placement command?

What does the -disallow_one_site_gaps argument do in detailed_placement?

What does the argument -disallow_one_site_gaps do in the detailed_placement command?

What does the argument -report_file_name do in the detailed_placement command?

What role does the -report_file_name argument play in the detailed_placement command?

What does the argument -report_file_name do in the detailed_placement command?

What does the Set Placement Padding command do?

How does the Set Placement Padding command optimize placement?

What does the Set Placement Padding command do?

What is the significance of the filler_placement command?

What is the significance of the filler_placement command?

What is the significance of the filler_placement command?

What is the purpose of the remove_fillers command?

What does the remove_fillers command achieve?

What is the purpose of the remove_fillers command?

What does the check_placement command do?

What does the check_placement command verify?

What does the check_placement command do?

What does the argument -verbose in the check_placement command do?

How does the -verbose argument in the check_placement command enhance its functionality?

What does the argument -verbose in the check_placement command do?

What does the argument -disallow_one_site_gaps in the check_placement command do?

What effect does the -disallow_one_site_gaps argument have in the check_placement command?

What does the argument -disallow_one_site_gaps in the check_placement command do?

What role does this argument -report_file_name play in the check_placement command?

What information does the -report_file_name argument provide in the check_placement command?

What role does this argument -report_file_name play in the check_placement command?

What does the optimize_mirroring command do?

What optimizations does the optimize_mirroring command perform?

What does the optimize_mirroring command do?

What are some useful developer commands in the detailed placement module in OpenROAD (dpl)?

Can you list some advanced developer commands in the detailed placement module (dpl)?
What are some useful developer commands in the detailed placement module in OpenROAD (dpl)?
What does the argument [-max_length <int>] do in the set_dft_config command of DFT- Design For Testing?
What does the [-max_length <int>] argument in the set_dft_config command control?
What does the argument [-max_length <int>] do in the set_dft_config command of DFT- Design For Testing?

What does the argument [-clock_mixing] do in the set_dft_config command of DFT?

How does the [-clock_mixing] argument in the set_dft_config command affect DFT?
What does the argument [-clock_mixing] do in the set_dft_config command of DFT?
What does the report_dft_config command do in DFT- Design For Testing?
What insights does the report_dft_config command offer in DFT- Design For Testing?
What does the report_dft_config command do in DFT- Design For Testing?

What does the preview_dft command do in DFT- Design For Testing?
What preview does the preview_dft command provide in DFT- Design For Testing?

What does the preview_dft command do in DFT- Design For Testing?
What does the argument [-verbose] do in the preview_dft command of DFT- Design For Testing?
How does the [-verbose] argument in the preview_dft command enhance its output?
What does the argument [-verbose] do in the preview_dft command of DFT- Design For Testing?

What does the insert_dft command do in DFT- Design For Testing?
What is the process of the insert_dft command in DFT- Design For Testing?

What does the insert_dft command do in DFT- Design For Testing?

Can you give me an example of a basic Design for Testing command?

Could you provide a basic example of a Design for Testing command?

Can you give me an example of a basic Design for Testing command?

What are the limitations of Design for Testing (DFT)?

What are the known limitations of Design for Testing (DFT)?

What are the limitations of Design for Testing (DFT)?

What is the report_cts command in Clock Tree Synthesis (cst) in OpenROAD used for?

What is the purpose of the report_cts command in Clock Tree Synthesis (cts) in OpenROAD?

What is the report_cts command in Clock Tree Synthesis (cst) in OpenROAD used for?

What does the argument -out_file in report_cts command in Clock Tree Synthesis (cst) in OpenROAD do?

What does the -out_file argument in the report_cts command achieve?

What does the argument -out_file in report_cts command in Clock Tree Synthesis (cst) in OpenROAD do?

What does the clock_tree_synthesis_debug command in Clock Tree Synthesis (cst) in OpenROAD do?

How does the clock_tree_synthesis_debug command assist in CST troubleshooting?

What does the clock_tree_synthesis_debug command in Clock Tree Synthesis (cst) in OpenROAD do?

Which environment is required for setting up OpenROAD flow scripts?

What setup is necessary for OpenROAD flow scripts?

Which environment is required for setting up OpenROAD flow scripts?

What are the ways of installing OpenROAD flow scripts/ ORFS?

What installation methods exist for OpenROAD flow scripts/ORFS?

What are the ways of installing OpenROAD flow scripts/ ORFS?

What is the basic build command for OpenROAD flow scripts/ ORFS?

What is the fundamental build command for OpenROAD flow scripts/ORFS?

What is the basic build command for OpenROAD flow scripts/ ORFS?

What does the following argument do in the build command of ORFS: -o or

How does the -o or —local argument function in the build command of ORFS?

What does the following argument do in the build command of ORFS: -o or

What operation does the -l or --latest argument perform in the build command of ORFS?

What is accomplished by the -l or --latest argument in the build command of ORFS?

What operation does the -l or --latest argument perform in the build command of ORFS?

How is this argument utilized in the build command of ORFS: --or_branch BRANCH_NAME?

How is the --or_branch BRANCH_NAME argument used in the build command of ORFS?

How is this argument utilized in the build command of ORFS: --or_branch BRANCH_NAME?

What role does this argument play in the build command of ORFS: --or_repo REPO_URL?

What does the --or_repo REPO_URL argument specify in the build command of ORFS?

What role does this argument play in the build command of ORFS: --or_repo REPO_URL?

What does the following argument do in the build command of ORFS: --no_init?

What does the --no_init argument accomplish in the build command of ORFS?

What does the following argument do in the build command of ORFS: --no_init?

How does the following argument function in the build command of ORFS: -t N or --threads N?

What does the -t N or --threads N argument control in the build command of ORFS?

How does the following argument function in the build command of ORFS: -t N or --threads N?

What is the purpose of the following argument in the build command of ORFS: -n or --nice?

What is the purpose of the -n or --nice argument in the build command of ORFS?

What is the purpose of the following argument in the build command of ORFS: -n or --nice?

What does the following argument do in the build command of ORFS: —yosys-args-overwrite?

What does the --yosys-args-overwrite argument modify in the build command of ORFS?

What does the following argument do in the build command of ORFS: —yosys-args-overwrite?

What is the purpose of the following argument in the build command of ORFS: —yosys-args STRING?

How does the --yosys-args STRING argument customize the build process in ORFS?

What is the purpose of the following argument in the build command of ORFS: —yosys-args STRING?

What function does the following argument serve in the build command of ORFS: —openroad-args-overwrite?

What modification does the --openroad-args-overwrite argument make in the build command of ORFS?

What function does the following argument serve in the build command of ORFS: —openroad-args-overwrite?

What does the following argument do in the build command of ORFS: —openroad-args STRING?

How does the --openroad-args STRING argument alter the build process in ORFS?

What does the following argument do in the build command of ORFS: —openroad-args STRING?

What is the purpose of the following argument in the build command of ORFS: —lsoracle-enable?

What is the function of the --lsoracle-enable argument in the build command of ORFS?

What is the purpose of the following argument in the build command of ORFS: —lsoracle-enable?

What does the following argument do in the build command of ORFS: —lsoracle-args-overwrite?

What does the --lsoracle-args-overwrite argument change in the build command of ORFS?

What does the following argument do in the build command of ORFS: —lsoracle-args-overwrite?

What does the following argument achieve in the build command of ORFS: —lsoracle-args STRING?

How does the --lsoracle-args STRING argument customize the ORFS build process?

What does the following argument achieve in the build command of ORFS: —lsoracle-args STRING?

What function does the following argument serve in the build command of ORFS: —install-path PATH?

What does the --install-path PATH argument specify in the build command of ORFS?

What function does the following argument serve in the build command of ORFS: —install-path PATH?

What does the following argument do in the build command of ORFS: —clean?

What is achieved by the --clean argument in the build command of ORFS?

What does the following argument do in the build command of ORFS: —clean?

What does the following argument do in the build command of ORFS: —clean-force?

How does the --clean-force argument function in the build command of ORFS?

What does the following argument do in the build command of ORFS: —clean-force?

What function does the following argument serve in the build command of ORFS: -c or --copy-platforms?

What does the -c or --copy-platforms argument do in the build command of ORFS?

What function does the following argument serve in the build command of ORFS: -c or --copy-platforms?

What does the following argument do in the build command of ORFS: —docker-args-overwrite?

What modification does the --docker-args-overwrite argument make in the build command of ORFS?

What does the following argument do in the build command of ORFS: —docker-args-overwrite?

What does the following argument do in the build command of ORFS: —docker-args STRING?

How does the --docker-args STRING argument customize the ORFS build process?

What does the following argument do in the build command of ORFS: —docker-args STRING?

What is OpenROAD?

Can you provide an overview of OpenROAD and its functionalities?

What is OpenROAD?

What is AutoTuner?

What is AutoTuner and its purpose in OpenROAD?

What is AutoTuner?

WHat are the current supported search algorithms by AutoTuner?

Which search algorithms are currently supported by AutoTuner?

WHat are the current supported search algorithms by AutoTuner?

How to set the direction of tuning in AutoTuner?

How can the tuning direction be set in AutoTuner?

How to set the direction of tuning in AutoTuner?

What environment is required for AutoTuner?

What environment is necessary for operating AutoTuner?

What environment is required for AutoTuner?

Which parameters/variables can be used for tune or sweep?

What parameters or variables can be tuned or swept in AutoTuner?

Which parameters/variables can be used for tune or sweep?

How to add verilog designs to ORFS repository for a full RTL-GDS flow execution?

How can Verilog designs be added to the ORFS repository for complete RTL-GDSII flow execution?

How to add verilog designs to ORFS repository for a full RTL-GDS flow execution?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export PLATFORM?

For a Single-port memory design using the gf180 platform, what should be the value for the parameter, export PLATFORM?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export PLATFORM?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

For a Single-port memory design using the gf180 platform, what should be the value for the parameter, export DESIGN_NAME?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

For a Single-port memory design using the gf180 platform, what should be the value for the parameter, export VERILOG_FILES?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export SDC_FILE?

For a Single-port memory design using the gf180 platform, what should be the value for the parameter, export SDC_FILE?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export SDC_FILE?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

For a Single-port memory design using the gf180 platform, what should be the value for the parameter, export CORE_UTILIZATION?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

For a Single-port memory design using the gf180 platform, what should be the value for the parameter, export PLACE_DENSITY?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

For a Single-port memory design using the gf180 platform, what should be the value for the parameter, export TNS_END_PERCENT?

While designing spm that implements a Single-port memory using gf180 platform, what can be the value of the following parameter, export

What is the function of the Environment Variables for the OpenROAD Flow Scripts?

How do Environment Variables influence the OpenROAD Flow Scripts?

What is the function of the Environment Variables for the OpenROAD Flow Scripts?

What does the general variables for all stages, SKIP_REPORT_METRICS do?

What is the role of the SKIP_REPORT_METRICS variable in all stages?

What does the general variables for all stages, SKIP_REPORT_METRICS do?

Can you explain the usage of the Library Setup variable, PROCESS?

Can you detail the PROCESS variable in Library Setup?

Can you explain the usage of the Library Setup variable, PROCESS?

What is the function of the Library Setup variable, CORNER?

What is the purpose of the CORNER variable in Library Setup?

What is the function of the Library Setup variable, CORNER?

What is the description of Library Setup variable, TECH_LEF

What does the TECH_LEF variable in Library Setup describe?

What is the description of Library Setup variable, TECH_LEF

What is the description of Library Setup variable, SC_LEF

What does the SC_LEF variable in Library Setup denote?

What is the description of Library Setup variable, SC_LEF

What is the function of the Library Setup variable, GDS_FILES

What is the role of the GDS_FILES variable in Library Setup?

What is the function of the Library Setup variable, GDS_FILES

What is the description of Library Setup variable, LIB_FILES

What does the LIB_FILES variable in Library Setup specify?

What is the description of Library Setup variable, LIB_FILES

Can you explain the usage of the Library Setup variable, DONT_USE_CELLS?

How is the DONT_USE_CELLS variable used in Library Setup?

Can you explain the usage of the Library Setup variable, DONT_USE_CELLS?

What does the following Synthesis variable, SYNTH_HIERARCHICAL do?

What does the SYNTH_HIERARCHICAL variable in Synthesis control?

What does the following Synthesis variable, SYNTH_HIERARCHICAL do?

What does the following Synthesis variable, LATCH_MAP_FILE do?

What is the purpose of the LATCH_MAP_FILE variable in Synthesis?
What does the following Synthesis variable, LATCH_MAP_FILE do?
What does the following Synthesis variable, CLKGATE_MAP_FILE do?
How does the CLKGATE_MAP_FILE variable in Synthesis function?
What does the following Synthesis variable, CLKGATE_MAP_FILE do?
Can you explain the usage of this Synthesis variable, ADDER_MAP_FILE?
What does the ADDER_MAP_FILE variable in Synthesis specify?
Can you explain the usage of this Synthesis variable, ADDER_MAP_FILE?

What does the following Synthesis variable, TIEHI_CELL_AND_PORT do?

What does the TIEHI_CELL_AND_PORT variable in Synthesis define?
What does the following Synthesis variable, TIEHI_CELL_AND_PORT do?
What does the following Synthesis variable, TIELO_CELL_AND_PORT do?
indicate?
What does the following Synthesis variable, TIELO_CELL_AND_PORT do?
What does the following Synthesis variable, MIN_BUF_CELL_AND_PORTS do?
How does the MIN_BUF_CELL_AND_PORTS variable in Synthesis operate?
What does the following Synthesis variable, MIN_BUF_CELL_AND_PORTS do?

What does the following Synthesis variable, ABC_CLOCK_PERIOD_IN_PS do?
What does the ABC_CLOCK_PERIOD_IN_PS variable in Synthesis determine?
What does the following Synthesis variable, ABC_CLOCK_PERIOD_IN_PS do?
What does the following Synthesis variable, ABC_DRIVER_CELL do?
What does the ABC_DRIVER_CELL variable in Synthesis specify?
What does the following Synthesis variable, ABC_DRIVER_CELL do?
What does the following Synthesis variable, ABC_LOAD_IN_FF do?
What does the ABC_LOAD_IN_FF variable in Synthesis imply?
What does the following Synthesis variable, ABC_LOAD_IN_FF do?
What does the following Synthesis variable, MAX_UNGROUP_SIZE do?
What does the MAX_UNGROUP_SIZE variable in Synthesis control?
What does the following Synthesis variable, MAX_UNGROUP_SIZE do?
Tell me about the Floorplan variable, FLOORPLAN_DEF?
Can you explain the FLOORPLAN_DEF variable in Floorplan?
Tell me about the Floorplan variable, FLOORPLAN_DEF?
Elaborate on the Floorplan variable, PLACE_SITE?
What is the role of the PLACE_SITE variable in Floorplan?
Elaborate on the Floorplan variable, PLACE_SITE?
Describe the use this Floorplan variable, TAPCELL_TCL?
How does the TAPCELL_TCL variable in Floorplan function?
Describe the use this Floorplan variable, TAPCELL_TCL?
What is the function of the Floorplan variable, RTLMP_FLOW?
What does the RTLMP_FLOW variable in Floorplan control?
What is the function of the Floorplan variable, RTLMP_FLOW?
Tell me about the Floorplan variable, MACRO_HALO?
Can you describe the MACRO_HALO variable in Floorplan?
Tell me about the Floorplan variable, MACRO_HALO?

Tell me about the Floorplan variable, MACRO_PLACEMENT?

What does the MACRO_PLACEMENT variable in Floorplan specify?

Tell me about the Floorplan variable, MACRO_PLACEMENT?

Elaborate on the Floorplan variable, MACRO_PLACEMENT_TCL?

operate?

Elaborate on the Floorplan variable, MACRO_PLACEMENT_TCL?

Describe the Floorplan variable, MACRO_PLACE_HALO?

What does the MACRO_PLACE_HALO variable in Floorplan indicate?

Describe the Floorplan variable, MACRO_PLACE_HALO?

Tell me about the Floorplan variable, MACRO_PLACE_CHANNEL?

What does the MACRO_PLACE_CHANNEL variable in Floorplan define?

Tell me about the Floorplan variable, MACRO_PLACE_CHANNEL?

Give me details on the Floorplan variable, MACRO_BLOCKAGE_HALO?

Can you detail the MACRO_BLOCKAGE_HALO variable in Floorplan?

Give me details on the Floorplan variable, MACRO_BLOCKAGE_HALO?

Inform me about the Floorplan variable, PDN_TCL?

What is the purpose of the PDN_TCL variable in Floorplan?

Inform me about the Floorplan variable, PDN_TCL?

Tell me about the Floorplan variable, MAKE_TRACKS.

How does the MAKE_TRACKS variable in Floorplan function?

Tell me about the Floorplan variable, MAKE_TRACKS.

What is the function of the Floorplan variable, IO_PLACER_H?

What does the IO_PLACER_H variable in Floorplan specify?

What is the function of the Floorplan variable, IO_PLACER_H?

Inform me about the Floorplan variable, IO_PLACER_V?

How does the IO_PLACER_V variable in Floorplan operate?

Inform me about the Floorplan variable, IO_PLACER_V?

Tell me about the Floorplan variable, GUI_NO_TIMING?

What does the GUI_NO_TIMING variable in Floorplan control?

Tell me about the Floorplan variable, GUI_NO_TIMING?

Give me a description for the following ‘Placement’ tool variable, HAS_IO_CONSTRAINTS?

Can you describe the HAS_IO_CONSTRAINTS variable in the Placement tool?

Give me a description for the following ‘Placement’ tool variable, HAS_IO_CONSTRAINTS?

Tell me about the ‘Placement’ tool variable, CELL_PAD_IN_SITES_GLOBAL_PLACEMENT?

What does the CELL_PAD_IN_SITES_GLOBAL_PLACEMENT variable in Placement do?

Tell me about the 'Placement' tool variable,
CELL_PAD_IN_SITES_GLOBAL_PLACEMENT?
What does this 'Placement' tool variable,
CELL_PAD_IN_SITES_DETAIL_PLACEMENT do?
How does the CELL_PAD_IN_SITES_DETAIL_PLACEMENT variable in Placement function?
What does this 'Placement' tool variable,
CELL_PAD_IN_SITES_DETAIL_PLACEMENT do?
Give me a description for the following 'Placement' tool variable,
PLACE_DENSITY?

What does the PLACE_DENSITY variable in Placement control?
Give me a description for the following 'Placement' tool variable,
What does this 'Placement' tool variable, PLACE_DENSITY_LB_ADDON do?
How does the PLACE_DENSITY_LB_ADDON variable in Placement operate?
What does this 'Placement' tool variable, PLACE_DENSITY_LB_ADDON do?
State the function of this OpenROAD 'Placement' tool variable,
REPAIR_PDN_VIA_LAYER?
specify?
State the function of this OpenROAD 'Placement' tool variable,
REPAIR_PDN_VIA_LAYER?
Give me a description for the following OpenROAD 'Placement' tool variable,
GLOBAL_PLACEMENT_ARGS?
Can you explain the GLOBAL_PLACEMENT_ARGS variable in Placement?
Give me a description for the following OpenROAD 'Placement' tool variable,
GLOBAL_PLACEMENT_ARGS?
What does this OpenROAD 'Placement' tool variable, ENABLE_DPO do?
What does the ENABLE_DPO variable in Placement do?
What does this OpenROAD 'Placement' tool variable, ENABLE_DPO do?
Give me a description for the following OpenROAD 'Placement' tool variable,
DPO_MAX_DISPLACEMENT?
How does the DPO_MAX_DISPLACEMENT variable in Placement function?
Give me a description for the following OpenROAD 'Placement' tool variable,
DPO_MAX_DISPLACEMENT?

Give me a description for the following OpenROAD 'Placement' tool variable,
GPL_TIMING_DRIVEN?

What does the GPL_TIMING_DRIVEN variable in Placement control?
Give me a description for the following OpenROAD 'Placement' tool variable,
GPL_TIMING_DRIVEN?

Give me a description for the following OpenROAD ‘Placement’ tool variable, GPL_ROUTABILITY_DRIVEN?

How does the GPL_ROUTABILITY_DRIVEN variable in Placement operate?

Give me a description for the following OpenROAD ‘Placement’ tool variable, GPL_ROUTABILITY_DRIVEN?

Give me a description for the following OpenROAD ‘Placement’ tool variable, CAP_MARGIN?

What does the CAP_MARGIN variable in Placement specify?

Give me a description for the following OpenROAD ‘Placement’ tool variable, CAP_MARGIN?

Give me a description for the following OpenROAD ‘Placement’ tool variable, SLEW_MARGIN?

How does the SLEW_MARGIN variable in Placement function?

Give me a description for the following OpenROAD ‘Placement’ tool variable, SLEW_MARGIN?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, CTS_ARGS?

What does the CTS_ARGS variable in Clock Tree Synthesis control?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, CTS_ARGS?

What does CTS_BUF_CELL do in ORFS?

What is the purpose of the CTS_BUF_CELL variable in ORFS?

What does CTS_BUF_CELL do in ORFS?

What is the use of the ORFS variable FILL_CELLS?

How does the FILL_CELLS variable in ORFS function?

What is the use of the ORFS variable FILL_CELLS?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, HOLD_SLACK_MARGIN?

What does the HOLD_SLACK_MARGIN variable in Clock Tree Synthesis specify?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, HOLD_SLACK_MARGIN?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, SETUP_SLACK_MARGIN?

How does the SETUP_SLACK_MARGIN variable in Clock Tree Synthesis operate?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, SETUP_SLACK_MARGIN?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, SKIP_GATE_CLONING?

What does the SKIP_GATE_CLONING variable in Clock Tree Synthesis control?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, SKIP_GATE_CLONING?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, SKIP_PIN_SWAP?

What does the SKIP_PIN_SWAP variable in Clock Tree Synthesis indicate?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, SKIP_PIN_SWAP?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, TNS_END_PERCENT?

How does the TNS_END_PERCENT variable in Clock Tree Synthesis function?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, TNS_END_PERCENT?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, EQUIVALENCE_CHECK?

What does the EQUIVALENCE_CHECK variable in Clock Tree Synthesis specify?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, EQUIVALENCE_CHECK?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, REMOVE_CELLS_FOR_EQY?

How does the REMOVE_CELLS_FOR_EQY variable in Clock Tree Synthesis operate?

What is the use of the following OpenROAD Clock Tree Synthesis (CTS) variable, REMOVE_CELLS_FOR_EQY?

What does the create_power_domain command do in upf?

What does the create_power_domain command achieve in upf?

What does the create_power_domain command do in upf?

What does the create_logic_port command do in upf?

What function does the create_logic_port command serve in upf?

What does the create_logic_port command do in upf?

What does restructuring do?

How does restructuring contribute to design optimization?

What does restructuring do?

What is the difference between Yosys and OpenROAD?

Can you contrast Yosys with OpenROAD in terms of functionality?

What is the difference between Yosys and OpenROAD?

Is there a way of knowing which instances were modified by resize command ?

Is it possible to identify instances modified by the resize command?

Is there a way of knowing which instances were modified by resize command ?

How can I improve runtime?

How might I enhance the runtime efficiency of my design?

How can I improve runtime?

Why does my design take so long?

What factors contribute to prolonged design processing times?

Why does my design take so long?

What is DBU?

Can you explain the concept of DBU in OpenROAD?

What is DBU?

How do I convert from DBU?

How is conversion from DBU performed?

How do I convert from DBU?

How to access GUI

How can the GUI be accessed in OpenROAD?

How to access GUI

Is there a simple way to open the GUI in ORFS?

Is there a straightforward method to launch the GUI in ORFS?

Is there a simple way to open the GUI in ORFS?

Which commands are essential for manipulating design data in OpenROAD?

What do these Debug output mean?

```
[INFO MPL-0024] [Multilevel Autoclustering] Creating clustered netlist.
[INFO MPL-0039] [Coarse Shaping] Determining shape functions for clusters.
[INFO MPL-0028] [Hierarchical Macro Placement] Placing clusters and macros.
[INFO MPL-0037] Updated location of 95 macros
Delete buffers for RTLMP flow...
[INFO RSZ-0026] Removed 0 buffers.
```

Answers

The PDNGEN module (pdn) in OpenROAD aims to simplify the process of adding a power grid into a floorplan. A user can specify a small set of power grid policies to be applied to the design, such as the layers to use, stripe width, and spacing, and then pdn will generate the actual metal straps. Power grid policies can be defined over both the standard cell area and areas occupied by macros.

The PDNGEN module (pdn) in OpenROAD aims to simplify the process of adding a power grid into a floorplan. A user can specify a small set of power grid policies to be applied to the design, such as the layers to use, stripe width, and spacing, and then pdn will generate the actual metal straps. Power grid policies can be defined over both the standard cell area and areas occupied by macros.

OpenROAD's Power Distribution Network Generator (PDNGEN) module, abbreviated as pdn, streamlines the integration of a power grid into a design's floorplan. Users can establish a few power grid guidelines, such as selecting layers, setting stripe width, and spacing, after which pdn automatically constructs the metal straps. These guidelines can be applied across both the standard cell region and macro-occupied areas.

During power switch insertion, the -switch_cell argument is used to specify the name of a coarse-grain power switch cell that is to be inserted wherever the standard cell rail connects to the rest of the power grid. The mesh layers are associated with the unswitched power net of the voltage domain, whereas the standard cell rail is associated with the switched power net of the voltage domain. The placement of a power switch cell connects the unswitched power mesh to the switched power rail through a power switch defined by the define_power_switch_cell command.

During power switch insertion, the -switch_cell argument is used to specify the name of a coarse-grain power switch cell that is to be inserted wherever the standard cell rail connects to the rest of the power grid. The mesh layers are associated with the unswitched power net of the voltage domain, whereas the standard cell rail is associated with the switched power net of the voltage domain. The placement of a power switch cell connects the unswitched power mesh to the switched power rail through a power switch defined by the define_power_switch_cell command.

For inserting power switches, the -switch_cell parameter is utilized to denote a coarse-grain power switch cell's name for insertion at points where the standard cell rail merges with the broader power grid. The mesh layers correspond with the voltage domain's unswitched power net, and the standard cell rail links to the switched power net. Inserting a power switch cell bridges the unswitched power mesh and the switched power rail, facilitated by the define_power_switch_cell command.

The partitioning module (par) is based on TritonPart, an open-source constraints-driven partitioner. par can be used to partition a hypergraph or a gate-level netlist. TritonPart solves multi-way partitioning by 1) constraints-driven coarsening, 2) initial partitioning, 3) refinement, 4) cut-overlay clustering and partitioning (COCP), and 5) V-cycle refinement. The purpose of partitioning a netlist is to reduce the complexity of a circuit so that algorithms can operate on smaller chunks of the netlist at a time. Partitioning a netlist may slightly reduce the optimization opportunities compared to a non-partitioned approach, but it greatly benefits the runtime. Partitioning a netlist may be the only option to feasibly operate on very large circuits.

The partitioning module (par) is based on TritonPart, an open-source constraints-driven partitioner. par can be used to partition a hypergraph or a gate-level netlist. TritonPart solves multi-way partitioning by 1) constraints-driven coarsening, 2) initial partitioning, 3) refinement, 4) cut-overlay clustering and partitioning (COCP), and 5) V-cycle refinement. The purpose of partitioning a netlist is to reduce the complexity of a circuit so that algorithms can operate on smaller chunks of the netlist at a time. Partitioning a netlist may slightly reduce the optimization opportunities compared to a non-partitioned approach, but it greatly benefits the runtime. Partitioning a netlist may be the only option to feasibly operate on very large circuits.

The partitioning tool, par, leverages TritonPart, a freely available constraint-based partitioning software. It enables the partitioning of hypergraphs or gate-level netlists through a process involving constraint-led coarsening, initial partitioning, refinement, cut-overlay clustering and partitioning (COCP), and V-cycle refinement. The main goal is to simplify the circuit's complexity, allowing algorithms to manage smaller sections of the netlist efficiently. Though partitioning may slightly limit optimization chances, it significantly enhances runtime and is often the sole strategy for managing vast circuits.

The Detailed Routing (drt) module in OpenROAD is based on the open-source detailed router TritonRoute. TritonRoute consists of several main building blocks, including pin access analysis, track assignment, initial detailed routing, search and repair, and a DRC engine. The initial development of the router is inspired by the ISPD-2018 initial detailed routing contest. However, the current drt framework differs and is built from scratch, aiming for an industrial-oriented, scalable, and flexible flow.

The Detailed Routing (drt) module in OpenROAD is based on the open-source detailed router TritonRoute. TritonRoute consists of several main building blocks, including pin access analysis, track assignment, initial detailed routing, search and repair, and a DRC engine. The initial development of the router is inspired by the ISPD-2018 initial detailed routing contest. However, the current drt framework differs and is built from scratch, aiming for an industrial-oriented, scalable, and flexible flow.

The Detailed Routing (drt) module of OpenROAD, founded on the open-source detailed router TritonRoute, incorporates key components like pin access analysis, track assignment, initial routing, search and repair, and a DRC (Design Rule Checking) engine. Initially inspired by the ISPD-2018 detailed routing challenge, the current drt architecture is entirely new, designed for industrial-grade scalability and flexibility.

The structure of OpenDB is based on the text file formats LEF (library) and DEF (design) formats version 5.6. OpenDB supports a binary file format to save and load the design much faster than using LEF and DEF. OpenDB is written in C++ 98 with standard library-style iterators. The classes are designed to be fast enough to base an application on without having to copy them into application-specific structures.

The structure of OpenDB is based on the text file formats LEF (library) and DEF (design) formats version 5.6. OpenDB supports a binary file format to save and load the design much faster than using LEF and DEF. OpenDB is written in C++ 98 with standard library-style iterators. The classes are designed to be fast enough to base an application on without having to copy them into application-specific structures.

OpenDB's architecture mirrors the LEF (Library Exchange Format) and DEF (Design Exchange Format) text file standards version 5.6, supporting a binary file format for quicker design saving and loading than possible with LEF and DEF. Written in C++ 98, OpenDB employs standard library-style iterators and classes designed for rapid operation, obviating the need for copying into specific application structures.

FastRoute innovates over prior routing frameworks because it integrates several novel techniques: fast congestion-driven via-aware Steiner tree construction, 3-bend routing, virtual capacity adjustment, multi-source multi-sink maze routing, and spiral layer assignment. These techniques not only address the routing congestion measured at the edges of global routing grids but also minimize the total wirelength and via usage, which is critical for subsequent detailed routing, yield, and manufacturability. Experimental results show that FastRoute is highly effective and efficient in solving ISPD07 and ISPD08 global routing benchmark suites. The results outperform recently published academic global routers in both routability and runtime. In particular, for ISPD07 and ISPD08 global routing benchmarks, FastRoute generates 12 congestion-free solutions out of 16 benchmarks with a speed significantly faster than other routers.

FastRoute innovates over prior routing frameworks because it integrates several novel techniques: fast congestion-driven via-aware Steiner tree construction, 3-bend routing, virtual capacity adjustment, multi-source multi-sink maze routing, and spiral layer assignment. These techniques not only address the routing congestion measured at the edges of global routing grids but also minimize the total wirelength and via usage, which is critical for subsequent detailed routing, yield, and manufacturability. Experimental results show that FastRoute is highly effective and efficient in solving ISPD07 and ISPD08 global routing benchmark suites. The results outperform recently published academic global routers in both routability and runtime. In particular, for ISPD07 and ISPD08 global routing benchmarks, FastRoute generates 12 congestion-free solutions out of 16 benchmarks with a speed significantly faster than other routers.

FastRoute distinguishes itself from previous routing frameworks by integrating innovative approaches such as fast congestion-driven via-aware Steiner tree construction, 3-bend routing, virtual capacity adjustment, multi-source multi-sink maze routing, and spiral layer assignment. These methods tackle routing congestion at global routing grid edges and aim to minimize overall wirelength and via count, vital for detailed routing, yield, and manufacturability. According to tests, FastRoute excels in efficiency and effectiveness across ISPD07 and ISPD08 global routing benchmarks, outperforming contemporary academic routers in routability and speed, delivering congestion-free outcomes for 12 out of 16 benchmarks.

A macro placer (MP) is an algorithm which specifically focuses on placing macros in the core area. RTL-MP is a novel macro placer that utilizes RTL information and tries to “mimic” the interaction between the frontend RTL designer and the back- end physical design engineer to produce human--quality floorplans. By exploiting the logical hierarchy and processing logical modules based on connection signatures, RTL-MP can capture the dataflow inherent in the RTL and use the dataflow information to guide macro placement.

A macro placer (MP) is an algorithm which specifically focuses on placing macros in the core area. RTL-MP is a novel macro placer that utilizes RTL information and tries to “mimic” the interaction between the frontend RTL designer and the back- end physical design engineer to produce human--quality floorplans. By exploiting the logical hierarchy and processing logical modules based on connection signatures, RTL-MP can capture the dataflow inherent in the RTL and use the dataflow information to guide macro placement.

A macro placer (MP) specifically targets macro placement within the core area. The novel RTL-MP leverages RTL data to emulate the collaborative process between frontend RTL designers and backend physical design engineers, producing floorplans of comparable quality to human-generated ones. By analyzing logical hierarchies and connections, RTL-MP captures the RTL's inherent data flow to inform its macro placement

Recently, with the increasing complexity of IP blocks, and in particular with auto-generated RTL for machine learning (ML) accelerators, the number of macros in a single RTL block can easily run into several hundred. This makes the task of generating an automatic floorplan (.def) with IO pin and macro placements for front-end physical synthesis even more critical and challenging. The so-called peripheral approach of forcing macros to the periphery of the layout is no longer viable when the ratio of the sum of the macro perimeters to the floorplan perimeter is large since this increases the required stacking depth of macros. Thus, a novel multilevel physical planning approach that exploits the hierarchy and dataflow inherent in the design RTL, and describes its realization in a new hierarchical macro placer, Hier-RTLMP was developed.

Recently, with the increasing complexity of IP blocks, and in particular with auto-generated RTL for machine learning (ML) accelerators, the number of macros in a single RTL block can easily run into several hundred. This makes the task of generating an automatic floorplan (.def) with IO pin and macro placements for front-end physical synthesis even more critical and challenging. The so-called peripheral approach of forcing macros to the periphery of the layout is no longer viable when the ratio of the sum of the macro perimeters to the floorplan perimeter is large since this increases the required stacking depth of macros. Thus, a novel multilevel physical planning approach that exploits the hierarchy and dataflow inherent in the design RTL, and describes its realization in a new hierarchical macro placer, Hier-RTLMP was developed.

The increasing complexity of IP blocks, especially auto-generated RTL for machine learning (ML) accelerators, has led to designs with hundreds of macros. This escalates the difficulty of automatically generating a floorplan (.def) that includes IO pin and macro placements for front-end physical synthesis. The traditional peripheral placement strategy becomes impractical when the macro perimeter sum greatly exceeds the floorplan's perimeter, necessitating a novel multilevel physical planning approach. This approach, realized in the Hier-RTLMP, leverages design RTL's inherent hierarchy and data flow for effective macro placement.

Hier-RTLMP consists of an auto-clustering engine that transforms the logical hierarchy into a physical hierarchy. Unlike RTL-MP where the physical hierarchy is a single level, Hier-RTLMP's autoclustering engine creates a multilevel physical hierarchy of physical clusters. This enables the handling of large RTLs with hundreds of macros and allows for the placement of macros within the core area.

Hier-RTLMP consists of an auto-clustering engine that transforms the logical hierarchy into a physical hierarchy. Unlike RTL-MP where the physical hierarchy is a single level, Hier-RTLMP's autoclustering engine creates a multilevel physical hierarchy of physical clusters. This enables the handling of large RTLs with hundreds of macros and allows for the placement of macros within the core area.

Hier-RTLMP introduces an auto-clustering engine that converts logical hierarchies into a multilevel physical structure. Unlike the single-level physical hierarchy in RTL-MP, Hier-RTLMP's engine facilitates managing extensive RTL designs with numerous macros, allowing for their strategic placement within the core area.

The first part of the FastRoute framework is topology generation. Because FastRoute tries to avoid rip-up and reroute to reduce both wirelength and runtime, the initial tree topology has significant impacts. With the congestion-driven and via-aware topology, the next stage of FastRoute is actual routing. We find that there exists significant potential to improve traditional routing techniques in terms of via control and congestion reduction. The most commonly used routing techniques in global routing include L/Z/U pattern routing, monotonic routing, and maze routing. In addition to new topology and routing techniques, Fast Route integrates several performance enhancement techniques to further improve routing quality and reduce run

The first part of the FastRoute framework is topology generation. Because FastRoute tries to avoid rip-up and reroute to reduce both wirelength and runtime, the initial tree topology has significant impacts. With the congestion-driven and via-aware topology, the next stage of FastRoute is actual routing. We find that there exists significant potential to improve traditional routing techniques in terms of via control and congestion reduction. The most commonly used routing techniques in global routing include L/Z/U pattern routing, monotonic routing, and maze routing. In addition to new topology and routing techniques, Fast Route integrates several performance enhancement techniques to further improve routing quality and reduce run

FastRoute's framework begins with topology generation, aiming to minimize rip-up and reroute actions to reduce wirelength and runtime. The initial tree topology, based on congestion-driven and via-aware principles, sets the stage for the routing phase, which shows significant improvement opportunities over traditional methods in terms of via management and congestion alleviation. Besides innovative topology and routing strategies, FastRoute incorporates various performance-boosting techniques to enhance routing quality and

Traditionally, global routing just uses tree structures like RMST or RSMT while RSMT is becoming more popular due to its minimal wirelength to connect a multi-pin net.

Traditionally, global routing just uses tree structures like RMST or RSMT while RSMT is becoming more popular due to its minimal wirelength to connect a multi-pin net.

Global routing traditionally utilizes tree structures such as RMST or RSMT, with RSMT growing in preference due to its efficiency in connecting multi-pin nets with minimal wirelength.

The OpenROAD GUI is made from custom Qt code and does not apply to the console. OpenROAD relies on the tclreadline package, which doesn't work well with imported namespaces.

The OpenROAD GUI is made from custom Qt code and does not apply to the console. OpenROAD relies on the tclreadline package, which doesn't work well with imported namespaces.

The OpenROAD GUI is built using custom Qt programming and is not applicable to command-line operations. It utilizes the tclreadline package, which faces compatibility issues with imported namespaces.

Messages with the MPL prefix are from the macro placement (mpl) module. These messages are progress messages, informing the user which step of the HierRTLMP flow is being executed.

In global_placement, -pad_right and -pad_left add padding which is measured in cell sites to effectively make the cell appear wider. pad_left adds a pad to the left side, pad_right adds it to the right side. Wider cells during global placement can help reduce placement congestion and make finding a valid placement easier.

In global_placement, -pad_right and -pad_left add padding which is measured in cell sites to effectively make the cell appear wider. pad_left adds a pad to the left side, pad_right adds it to the right side. Wider cells during global placement can help reduce placement congestion and make finding a valid placement easier.

In the global placement process, the options -pad_right and -pad_left introduce padding in terms of cell sites, effectively broadening the cell. This padding on either side aids in alleviating placement congestion and facilitates easier placement validation.

Cell padding only impacts the way the global placer (gpl) and detailed placer (dpl) sees the cells. Cell padding does not change the logic area of the cell from the user perspective. If you were to use the report_design_area command, you would see the unpadded design utilization reported.

Cell padding only impacts the way the global placer (gpl) and detailed placer (dpl) sees the cells. Cell padding does not change the logic area of the cell from the user perspective. If you were to use the report_design_area command, you would see the unpadded design utilization reported.

Padding adjustments affect only how global and detailed placers view the cells, leaving the cell's logical area as perceived by the user unchanged. Utilizing the report_design_area command will display the design's utilization without the padding.

Cell padding can reduce the maximum possible design utilization. If you don't want padding in your design, you can reduce the default padding or turn it off. However, padding is usually necessary to achieve a cleanly routed design, because cells which are too close together may make routing impossible. If you want to optimize for very high-density designs, I believe you can also set padding on a master-by-master basis so that only the most difficult cells are padded. This process may take some trial and error.

Cell padding can reduce the maximum possible design utilization. If you don't want padding in your design, you can reduce the default padding or turn it off. However, padding is usually necessary to achieve a cleanly routed design, because cells which are too close together may make routing impossible. If you want to optimize for very high-density designs, I believe you can also set padding on a master-by-master basis so that only the most difficult cells are padded. This process may take some trial and error.

Implementing cell padding may limit the maximum design utilization achievable. Adjusting or disabling default padding is an option, though padding is often crucial for achieving a neatly routed design by preventing overly congested cell placements. For high-density design optimization, padding can be selectively applied to particularly challenging cells, requiring some experimentation.

If you are using ORFS, the CORE_UTILIZATION variable only determines the core utilization after Yosys (synthesis) runs using the netlist data. The utilization can change dramatically as OpenROAD performs timing optimization, which will insert buffers, remove buffers, and resize cells. For example, if you set CORE_UTILIZATION to 70% utilization during the initialization step, you may end up with 90%+ utilization because of optimization to meet timing constraints. It is currently up to the designer to ensure that the CORE_UTILIZATION is set appropriately, which may require some trial and error.

If you are using ORFS, the CORE_UTILIZATION variable only determines the core utilization after Yosys (synthesis) runs using the netlist data. The utilization can change dramatically as OpenROAD performs timing optimization, which will insert buffers, remove buffers, and resize cells. For example, if you set CORE_UTILIZATION to 70% utilization during the initialization step, you may end up with 90%+ utilization because of optimization to meet timing constraints. It is currently up to the designer to ensure that the CORE_UTILIZATION is set appropriately, which may require some trial and error.

With ORFS, the CORE_UTILIZATION variable reflects core utilization post-Yosys synthesis, based on netlist data. Subsequent timing optimization in OpenROAD, which includes buffer insertion, deletion, and cell resizing, can significantly alter utilization levels. Designers must judiciously set CORE_UTILIZATION, anticipating possible adjustments through trial and error.

Relaxing timing constraints can usually reduce your design utilization because fewer buffers will need to be inserted. However, detailed placement can fail for several reasons, including 1) setting core utilization too high, 2) not having enough cell padding during global placement, 3) having too much cell padding during detailed placement, 4) having a poor macro placement which will block placement of some standard cells

Relaxing timing constraints can usually reduce your design utilization because fewer buffers will need to be inserted. However, detailed placement can fail for several reasons, including 1) setting core utilization too high, 2) not having enough cell padding during global placement, 3) having too much cell padding during detailed placement, 4) having a poor macro placement which will block placement of some standard cells

Easing timing constraints can lead to lower design utilization by reducing the need for buffer insertion. However, detailed placement might fail due to reasons like excessive core utilization, insufficient global placement padding, excessive detailed placement padding, or suboptimal macro placement impeding

Empiracally, the OpenROAD team has found that OpenROAD optimizes better when the buffers from Yosys are removed, because Yosys does not do physical synthesis.

Empiracally, the OpenROAD team has found that OpenROAD optimizes better when the buffers from Yosys are removed, because Yosys does not do physical synthesis.

The OpenROAD team has empirically observed that removing Yosys-inserted buffers leads to better optimization outcomes, as Yosys lacks physical synthesis capabilities.

Empirically, applying timing constraints to Yosys for synthesis does not have much impact on the output netlist. However, the results may not be identical if the timing constraints are changed.

Empirically, applying timing constraints to Yosys for synthesis does not have much impact on the output netlist. However, the results may not be identical if the timing constraints are changed.

Applying timing constraints during Yosys synthesis shows limited influence on the netlist outcome, with possible variations if timing constraints are altered.

This warning likely occurs because of multiple block abstracts with the same via. To fix this problem, one should ensure that the block abstracts do not have duplicates via definitions.

I would advise against using low-level OBD APIs to interact with OpenROAD. User-facing APIs in OpenROAD have documentation and error checking to prevent improper usage, however low-level APIs may not have error checking for performance reasons. In this case, it appears the db you are creating does not have a logger and can't issue any messages.

I would advise against using low-level OBD APIs to interact with OpenROAD. User-facing APIs in OpenROAD have documentation and error checking to prevent improper usage, however low-level APIs may not have error checking for performance reasons. In this case, it appears the db you are creating does not have a logger and can't issue any messages.

I recommend against using the low-level OBD APIs in OpenROAD since user-friendly APIs offer documentation and error checking to prevent misuse, unlike low-level APIs that might lack error checks for efficiency. In your situation, the database being created lacks a logger, preventing message issuance.

The IR Drop Analysis module in OpenROAD (psm) is based on PDNSim, an open-source static IR analyzer. PDNSim will perform a simulation based on the estimated current draw of each instance to estimate what is the voltage drop seen at each cell. That is, it shows the voltage drop between the voltage source(s) of the chip and each instance.

The IR Drop Analysis module in OpenROAD (psm) is based on PDNSim, an open-source static IR analyzer. PDNSim will perform a simulation based on the estimated current draw of each instance to estimate what is the voltage drop seen at each cell. That is, it shows the voltage drop between the voltage source(s) of the chip and each instance.

The IR Drop Analysis module in OpenROAD, based on PDNSim, performs simulations to estimate the voltage drop across each cell by analyzing the current draw of each instance, illustrating the drop from the chip's voltage source(s) to each instance.

The features of IR Drope analysis are: reporting the worst IR drop, reporting the worst current density over all nodes and wire segments in the power distribution network (given a placed and PDN-synthesized design), checking for floating PDN stripes on the power and ground nets, and spice netlist writer for power distribution network wire segments.

The features of IR Drope analysis are: reporting the worst IR drop, reporting the worst current density over all nodes and wire segments in the power distribution network (given a placed and PDN-synthesized design), checking for floating PDN stripes on the power and ground nets, and spice netlist writer for power distribution network wire segments.

IR Drop analysis features include identifying the most severe IR drop and current density across all nodes and wire segments of the power distribution network, detecting floating PDN stripes on power and ground nets, and generating spice netlists for PDN wire segments.

Flute3 is an open-source rectilinear Steiner minimum tree heuristic with improvements made by UFRGS students and James Cherry. This tool is used for the calculation of wirelength in grt and rsz.

Flute3 is an open-source rectilinear Steiner minimum tree heuristic with improvements made by UFRGS students and James Cherry. This tool is used for the calculation of wirelength in grt and rsz.

Flute3, enhanced by UFRGS students and James Cherry, is an open-source heuristic for calculating the rectilinear Steiner minimum tree, crucial for estimating wirelength in tools like grt and rsz.

OpenDB is a design database to support tools for physical chip design. It was originally developed by Athena Design Systems. Nefelus, Inc. acquired the rights to the code and open-sourced it with BSD-3 license in 2019 to support the DARPA OpenROAD project. The structure of OpenDB is based on the text file formats LEF (library) and DEF (design) formats version 5.6. OpenDB supports a binary file format to save and load the design much faster than using LEF and DEF. OpenDB is written in C++ 98 with standard library style iterators. The classes are designed to be fast enough to base an application on without having to copy them into

OpenDB is a design database to support tools for physical chip design. It was originally developed by Athena Design Systems. Nefelus, Inc. acquired the rights to the code and open-sourced it with BSD-3 license in 2019 to support the DARPA OpenROAD project. The structure of OpenDB is based on the text file formats LEF (library) and DEF (design) formats version 5.6. OpenDB supports a binary file format to save and load the design much faster than using LEF and DEF. OpenDB is written in C++ 98 with standard library style iterators. The classes are designed to be fast enough to base an application on without having to copy them into

OpenDB, originally developed by Athena Design Systems and later open-sourced by Nefelus, Inc. under a BSD-3 license for the DARPA OpenROAD project in 2019, serves as a database for physical chip design, utilizing LEF and DEF formats version 5.6 and supporting a binary format for efficient design loading and saving.

The automatic code generator in OpenROAD is used to generate code for OpenDB objects and Iterators. It uses JSON input and automatically generates corresponding C++ files

The automatic code generator in OpenROAD is used to generate code for OpenDB objects and Iterators. It uses JSON input and automatically generates corresponding C++ files

OpenROAD's automatic code generator produces C++ files for OpenDB objects and iterators from JSON input, streamlining code generation.

OpenROAD is run using Tcl scripts. The following commands are used to read and write design data.
read_lef [-tech] [-library] filename
read_def filename
write_def [-version 5.8|5.7|5.6|5.5|5.4|5.3] filename
read_verilog filename
write_verilog filename
read_db filename
write_db filename
write_abstract_lef filename
OpenROAD contains an abstract LEF writer that can take your current design and emit an abstract LEF representing the external pins of your design and metal obstructions. Use the write_abstract_lef command to generate the abstract LEF.
OpenROAD contains an abstract LEF writer that can take your current design and emit an abstract LEF representing the external pins of your design and metal obstructions. Use the write_abstract_lef command to generate the abstract LEF.
The abstract LEF writer in OpenROAD allows for the creation of an abstract LEF from your design, detailing external pins and metal obstructions, using the write_abstract_lef command.
the supply pins on cells. The global_connect command is used to define logical connections between supply pins on design instances and their respective supply nets. Note that the global_connect command only creates a logical connection; it does not perform any routing for a physical connection.
Global connections are typically used to define connections between a supply net (such as power/ground) and the supply pins on cells. The global_connect command is used to define logical connections between supply pins on design instances and their respective supply nets. Note that the global_connect command only creates a logical connection; it does not perform any routing for a physical connection.
Global connections are designated for linking supply nets to cell supply pins without establishing physical routes, instead creating logical connections that are made using the global_connect command.
Global connections are typically used to define connections between a supply net (such as power/ground) and the supply pins on cells. The clear_global_connect command is used to clear previously declared connections between supply pins on design instances and their respective supply nets. Note that the clear_global_connect command only clears the logical connections; it does not remove any routing or physical connections.
Global connections are typically used to define connections between a supply net (such as power/ground) and the supply pins on cells. The clear_global_connect command is used to clear previously declared connections between supply pins on design instances and their respective supply nets. Note that the clear_global_connect command only clears the logical connections; it does not remove any routing or physical connections.
The clear_global_connect command removes logical connections between supply pins and their nets without affecting any physical routing or connections.
Global connections are typically used to define connections between a supply net (such as power/ground) and the supply pins on cells. The report_global_connect command is used to print out the currently defined global connection rules.
Global connections are typically used to define connections between a supply net (such as power/ground) and the supply pins on cells. The report_global_connect command is used to print out the currently defined global connection rules.
Global connections typically link supply nets to cell supply pins, and the report_global_connect command prints the existing global connection rules.
The report_cell_usage command is used to print out the number of instances of each type of cell (master) used in the design.
The report_cell_usage command is used to print out the number of instances of each type of cell (master) used in the design.
The report_cell_usage command displays the count of each cell type used within the design.

To compute the die area for the initialize_floorplan command, OpenROAD first calculates the core area by dividing the total logic area of the instances coming from synthesis and by the specified core_utilization. OpenROAD then shapes that core area based on the aspect_ratio parameter. Finally, OpenROAD expands the core area by adding a core margin on each edge of the core area. Altogether, this forms the die area.

To compute the die area for the initialize_floorplan command, OpenROAD first calculates the core area by dividing the total logic area of the instances coming from synthesis and by the specified core_utilization. OpenROAD then shapes that core area based on the aspect_ratio parameter. Finally, OpenROAD expands the core area by adding a core margin on each edge of the core area. Altogether, this forms the die area.

To calculate die area, OpenROAD first determines the core area by dividing the total logic area by the core utilization, shapes this area using the aspect ratio, and then expands it by adding core margins, resulting in the die area.

To capture output from OpenROAD, you can use standard Unix file operations and redirections. OpenROAD outputs all messages directly to the stdout I/O stream.

To capture output from OpenROAD, you can use standard Unix file operations and redirections. OpenROAD outputs all messages directly to the stdout I/O stream.

Capturing OpenROAD output can be done using standard Unix redirections, as it sends all messages directly to the stdout stream.

OpenROAD has the theoretical ability to route as few as two layers, but it has rarely been tried by the developers due to the lack of a specialized channel router. It is expected that OpenROAD will hit some issues and have to iterate. If you try this and run into issues, please kindly file an issue on GitHub. However, if the PDK is proprietary, it will be more difficult for the OpenROAD team to diagnose and debug.

OpenROAD has the theoretical ability to route as few as two layers, but it has rarely been tried by the developers due to the lack of a specialized channel router. It is expected that OpenROAD will hit some issues and have to iterate. If you try this and run into issues, please kindly file an issue on GitHub. However, if the PDK is proprietary, it will be more difficult for the OpenROAD team to diagnose and debug.

OpenROAD theoretically can route with as few as two layers, but this is rarely tested and may encounter issues, for which feedback on GitHub is appreciated, especially for proprietary PDKs.

OpenROAD supports using multi-VT cell libraries, and it can swap between VT cells during optimization.

OpenROAD supports using multi-VT cell libraries, and it can swap between VT cells during optimization.

OpenROAD allows the use of multi-VT cell libraries, enabling swapping between VT cells during optimization phases.

OpenROAD supports multi-corner Static Timing Analysis (STA), but it doesn't currently support multi-mode STA. "MMMC" files from proprietary tools are stored in proprietary formats, which OpenROAD cannot support. The OpenSTA manual will contain more information about how to run multi-corner analysis.

OpenROAD supports multi-corner Static Timing Analysis (STA), but it doesn't currently support multi-mode STA. "MMMC" files from proprietary tools are stored in proprietary formats, which OpenROAD cannot support. The OpenSTA manual will contain more information about how to run multi-corner analysis.

While OpenROAD supports multi-corner Static Timing Analysis, it does not yet accommodate multi-mode STA due to proprietary format constraints of "MMMC" files.

The OpenROAD Flow is limited to the RTL language support that the Yosys synthesizer provides. Yosys currently provides support for a limited subset of SystemVerilog and full support for Verilog. OpenROAD only supports structural Verilog netlists.

The OpenROAD Flow is limited to the RTL language support that the Yosys synthesizer provides. Yosys currently provides support for a limited subset of SystemVerilog and full support for Verilog. OpenROAD only supports structural Verilog netlists.

OpenROAD's RTL language support is contingent on Yosys synthesizer's capabilities, with full support for Verilog and partial support for SystemVerilog.

Place pins on the boundary of the die on the track grid to minimize net wirelengths. Pin placement also creates a metal shape for each pin using min-area rules. For designs with unplaced cells, the net wirelength is computed considering the center of the die area as the unplaced cells' position.

Place pins on the boundary of the die on the track grid to minimize net wirelengths. Pin placement also creates a metal shape for each pin using min-area rules. For designs with unplaced cells, the net wirelength is computed considering the center of the die area as the unplaced cells' position.

For minimized net wirelengths, place pins on the die's boundary on the track grid, creating metal shapes for each pin according to min-area rules and considering the die center as the position for unplaced cells.

This tool checks antenna violations and generates a report to indicate violated nets. See LEF/DEF 5.8 Language Reference, Appendix C, "Calculating and Fixing Process Antenna Violations" (p.389) for a description of antenna violations.

This tool checks antenna violations and generates a report to indicate violated nets. See LEF/DEF 5.8 Language Reference, Appendix C, "Calculating and Fixing Process Antenna Violations" (p.389) for a description of antenna violations.

The tool checks for antenna violations and reports nets that violate these rules, with guidance found in the LEF/DEF 5.8 Language Reference.

Clock tree synthesis (CTS) is the step of distributing a clock to all endpoints (such as flip-flops) while trying to minimize power and skew (the different in clock arrival times between two registers). The clock tree synthesis module in OpenROAD (cts) is based on TritonCTS 2.0, and can be run using the clock_tree_synthesis command. TritonCTS 2.0 features on-the-fly characterization, and therefore does not need to pre-generate characterization data. The on-the-fly characterization feature can be optionally controlled by parameters specified by the configure_cts_characterization command. You can use the set_wire_rc command to set the clock routing layer used when calculating parasitics in this step.

Clock tree synthesis (CTS) is the step of distributing a clock to all endpoints (such as flip-flops) while trying to minimize power and skew (the different in clock arrival times between two registers). The clock tree synthesis module in OpenROAD (cts) is based on TritonCTS 2.0, and can be run using the clock_tree_synthesis command. TritonCTS 2.0 features on-the-fly characterization, and therefore does not need to pre-generate characterization data. The on-the-fly characterization feature can be optionally controlled by parameters specified by the configure_cts_characterization command. You can use the set_wire_rc command to set the clock routing layer used when calculating parasitics in this step.

Clock tree synthesis in OpenROAD, facilitated by TritonCTS 2.0, distributes clock signals to endpoints while minimizing power and skew, offering on-the-fly characterization to avoid pre-generated data needs.

The detailed placement module in OpenROAD (dpl) is based on OpenDP or Open-Source Detailed Placement Engine. Its key features are

fence region and fragmented ROWs.

The detailed placement module in OpenROAD (dpl) is based on OpenDP or Open-Source Detailed Placement Engine. Its key features are

fence region and fragmented ROWs.

The detailed placement module, based on OpenDP, focuses on fence regions and fragmented rows for placement optimization.

The restructure module in OpenROAD (rmp) is based on an interface to ABC for local resynthesis. The package allows logic restructuring that targets area or timing. It extracts a cloud of logic using the OpenSTA timing engine and passes it to ABC through the blif interface. Multiple recipes for area or timing are run to obtain multiple structures from ABC; the most desirable among these is used to improve the netlist. The ABC output is read back by a blif reader which is integrated into OpenDB. Blif writer and reader also support constants from and to OpenDB. Reading back of constants requires insertion of tie cells which should be provided by the user as per the interface described below.

The restructure module in OpenROAD (rmp) is based on an interface to ABC for local resynthesis. The package allows logic restructuring that targets area or timing. It extracts a cloud of logic using the OpenSTA timing engine and passes it to ABC through the blif interface. Multiple recipes for area or timing are run to obtain multiple structures from ABC; the most desirable among these is used to improve the netlist. The ABC output is read back by a blif reader which is integrated into OpenDB. Blif writer and reader also support constants from and to OpenDB. Reading back of constants requires insertion of tie cells which should be provided by the user as per the interface described below.
The restructure module interfaces with ABC for logic resynthesis aimed at area or timing improvement, utilizing OpenSTA for logic extraction and integrating ABC's output through OpenDB's blif reader and writer.
RePlAce is a tool for advancing solution quality and routability validation in Global Placement. Its features are analytic and nonlinear placement algorithms. it solves electrostatic force equations using Nesterov's method, verified with various commercial technologies and research enablements using OpenDB, verified deterministic solution generation with various compilers and OS, and supports Mixed-size placement mode.
RePlAce is a tool for advancing solution quality and routability validation in Global Placement. Its features are analytic and nonlinear placement algorithms. it solves electrostatic force equations using Nesterov's method, verified with various commercial technologies and research enablements using OpenDB, verified deterministic solution generation with various compilers and OS, and supports Mixed-size placement mode.
RePlAce advances global placement solution quality and routability with analytic and nonlinear algorithms, solving electrostatic force equations and ensuring deterministic solutions across various technologies.
"Hier-RTLMP" is defined as a hierarchical automatic macro placer for large-scale complex IP blocks. This tool builds on the existing RTLMP (mpl) framework, adopting a multilevel physical planning approach that exploits the hierarchy and data flow inherent in the design RTL.
"Hier-RTLMP" is defined as a hierarchical automatic macro placer for large-scale complex IP blocks. This tool builds on the existing RTLMP (mpl) framework, adopting a multilevel physical planning approach that exploits the hierarchy and data flow inherent in the design RTL.
Hier-RTLMP, building on the RTLMP framework, is a hierarchical macro placer leveraging design RTL's inherent hierarchy and data flow for complex IP blocks.
OpenSTA is a gate-level static timing verifier. As a stand-alone executable, it can be used to verify the timing of a design using standard file formats. OpenSTA uses a TCL command interpreter to read the design, specify timing constraints, and print timing reports.
OpenSTA is a gate-level static timing verifier. As a stand-alone executable, it can be used to verify the timing of a design using standard file formats. OpenSTA uses a TCL command interpreter to read the design, specify timing constraints, and print timing reports.
OpenSTA performs gate-level static timing analysis, using TCL for design reading, timing constraints specification, and timing report generation.
The Following standard file formats are supported by Parallax Static Timing Analyzer or OpenSTA: Verilog netlist, Liberty library, SDC timing constraints,SDF delay annotation, and SPEF parasitics.
The Following standard file formats are supported by Parallax Static Timing Analyzer or OpenSTA: Verilog netlist, Liberty library, SDC timing constraints,SDF delay annotation, and SPEF parasitics.
OpenSTA supports Verilog netlist, Liberty library, SDC timing constraints, SDF delay annotation, and SPEF parasitics for timing verification.
I currently recommend using `io_placer` instead of `auto_place_pins`. Here's an example:```io_placer -hor_layer 3 -ver_layer 2```Note that `io_placer` uses cell placement information to guide the I/O pin placement. If placement has not yet been run, the `-random` flag is required, which will distribute the pins
I currently recommend using `io_placer` instead of `auto_place_pins`. Here's an example:```io_placer -hor_layer 3 -ver_layer 2```Note that `io_placer` uses cell placement information to guide the I/O pin placement. If placement has not yet been run, the `-random` flag is required, which will distribute the pins

For pin placement, io_placer is recommended over auto_place_pins, using layer specifications for guided I/O pin placement or the -random flag for preliminary placement before cell placement.

This tool is an implementation of Design For Testing. New nets and logic are added to allow IC designs to be tested for errors in manufacturing. Physical imperfections can cause hard failures and variability can cause

This tool is an implementation of Design For Testing. New nets and logic are added to allow IC designs to be tested for errors in manufacturing. Physical imperfections can cause hard failures and variability can cause

This tool implements Design For Testing (DFT) by introducing new nets and logic, facilitating the testing of IC designs for manufacturing errors. It addresses physical defects and variability, which can lead to hard failures and timing errors, respectively.

A simple DFT insertion consists of the following parts: a scan_in pin where the test patterns are shifted in, a scan_out pin where the test patterns are read from, scan cells that replace flops with registers that allow for testing, one or more scan chains (shift registers created from your scan cells), a scan_enable pin to allow your design to enter and leave the test mode.

A simple DFT insertion consists of the following parts: a scan_in pin where the test patterns are shifted in, a scan_out pin where the test patterns are read from, scan cells that replace flops with registers that allow for testing, one or more scan chains (shift registers created from your scan cells), a scan_enable pin to allow your design to enter and leave the test mode.

DFT insertion typically involves a scan_in pin for inputting test patterns, a scan_out pin for outputting test patterns, scan cells that are substituted for flops, one or more scan chains forming shift registers, and a scan_enable pin for toggling test mode on and off.

This module contains functionality to read, and modify information from Unified Power Format (UPF) files.

This module contains functionality to read, and modify information from Unified Power Format (UPF) files.

This module offers tools to access and alter information within Unified Power Format (UPF) files.

Metal filling is a common process in integrated circuit design to enhance manufacturability and yield by making the density of metal shapes more uniform across the design. In OpenROAD, the Finale module (fin) inserts floating metal fill shapes to meet metal density design rules while obeying DRC constraints. The rules for generating metal fill shapes are driven by a JSON configuration file, and the schema can be found in the

Metal filling is a common process in integrated circuit design to enhance manufacturability and yield by making the density of metal shapes more uniform across the design. In OpenROAD, the Finale module (fin) inserts floating metal fill shapes to meet metal density design rules while obeying DRC constraints. The rules for generating metal fill shapes are driven by a JSON configuration file, and the schema can be found in the

Metal filling, used to improve manufacturability and yield in IC design by standardizing metal shape density, is implemented in OpenROAD's Finale module. It places non-connecting metal fill shapes in accordance with metal density rules and Design Rule Checking (DRC) constraints, guided by a JSON configuration file detailed in the OpenROAD documentation.

The chip-level connections module in OpenROAD (pad) is based on the open-source tool ICeWall. In this utility, either place an IO ring around the boundary of the chip and connect with either wirebond pads or a

The chip-level connections module in OpenROAD (pad) is based on the open-source tool ICeWall. In this utility, either place an IO ring around the boundary of the chip and connect with either wirebond pads or a

OpenROAD's chip-level connections module (pad) utilizes the open-source ICeWall tool to place an IO ring around the chip's boundary, connecting through wirebond pads or a bump array.

The parasitics extraction module in OpenROAD (rcx) is based on the open-source OpenRCX, a Parasitic Extraction (PEX, or RCX) tool that works on OpenDB design APIs. It extracts routed designs based on the LEF/DEF layout model.

OpenRCX extracts both Resistance and Capacitance for wires, based on coupling distance to the nearest wire and the track density context over and/or under the wire of interest, as well as cell abstracts. The capacitance and resistance measurements are based on equations of coupling distance interpolated on exact measurements from a calibration file, called the Extraction Rules file. The Extraction Rules file (RC technology file) is generated once for every process node and corner, using a provided utility for DEF wire pattern generation and regression modeling.

OpenRCX stores resistance, coupling capacitance, and ground (i.e., grounded) capacitance on OpenDB

The parasitics extraction module in OpenROAD (rcx) is based on the open-source OpenRCX, a Parasitic Extraction (PEX, or RCX) tool that works on OpenDB design APIs. It extracts routed designs based on the LEF/DEF layout model.

OpenRCX extracts both Resistance and Capacitance for wires, based on coupling distance to the nearest wire and the track density context over and/or under the wire of interest, as well as cell abstracts. The capacitance and resistance measurements are based on equations of coupling distance interpolated on exact measurements from a calibration file, called the Extraction Rules file. The Extraction Rules file (RC technology file) is generated once for every process node and corner, using a provided utility for DEF wire pattern generation and regression modeling.

OpenRCX stores resistance, coupling capacitance, and ground (i.e., grounded) capacitance on OpenDB

The parasitics extraction module in OpenROAD, based on OpenRCX, performs parasitic extraction on routed designs using the LEF/DEF model. It calculates wire resistance and capacitance by considering coupling distance and track density, utilizing a calibration file for accurate measurements. OpenRCX can also produce a .spef file for further analysis.

The resizer commands stop when the design area is -max_utilization util percent of the core area. util is between 0 and 100. The resizer stops and reports an error if the maximum utilization is exceeded.

The resizer commands stop when the design area is -max_utilization util percent of the core area. util is between 0 and 100. The resizer stops and reports an error if the maximum utilization is exceeded.

The resizer tool halts if the design exceeds -max_utilization percent of the core area's utilization, throwing an error if the limit is surpassed.

The macro placement module in OpenROAD (mpl) is based on TritonMacroPlacer, an open-source ParquetFP-based macro cell placer. The macro placer places macros/blocks honoring halos, channels, and cell row "snapping". Run global_placement before macro placement.

The macro placement module in OpenROAD (mpl) is based on TritonMacroPlacer, an open-source ParquetFP-based macro cell placer. The macro placer places macros/blocks honoring halos, channels, and cell row "snapping". Run global_placement before macro placement.

The macro placement module in OpenROAD, built on TritonMacroPlacer, places macros/blocks while respecting design constraints like halos, channels, and cell row alignment. It's recommended to run global_placement prior to macro placement.

The global routing module in OpenROAD (grt) is based on FastRoute, an open-source global router originally derived from Iowa State University's FastRoute4.1 algorithm. Global routing is responsible for creating routing guides for each net to simplify the job of the detailed router. The purpose of global routing is mainly to avoid overcongestion when creating the guides.

The global routing module in OpenROAD (grt) is based on FastRoute, an open-source global router originally derived from Iowa State University's FastRoute4.1 algorithm. Global routing is responsible for creating routing guides for each net to simplify the job of the detailed router. The purpose of global routing is mainly to avoid overcongestion when creating the guides.

OpenROAD's global routing module, built on the FastRoute algorithm, creates routing guides for each net to manage congestion during detailed routing, aiming to prevent overcongestion.

FastRoute is a global routing tool for VLSI back-end design. It is based on sequential rip-up and re-route (RRR) and a lot of novel techniques. FastRoute 1.0 first uses FLUTE to construct congestion-driven Steiner trees, which will later undergo the edge-shifting process to optimize tree structure to reduce congestion. It then uses pattern routing and maze routing with a logistic function-based cost function to solve the congestion problem. FastRoute 2.0 proposed monotonic routing and multi-source multi-sink maze routing techniques to enhance the capability to reduce congestion. FastRoute 3.0 introduced the virtual capacity technique to adaptively change the capacity associated with each global edge to divert wire usage from highly congested regions to less congested regions. FastRoute 4.0 proposed via-aware Steiner tree, 3-bend routing, and a delicate layer assignment algorithm to effectively reduce via count while maintaining outstanding congestion reduction capability. FastRoute 4.1 simplifies the way the virtual capacities are updated and applies a single set of tuning parameters to all benchmark circuits

FastRoute is a global routing tool for VLSI back-end design. It is based on sequential rip-up and re-route (RRR) and a lot of novel techniques. FastRoute 1.0 first uses FLUTE to construct congestion-driven Steiner trees, which will later undergo the edge-shifting process to optimize tree structure to reduce congestion. It then uses pattern routing and maze routing with a logistic function-based cost function to solve the congestion problem. FastRoute 2.0 proposed monotonic routing and multi-source multi-sink maze routing techniques to enhance the capability to reduce congestion. FastRoute 3.0 introduced the virtual capacity technique to adaptively change the capacity associated with each global edge to divert wire usage from highly congested regions to less congested regions. FastRoute 4.0 proposed via-aware Steiner tree, 3-bend routing, and a delicate layer assignment algorithm to effectively reduce via count while maintaining outstanding congestion reduction capability. FastRoute 4.1 simplifies the way the virtual capacities are updated and applies a single set of tuning parameters to all benchmark circuits

FastRoute utilizes a series of innovative techniques for global routing, including FLUTE for constructing congestion-driven Steiner trees, edge-shifting, pattern routing, and maze routing to address congestion, with further enhancements in subsequent versions to reduce congestion and via count effectively.

OpenROAD-flow-scripts (ORFS) is a fully autonomous, RTL-GDSII flow for rapid architecture and design space exploration, early prediction of QoR, and detailed physical design implementation. However, ORFS also enables manual intervention for finer user control of individual flow stages through Tcl commands and Python

OpenROAD-flow-scripts (ORFS) is a fully autonomous, RTL-GDSII flow for rapid architecture and design space exploration, early prediction of QoR, and detailed physical design implementation. However, ORFS also enables manual intervention for finer user control of individual flow stages through Tcl commands and Python

OpenROAD-flow-scripts offer a comprehensive, autonomous RTL-GDSII flow for rapid design exploration and physical design, allowing for manual intervention at stages through Tcl commands and Python APIs for greater

You can try the developer debugs to get a sense of what is going on. Put detailed_route_debug -pa -pa_markers -pin _3504_:A1 before you call detailed_route and run it in the GUI. It will stop at the pin and show you how it is trying to access it and any associated drc markers. You click the continue button to see the various steps. Hopefully, that gives you an idea as to what is going on. The other option is a support contract with precisioninno.com where we can help under NDA.

You can try the developer debugs to get a sense of what is going on. Put detailed_route_debug -pa -pa_markers -pin _3504_:A1 before you call detailed_route and run it in the GUI. It will stop at the pin and show you how it is trying to access it and any associated drc markers. You click the continue button to see the various steps. Hopefully, that gives you an idea as to what is going on. The other option is a support contract with precisioninno.com where we can help under NDA.

Developer debug options in OpenROAD can give insights into detailed routing processes, offering visual debug aids in the GUI and the option for NDA-supported assistance from precisioninno.com.

OpenROAD supports interoperability with other EDA tools through the industry standard Verilog, LEF, and DEF formats. In this case, you can import a DEF file into OpenROAD from an external tool, run RTL-MP2, and then export the DEF file. The DEF file format should be supported by nearly all EDA physical design tools.

OpenROAD supports interoperability with other EDA tools through the industry standard Verilog, LEF, and DEF formats. In this case, you can import a DEF file into OpenROAD from an external tool, run RTL-MP2, and then export the DEF file. The DEF file format should be supported by nearly all EDA physical design tools.

OpenROAD facilitates interoperability with other EDA tools through standard formats like Verilog, LEF, and DEF, allowing for seamless DEF file imports and exports to and from external tools.

In OpenROAD, you can use the check_drc command to run the design rule check (DRC). It should be noted that check_drc is not a general-purpose checker and only verifies signal routing according to routing rules from the technology LEF file. check_drc also requires routing guides in order to run, so global routing must be performed first. DRC is also commonly performed by an external tool which will check all layers, not just the metal routing layers.

In OpenROAD, you can use the check_drc command to run the design rule check (DRC). It should be noted that check_drc is not a general-purpose checker and only verifies signal routing according to routing rules from the technology LEF file. check_drc also requires routing guides in order to run, so global routing must be performed first. DRC is also commonly performed by an external tool which will check all layers, not just the metal routing layers.

The check_drc command in OpenROAD performs design rule checks based on routing rules from the technology LEF file, requiring prior global routing and typically supplemented by external tools for comprehensive DRC across all layers.

When using the read_def command with the -floorplan_initialize argument, OpenROAD will read only physical placement information such as pin locations and instance locations. It does not read or redefine the netlist.

When using the read_def command with the -floorplan_initialize argument, OpenROAD will read only physical placement information such as pin locations and instance locations. It does not read or redefine the netlist.

Using the read_def command with -floorplan_initialize in OpenROAD loads only the physical placement data, excluding netlist alterations.

These flags disable optimizations in case of trouble or unexpected results. They are mainly included as a failsafe for users, rather than something that is expected to be used.

These flags disable optimizations in case of trouble or unexpected results. They are mainly included as a failsafe for users, rather than something that is expected to be used.

Certain flags in OpenROAD disable optimizations as a safeguard against unexpected outcomes, intended as a last resort for troubleshooting.

The detailed_placement command moves instances to legal locations after global placement. While the global placer (gpl) may place cells in a roughly optimal position, gpl may not place the cells in legal locations because they may not be aligned to cell rows or sites. The detailed placer (dpl) will legalize the cell by shifting it to a nearby location which is aligned to the cell site grid. The detailed_placement command only performs basic legalization and does not attempt to optimize placement. The optimize_mirroring and improve_placement commands will perform optimization on the legalized cells.

The detailed_placement command moves instances to legal locations after global placement. While the global placer (gpl) may place cells in a roughly optimal position, gpl may not place the cells in legal locations because they may not be aligned to cell rows or sites. The detailed placer (dpl) will legalize the cell by shifting it to a nearby location which is aligned to the cell site grid. The detailed_placement command only performs basic legalization and does not attempt to optimize placement. The optimize_mirroring and improve_placement commands will perform optimization on the legalized cells.
The detailed_placement command in OpenROAD repositions instances to legal sites post-global placement, focusing on basic legalization without attempting placement optimization, which is handled by subsequent commands.
Max distance that an instance can be moved (in microns) when finding a site where it can be placed. Either set one value for both directions or set {disp_x disp_y} for individual directions. The default values are {0, 0}, and the allowed values within are integers [0, MAX_INT]. This argument is useful because it limits the amount of searching and therefore amount of runtime that the detailed placer can use.
Max distance that an instance can be moved (in microns) when finding a site where it can be placed. Either set one value for both directions or set {disp_x disp_y} for individual directions. The default values are {0, 0}, and the allowed values within are integers [0, MAX_INT]. This argument is useful because it limits the amount of searching and therefore amount of runtime that the detailed placer can use.

Specifies the maximum distance instances can be relocated during detailed placement, with default settings at {0, 0} and customizable values for x and y directions to limit search scope and runtime.
detailed_place -disallow_one_site_gaps will disallow the detailed placer from leaving gaps between cells that are exactly one cell sitewide. This feature is mainly useful for PDKs that do not have filler cells which are one site-wide. In this case, one-site-wide gaps are unfillable and will cause DRC violations.
detailed_place -disallow_one_site_gaps will disallow the detailed placer from leaving gaps between cells that are exactly one cell sitewide. This feature is mainly useful for PDKs that do not have filler cells which are one site-wide. In this case, one-site-wide gaps are unfillable and will cause DRC violations.
The detailed_place -disallow_one_site_gaps option prevents one-site-wide gaps between cells in detailed placement, addressing DRC issues in PDKs lacking one-site-wide filler cells.
The -report_file_name argument for the detailed_placement command specifies where the report for detailed placement should be saved. The report is saved in the JSON format and contains metrics related to detailed_placement. If this argument is not provided, no report will be saved for the detailed_placement
The -report_file_name argument for the detailed_placement command specifies where the report for detailed placement should be saved. The report is saved in the JSON format and contains metrics related to detailed_placement. If this argument is not provided, no report will be saved for the detailed_placement
The detailed_placement command's -report_file_name argument designates the storage location for the placement report in JSON format, detailing metrics of the placement. Without this argument, the command does not save a report.
The set_placement_padding command sets left and right padding in multiples of the row site width. Use the set_placement_padding command before legalizing placement to leave room for routing. Use the -global flag for padding that applies to all instances. Use -instances for instance-specific padding. The instances insts can be a list of instance names, or an instance object returned by the SDC get_cells command. To specify padding for all instances of a common master, use the -filter "ref_name == " option to get_cells.
The set_placement_padding command sets left and right padding in multiples of the row site width. Use the set_placement_padding command before legalizing placement to leave room for routing. Use the -global flag for padding that applies to all instances. Use -instances for instance-specific padding. The instances insts can be a list of instance names, or an instance object returned by the SDC get_cells command. To specify padding for all instances of a common master, use the -filter "ref_name == " option to get_cells.

The set_placement_padding command adjusts the left and right padding based on row site width multiples, essential for spacing before placement legalization. It supports global padding via -global or specific instance padding through -instances, with flexibility in selecting instances via a list or the SDC get_cells command.

The filler_placement command fills gaps between detail-placed instances to connect the power and ground rails in the rows. filler_masters is a list of master/macro names to use for filling the gaps. Wildcard matching is supported, so FILL* will match, e.g., FILLCELL_X1 FILLCELL_X16 FILLCELL_X2 FILLCELL_X32 FILLCELL_X4 FILLCELL_X8. To specify a different naming prefix from FILLER_ use -prefix <new prefix>.

The filler_placement command fills gaps between detail-placed instances to connect the power and ground rails in the rows. filler_masters is a list of master/macro names to use for filling the gaps. Wildcard matching is supported, so FILL* will match, e.g., FILLCELL_X1 FILLCELL_X16 FILLCELL_X2 FILLCELL_X32 FILLCELL_X4 FILLCELL_X8. To specify a different naming prefix from FILLER_ use -prefix <new prefix>.

The filler_placement command utilizes specified master names to fill gaps among placed instances for power and ground rail connectivity. It supports wildcard for master names selection and allows prefix customization with the -prefix option.

This command removes all filler cells.

This command removes all filler cells.

This command clears all placed filler cells.

The check_placement command checks the placement legality. It returns 0 if the placement is legal.

The check_placement command checks the placement legality. It returns 0 if the placement is legal.

The check_placement command verifies if the placement adheres to legality rules, returning 0 for a legal

The -verbose argument enables verbose logging in the check_placement command.

The -verbose argument enables verbose logging in the check_placement command.

Enabling the -verbose argument within the check_placement command activates detailed logging.

The argument -disallow_one_site_gaps disables one site gap during placement check.

The argument -disallow_one_site_gaps disables one site gap during placement check.

The -disallow_one_site_gaps argument prevents gaps of one site in the placement legality check.

The argument -report_file_name in check_placement command files name for saving the report
(e.g. report.json).

The argument -report_file_name in check_placement command files name for saving the report
(e.g. report.json).

Within the check_placement command, the -report_file_name argument designates the report's filename, such as report.json.

The optimize_mirroring command mirrors instances about the Y axis in a weak attempt to reduce the total half-perimeter wirelength (HPWL). No arguments are needed for this function.

The optimize_mirroring command mirrors instances about the Y axis in a weak attempt to reduce the total half-perimeter wirelength (HPWL). No arguments are needed for this function.

The optimize_mirroring command adjusts instances around the Y-axis to potentially lessen the total half-perimeter wirelength, requiring no additional arguments.

If you are a developer, you might find the following commands useful, 1. detailed_placement_debug: debug detailed placement. 2. get_masters_arg: get masters from a design. 3. get_inst_bbox: get the bounding box of an instance. 4. get_inst_grid_bbox: get the grid bounding box of an instance. 5. format_grid: format grid (takes in length x and site width w as inputs). 6. get_row_site: get row site name.

If you are a developer, you might find the following commands useful, 1. detailed_placement_debug: debug detailed placement. 2. get_masters_arg: get masters from a design. 3. get_inst_bbox: get the bounding box of an instance. 4. get_inst_grid_bbox: get the grid bounding box of an instance. 5. format_grid: format grid (takes in length x and site width w as inputs). 6. get_row_site: get row site name.
Developers might find commands like detailed_placement_debug, get_masters_arg, get_inst_bbox, get_inst_grid_bbox, format_grid, get_row_site beneficial for various debugging and configuration tasks.
The argument [-max_length <int>] takes an integer as input for setting the maximum number of bits that can be in each scan chain.
The argument [-max_length <int>] takes an integer as input for setting the maximum number of bits that can be in each scan chain.

The [-max_length <int>] argument sets a limit on the scan chain length in bits.
The argument [-clock_mixing] dictates how the architect mixes the scan flops based on the clock driver. The value no_mix creates scan chains with only one type of clock and edge. This may create unbalanced chains. The value clock_mix creates scan chains mixing clocks and edges. Falling edge flops are going to be stitched before the rising edge.
The argument [-clock_mixing] dictates how the architect mixes the scan flops based on the clock driver. The value no_mix creates scan chains with only one type of clock and edge. This may create unbalanced chains. The value clock_mix creates scan chains mixing clocks and edges. Falling edge flops are going to be stitched before the rising edge.
The [-clock_mixing] argument controls the scan flops' arrangement based on the clock driver, with options for mixing or separating clock types and edges.
The report_dft_config command prints the current DFT configuration to be used by preview_dft and insert_dft.

The report_dft_config command prints the current DFT configuration to be used by preview_dft and insert_dft.
The report_dft_config command outputs the current DFT setup for use in preview_dft and insert_dft.
This command prints a preview of the scan chains that will be stitched by insert_dft. Use this command to iterate and try different DFT configurations. This command does not perform any modification to the design.
This command prints a preview of the scan chains that will be stitched by insert_dft. Use this command to iterate and try different DFT configurations. This command does not perform any modification to the design.
This command offers a glimpse into the scan chains to be established by insert_dft, allowing for DFT configuration experimentation without altering the design.

The preview_dft command shows more information about each one of the scan chains that will be created.

The preview_dft command shows more information about each one of the scan chains that will be created.

The preview_dft command provides detailed insights into the forthcoming scan chains.
The insert_dft command implements the scan chains into the design by performing the following actions: Scan Replace, Scan Architect, Scan Stitch. The result is a design with scan flops connected to form the scan chains.
The insert_dft command implements the scan chains into the design by performing the following actions: Scan Replace, Scan Architect, Scan Stitch. The result is a design with scan flops connected to form the scan chains.
By executing actions like Scan Replace, Scan Architect, and Scan Stitch, the insert_dft command integrates scan chains into the design.
Here is an example that will create scan chains with a max length of 10 bits mixing all the scan flops in the scan chains: set_dft_config -max_length 10 -clock_mixing clock_mix report_dft_config preview_dft -verbose
Here is an example that will create scan chains with a max length of 10 bits mixing all the scan flops in the scan chains: set_dft_config -max_length 10 -clock_mixing clock_mix report_dft_config preview_dft -verbose

An example command to create 10-bit long scan chains with mixed flops is: set_dft_config -max_length 10 -clock_mixing clock_mix report_dft_config preview_dft -verbose insert_dft.
The limitations of DFT-Design for Testing are as follows: there are no optimizations for the scan chains, this is a WIP, there is no way to specify existing scan ports to be used by scan insertion, there is currently no way to define a user-defined scan path, and can only work with one-bit cells.
The limitations of DFT-Design for Testing are as follows: there are no optimizations for the scan chains, this is a WIP, there is no way to specify existing scan ports to be used by scan insertion, there is currently no way to define a user-defined scan path, and can only work with one-bit cells.
DFT limitations include the absence of scan chain optimizations, the inability to specify existing scan ports for insertion, the lack of a method for defining custom scan paths, and compatibility only with one-bit cells.
It is used to extract metrics after a successful clock_tree_synthesis run. These metrics are the number of Clock Roots, number of Buffers Inserted, number of Clock Subnets, and number of Sinks.
It is used to extract metrics after a successful clock_tree_synthesis run. These metrics are the number of Clock Roots, number of Buffers Inserted, number of Clock Subnets, and number of Sinks.
After running clock_tree_synthesis, use this to collect metrics such as the number of Clock Roots, Buffers Inserted, Clock Subnets, and Sinks.

The file to save cts reports. If this parameter is omitted, the report is streamed to stdout and not saved.
The file to save cts reports. If this parameter is omitted, the report is streamed to stdout and not saved.

Specifies the file for saving cts reports, defaulting to streaming to stdout if not provided.

This command is an option to plot the CTS to GUI.

This command is an option to plot the CTS to GUI.

This command allows for CTS visualization in the GUI.
You can use the bash shell to run commands and scripts.
You can use the bash shell to run commands and scripts.
The bash shell facilitates command and script execution.
These are the ways of installing OpenROAD flow scripts/ ORFS: Docker, Pre-built Binaries, Windows Subsystem for Linux (WSL), and Local Installation.
These are the ways of installing OpenROAD flow scripts/ ORFS: Docker, Pre-built Binaries, Windows Subsystem for Linux (WSL), and Local Installation.
OpenROAD flow scripts can be installed via Docker, Pre-built Binaries, WSL, or Local Installation.
The basic basic build command in ORFS is ./build_openroad.sh --help.

The basic basic build command in ORFS is ./build_openroad.sh --help.
The fundamental ORFS build command is ./build_openroad.sh --help.
The argument, o or —local, builds locally instead of building a Docker image.

The argument, o or —local, builds locally instead of building a Docker image.
The -o or --local argument opts for a local build over Docker.
The -l or --latest argument specifies to use of the head of branch –or_branch or 'master' by default for tools/OpenROAD.
The -l or --latest argument specifies to use of the head of branch –or_branch or 'master' by default for tools/OpenROAD.

The -l or --latest argument chooses the latest branch head for tools/OpenROAD, defaulting to master.

The -or_branch BRANCH_NAME argument stipulates the use of the head of branch BRANCH for tools/OpenROAD.
The -or_branch BRANCH_NAME argument stipulates the use of the head of branch BRANCH for tools/OpenROAD.

The -or_branch BRANCH_NAME argument selects the branch head of BRANCH for tools/OpenROAD.

This argument enumerates to use of a fork at REPO-URL (https/ssh) for tools/OpenROAD.

This argument enumerates to use of a fork at REPO-URL (https/ssh) for tools/OpenROAD.

Specifies using a tools/OpenROAD fork at a given REPO-URL.
The --no_init argument prescribes skipping initializing submodules.

The --no_init argument prescribes skipping initializing submodules.
The --no_init option omits submodule initialization.

The -t N or --threads N argument stipulates the use of N cpus when compiling software.

The -t N or --threads N argument stipulates the use of N cpus when compiling software.

The -t N or --threads N argument utilizes N CPUs for software compilation.
The -n or --nice flag in the build script instructs the build script to set all jobs to the maximim Unix "niceness". Niceness determines how much the operating system prefers to schedule process. The purpose of setting this flag is to ensure that a system is not overwhelmed by using all available processing primarily of the build jobs. The build will use all CPUs by default unless --threads is also given, then N threads are used instead.
The -n or --nice flag in the build script instructs the build script to set all jobs to the maximim Unix "niceness". Niceness determines how much the operating system prefers to schedule process. The purpose of setting this flag is to ensure that a system is not overwhelmed by using all available processing primarily of the build jobs. The build will use all CPUs by default unless --threads is also given, then N threads are used instead.
The build script's -n or --nice flag sets all jobs to the maximum Unix "niceness" level. This determines the priority with which the operating system schedules processes. The aim is to prevent system overload by allocating all processing power primarily to the build jobs. Without specifying otherwise, the build utilizes all CPUs, but specifying --threads allows for the use of a fixed number of threads.

This argument states to not use default flags set by this scrip during Yosys compilation.

This argument states to not use default flags set by this scrip during Yosys compilation.

This parameter instructs the script to ignore the default flags during Yosys compilation.

The '—yosys-args STRING' argument adds additional compilation flags for Yosys compilation.

The '—yosys-args STRING' argument adds additional compilation flags for Yosys compilation.

The '--yosys-args STRING' option allows for the inclusion of extra flags in Yosys compilations.
The '—openroad-args-overwrite' argument specifies to not use default flags set by this script during OpenROAD app compilation.

The '—openroad-args-overwrite' argument specifies to not use default flags set by this script during OpenROAD app compilation.
The '--openroad-args-overwrite' option commands the script not to apply its default flags during OpenROAD app compilations.

The '—openroad-args STRING' argument adds aditional compilation flags for OpenROAD app compilation.

The '—openroad-args STRING' argument adds aditional compilation flags for OpenROAD app compilation.

The '--openroad-args STRING' option introduces additional flags for compiling the OpenROAD app.

This argument initiates compiling LSOracle since it is disabled by default as it is not currently used on the flow.

This argument initiates compiling LSOracle since it is disabled by default as it is not currently used on the flow.

Activates the compilation of LSOracle, which is off by default as it's not part of the current workflow.
The '—lsoracle-args-overwrite' argument stipulates not to use default flags set by this script during LSOracle compilation.
The '—lsoracle-args-overwrite' argument stipulates not to use default flags set by this script during LSOracle compilation.

The '--lsoracle-args-overwrite' command directs the script to bypass default flags for LSOracle compilations.

This argument adds aditional compilation flags for LSOracle compilation.

This argument adds aditional compilation flags for LSOracle compilation.

This command introduces extra compilation flags for LSOracle.

The '—install-path PATH' delineates the path to install tools. Default is ${INSTALL_PATH}.

The '—install-path PATH' delineates the path to install tools. Default is ${INSTALL_PATH}.
The '--install-path PATH' option specifies the installation directory for tools, with ${INSTALL_PATH} as the default.
The '—clean' argument calls the git clean command interactively before compiling. It is useful to remove old build files.
The '—clean' argument calls the git clean command interactively before compiling. It is useful to remove old build files.
The '--clean' argument triggers an interactive git clean command before compiling, helping remove outdated build files.
This '—clean-force' argument will call git clean before compiling but WARNING: this option will not ask for confirmation. It is useful to remove old build files.
This '—clean-force' argument will call git clean before compiling but WARNING: this option will not ask for confirmation. It is useful to remove old build files.
The '--clean-force' command enforces a git clean operation before compilation without seeking user confirmation, aiding in the deletion of old build files.

This argument is only applicable to docker builds. It is used to copy platforms inside the docker image.

This argument is only applicable to docker builds. It is used to copy platforms inside the docker image.

This command is tailored for docker builds, enabling the copying of platforms into the docker image.
The ' —docker-args-overwrite' argument is only applicable for docker builds. They specify to not use default flags set by this script for Docker builds.
The ' —docker-args-overwrite' argument is only applicable for docker builds. They specify to not use default flags set by this script for Docker builds.
The '--docker-args-overwrite' command is designated for docker builds, instructing not to use the script's default Docker build flags.
This argument is only applicable to docker builds. It is used to add additional compilation flags for the Docker build.
This argument is only applicable to docker builds. It is used to add additional compilation flags for the Docker build.

This command, specific to docker builds, allows adding extra compilation flags for Docker builds.
The OpenROAD ("Foundations and Realization of Open, Accessible Design") Project was launched in June 2018 within the DARPA IDEA program. OpenROAD aims to bring down the barriers of cost, expertise, and unpredictability that currently block designers' access to hardware implementation in advanced technologies. The project team is developing a fully autonomous, open-source toolchain for digital SoC layout generation, focusing on the RTL-to-GDSII phase of system-on-chip design. Thus, OpenROAD holistically attacks the multiple facets of today's design cost crisis: engineering resources, design tool licenses, project schedule, and
The OpenROAD ("Foundations and Realization of Open, Accessible Design") Project was launched in June 2018 within the DARPA IDEA program. OpenROAD aims to bring down the barriers of cost, expertise, and unpredictability that currently block designers' access to hardware implementation in advanced technologies. The project team is developing a fully autonomous, open-source toolchain for digital SoC layout generation, focusing on the RTL-to-GDSII phase of system-on-chip design. Thus, OpenROAD holistically attacks the multiple facets of today's design cost crisis: engineering resources, design tool licenses, project schedule, and
Launched in June 2018 under DARPA's IDEA program, the OpenROAD ("Foundations and Realization of Open, Accessible Design") Project aims to democratize access to hardware implementation in cutting-edge technologies by developing a fully autonomous, open-source toolchain for digital SoC layout from RTL-to-GDSII. This initiative addresses the design cost crisis from various angles, including engineering resources, tool licenses, project timelines, and risk.
AutoTuner is a "no-human-in-loop" parameter tuning framework for commercial and academic RTL-to-GDS flows. AutoTuner provides a generic interface where users can define parameter configuration as JSON objects. This enables AutoTuner to easily support various tools and flows. AutoTuner also utilizes METRICS2.1 to capture PPA of individual search trials. With the abundant features of METRICS2.1, users can explore various reward functions that steer the flow autotuning to different PPA goals.
AutoTuner is a "no-human-in-loop" parameter tuning framework for commercial and academic RTL-to-GDS flows. AutoTuner provides a generic interface where users can define parameter configuration as JSON objects. This enables AutoTuner to easily support various tools and flows. AutoTuner also utilizes METRICS2.1 to capture PPA of individual search trials. With the abundant features of METRICS2.1, users can explore various reward functions that steer the flow autotuning to different PPA goals.
AutoTuner, a "no-human-in-loop" framework, facilitates parameter tuning across commercial and academic RTL-to-GDS flows. It employs a generic interface for parameter configurations via JSON objects and leverages METRICS2.1 to evaluate PPA across trials, supporting the exploration of diverse PPA-optimizing reward

AutoTuner contains top-level Python script for ORFS, each of which implements a different search algorithm. Current supported search algorithms are as follows:
Random/Grid Search, Population Based Training (PBT), Tree Parzen Estimator (HyperOpt), Bayesian + Multi-Armed Bandit (AxSearch), Tree Parzen Estimator + Covariance Matrix Adaptation Evolution Strategy (Optuna), Evolutionary Algorithm (Nevergrad)

AutoTuner contains top-level Python script for ORFS, each of which implements a different search algorithm. Current supported search algorithms are as follows:
Random/Grid Search, Population Based Training (PBT), Tree Parzen Estimator (HyperOpt), Bayesian + Multi-Armed Bandit (AxSearch), Tree Parzen Estimator + Covariance Matrix Adaptation Evolution Strategy (Optuna), Evolutionary Algorithm (Nevergrad)

AutoTuner incorporates a top-level Python script for ORFS, offering various search algorithms like Random/Grid Search, Population Based Training (PBT), Tree Parzen Estimator (HyperOpt), Bayesian + Multi-Armed Bandit (AxSearch), Tree Parzen Estimator + Covariance Matrix Adaptation Evolution Strategy (Optuna), and Evolutionary Algorithm (Nevergrad).

User-defined coefficient values (coeff_perform, coeff_power, coeff_area) of three objectives to set the direction of tuning are written in the script. Each coefficient is expressed as a global variable at the get_ppa function in PPAImprov class in the script (coeff_perform, coeff_power, coeff_area). Efforts to optimize each of the objectives are proportional to the specified coefficients.

User-defined coefficient values (coeff_perform, coeff_power, coeff_area) of three objectives to set the direction of tuning are written in the script. Each coefficient is expressed as a global variable at the get_ppa function in PPAImprov class in the script (coeff_perform, coeff_power, coeff_area). Efforts to optimize each of the objectives are proportional to the specified coefficients.

In AutoTuner, user-defined coefficient values (coeff_perform, coeff_power, coeff_area) direct the tuning efforts towards optimizing performance, power, and area. These coefficients are globally declared in the PPAImprov class's get_ppa function.

To set up AutoTuner, make sure you have a virtual environment set up with Python 3.9.X. There are plenty of ways to do this, we recommend using Miniconda, which is a free minimal installer for the package manager

To set up AutoTuner, make sure you have a virtual environment set up with Python 3.9.X. There are plenty of ways to do this, we recommend using Miniconda, which is a free minimal installer for the package manager

To initiate AutoTuner, ensure the setup of a virtual environment using Python 3.9.X, with Miniconda recommended for managing the environment.

Any variable that can be set from the command line can be used for tune or sweep.

Any variable that can be set from the command line can be used for tune or sweep.

Variables settable via command line can be employed for tuning or sweeping parameters.

The following design example is based on the design spm that implements a Single-port memory using gf180 platform. This procedure applies to any design for a given platform you choose. Start from the base directory OpenROAD-flow-scripts/flow. Step 1: Create the Verilog source files directory based on the top module name. Step 2: Create config.mk to define design configuration. Step 3: Define key design parameters in config.mk. Step 4: Define SDC constraints. Step 5: Add the design name to Makefile to run the flow with the make

The following design example is based on the design spm that implements a Single-port memory using gf180 platform. This procedure applies to any design for a given platform you choose. Start from the base directory OpenROAD-flow-scripts/flow. Step 1: Create the Verilog source files directory based on the top module name. Step 2: Create config.mk to define design configuration. Step 3: Define key design parameters in config.mk. Step 4: Define SDC constraints. Step 5: Add the design name to Makefile to run the flow with the make

Describing a design example for a Single-port memory using the gf180 platform, this process is adaptable to any design. It includes creating a Verilog source directory, configuring design settings in config.mk, defining key parameters, setting SDC constraints, and incorporating the design into the Makefile for execution.

While designing spm that implements a Single-port memory using gf180 platform, the export PLATFORM value can be gf180.

While designing spm that implements a Single-port memory using gf180 platform, the export PLATFORM value can be gf180.

For a Single-port memory design on the gf180 platform, the export PLATFORM variable should be set to gf180.
While designing spm that implements a Single-port memory using gf180 platform, the value of the parameter, export DESIGN_NAME can be spm.
While designing spm that implements a Single-port memory using gf180 platform, the value of the parameter, export DESIGN_NAME can be spm.
When designing a Single-port memory on the gf180 platform, the export DESIGN_NAME should be assigned to spm.
For the value of export VERILOG_FILES parameter while designing spm that implements a Single-port memory using gf180 platform, it can be $(sort $(wildcard ./designs/src/$(DESIGN_NICKNAME)/*.v))
For the value of export VERILOG_FILES parameter while designing spm that implements a Single-port memory using gf180 platform, it can be $(sort $(wildcard ./designs/src/$(DESIGN_NICKNAME)/*.v))
The export VERILOG_FILES parameter should be set to include all Verilog files under the design nickname directory for a Single-port memory design on the gf180 platform.
The parameter, export SDC_FILE can have the value ./designs/$(PLATFORM)/$(DESIGN_NICKNAME)/constraint.sdc while designing spm that implements a Single-
The parameter, export SDC_FILE can have the value ./designs/$(PLATFORM)/$(DESIGN_NICKNAME)/constraint.sdc while designing spm that implements a Single-
While designing a Single-port memory on the gf180 platform, the export SDC_FILE parameter should be set to the constraint file path.
The value of CORE_UTILIZATION may be subjective but one value for CORE_UTILIZATION while designing spm that implements a Single-port memory using gf180 platform40
The value of CORE_UTILIZATION may be subjective but one value for CORE_UTILIZATION while designing spm that implements a Single-port memory using gf180 platform40
The CORE_UTILIZATION value, while subjective, could be set to 40 for a Single-port memory design on the gf180 platform.
The value of the parameter, export PLACE_DENSITY while designing spm that implements a Single-port memory can be 0.6.
The value of the parameter, export PLACE_DENSITY while designing spm that implements a Single-port memory can be 0.6.

For a Single-port memory design, the export PLACE_DENSITY should be set to 0.6.
The value of the parameter, export TNS_END_PERCENT while designing spm that implements a Single-port memory can be 100.
The value of the parameter, export TNS_END_PERCENT while designing spm that implements a Single-port memory can be 100.

In the case of a Single-port memory design, the export TNS_END_PERCENT should be 100.
Environment variables are used in the OpenROAD flow to define various platform, design, and tool-specific variables to allow finer control and user overrides at various flow stages. These are defined in the config.mk file located in the platform and design-specific directories.
Environment variables are used in the OpenROAD flow to define various platform, design, and tool-specific variables to allow finer control and user overrides at various flow stages. These are defined in the config.mk file located in the platform and design-specific directories.
The OpenROAD flow utilizes environment variables defined in the config.mk file to specify various platform, design, and tool-specific settings, offering detailed control and customization at different stages.

The SKIP_REPORT_METRICS general variable if set to 1, then metrics, report_metrics does nothing. This is useful to speed up builds.

The SKIP_REPORT_METRICS general variable if set to 1, then metrics, report_metrics does nothing. This is useful to speed up builds.

If the SKIP_REPORT_METRICS variable is set to 1, metrics and report_metrics are bypassed, which accelerates the build process.

The variable, PROCESS signifies a technology node or process in use.

The variable, PROCESS signifies a technology node or process in use.

The PROCESS variable indicates the technology node or process in use.

This CORNER variable specifies the Library to select based on corner BC/TC/WC.

This CORNER variable specifies the Library to select based on corner BC/TC/WC.

The CORNER variable determines the Library selection based on corner cases like BC/TC/WC.

This variable, TECH_LEF, stipulates a technology LEF file of the PDK that includes all relevant information regarding metal layers, vias, and spacing requirements.

This variable, TECH_LEF, stipulates a technology LEF file of the PDK that includes all relevant information regarding metal layers, vias, and spacing requirements.

TECH_LEF defines a technology LEF file from the PDK containing details on metal layers, vias, and spacing requirements.

SC_LEF is used to specify the path to the technology standard cell LEF file.

SC_LEF is used to specify the path to the technology standard cell LEF file.

SC_LEF specifies the path to the technology standard cell LEF file.

GDS_FILES specifies the path to platform GDS files.

GDS_FILES specifies the path to platform GDS files.

GDS_FILES sets the path to platform GDS files.

LIB_FILES enumerates a Liberty file of the standard cell library with PVT characterization, input and output characteristics, timing, and power definitions for each cell.

LIB_FILES enumerates a Liberty file of the standard cell library with PVT characterization, input and output characteristics, timing, and power definitions for each cell.

LIB_FILES lists a Liberty file for the standard cell library, including PVT characterization, and definitions of input/output characteristics, timing, and power for each cell.

In OpenROAD Flow Scripts, the DONT_USE_CELLS variable stores a list of cells to avoid when performing both synthesis and place & route. Basic wildcard patterns (*) are supported. You may want to mark a cell as dont_use for several reasons, including 1) some cells may have complicated pin access patterns which are more likely to cause design rule violations during detailed routing, 2) some cells may be more prone to manufacturing variation, and will cause difficulty to close timing constraints, 3) A designer may not want to use certain cells during the implementation flow.

In OpenROAD Flow Scripts, the DONT_USE_CELLS variable stores a list of cells to avoid when performing both synthesis and place & route. Basic wildcard patterns (*) are supported. You may want to mark a cell as dont_use for several reasons, including 1) some cells may have complicated pin access patterns which are more likely to cause design rule violations during detailed routing, 2) some cells may be more prone to manufacturing variation, and will cause difficulty to close timing constraints, 3) A designer may not want to use certain cells during the implementation flow.

In OpenROAD Flow Scripts, DONT_USE_CELLS stores cells to avoid during synthesis and place & route, supporting basic wildcard patterns (*), due to reasons like complex pin access, susceptibility to manufacturing variation, or simply preference.

The variable SYNTH_HIERARCHICAL enables synthesis hierarchically, otherwise considered flat synthesis.

The variable SYNTH_HIERARCHICAL enables synthesis hierarchically, otherwise considered flat synthesis.

SYNTH_HIERARCHICAL toggles between hierarchical and flat synthesis.

LATCH_MAP_FILE variable specifies the list of latches treated as a black box by Yosys.

LATCH_MAP_FILE variable specifies the list of latches treated as a black box by Yosys.

LATCH_MAP_FILE lists latches considered as black boxes by Yosys.

This variable specifies a list of cells for the gating clock treated as a black box by Yosys.

This variable specifies a list of cells for the gating clock treated as a black box by Yosys.

Specifies cells to treat as black boxes for clock gating in Yosys.

List of adders treated as a black box by Yosys.

List of adders treated as a black box by Yosys.

Lists adders treated as black boxes by Yosys.

The variable, TIEHI_CELL_AND_PORT is used to tie high cells used in Yosys synthesis to replace a logical 1 in the Netlist.

The variable, TIEHI_CELL_AND_PORT is used to tie high cells used in Yosys synthesis to replace a logical 1 in the Netlist.

TIEHI_CELL_AND_PORT identifies high tie cells used in Yosys synthesis to replace logical 1 in the Netlist.

This variable is used to tie low cells used in Yosys synthesis to replace a logical 0 in the Netlist.

This variable is used to tie low cells used in Yosys synthesis to replace a logical 0 in the Netlist.

Defines low tie cells used in Yosys synthesis to replace logical 0 in the Netlist.

The MIN_BUF_CELL_AND_PORTS variable is used to insert a buffer cell to pass through wires. Used in

The MIN_BUF_CELL_AND_PORTS variable is used to insert a buffer cell to pass through wires. Used in synthesis.

MIN_BUF_CELL_AND_PORTS inserts a buffer cell for wire pass-through, used in synthesis.

The variable, ABC_CLOCK_PERIOD_IN_PS is used to specify the clock period to be used by STA during synthesis. Default value read from constraint.sdc.

The variable, ABC_CLOCK_PERIOD_IN_PS is used to specify the clock period to be used by STA during synthesis. Default value read from constraint.sdc.

ABC_CLOCK_PERIOD_IN_PS sets the clock period for STA during synthesis, with a default from constraint.sdc.

Default driver cell used during ABC synthesis.

Default driver cell used during ABC synthesis.

Specifies the default driver cell for ABC synthesis.

During synthesis, the set_load value specified by this, ABC_LOAD_IN_FF variable is used.

During synthesis, the set_load value specified by this, ABC_LOAD_IN_FF variable is used.

ABC_LOAD_IN_FF sets the load value during synthesis.

For hierarchical synthesis, this variable ungroups modules of the size given by this variable.

For hierarchical synthesis, this variable ungroups modules of the size given by this variable.

For hierarchical synthesis, ungroups modules based on specified size.

FLOORPLAN_DEF is used to specify the use of the DEF file to initialize floorplan.

FLOORPLAN_DEF is used to specify the use of the DEF file to initialize floorplan.

FLOORPLAN_DEF uses a DEF file to initialize the floorplan.

Placement site for core cells defined in the technology LEF file.

Placement site for core cells defined in the technology LEF file.

Defines the placement site for core cells in the technology LEF file.

TAPCELL_TCL specifies the path to the Endcap and Welltie cells file.

TAPCELL_TCL specifies the path to the Endcap and Welltie cells file.

TAPCELL_TCL points to the Endcap and Welltie cells file.

This variable, RTLMP_FLOW enables the Hierarchical RTLMP flow. By default it is disabled.

This variable, RTLMP_FLOW enables the Hierarchical RTLMP flow. By default it is disabled.

RTLMP_FLOW toggles the Hierarchical RTLMP flow, defaulting to off.

MACRO_HALO specifies to keep out a distance from macro, in X and Y, to standard cell row.

MACRO_HALO specifies to keep out a distance from macro, in X and Y, to standard cell row.

MACRO_HALO sets the keep-out distance around macros in X and Y to standard cell rows.

MACRO_PLACEMENT specifies the path of a file on how to place certain macros manually using read_macro_placement.

MACRO_PLACEMENT specifies the path of a file on how to place certain macros manually using read_macro_placement.

MACRO_PLACEMENT defines manual macro placement through a specified file.

This variable specifies the path of a TCL file on how to place certain macros manually.

This variable specifies the path of a TCL file on how to place certain macros manually.

Specifies a TCL file for manual macro placement.

Horizontal /vertical halo around macros (microns). Used by automatic macro placement.

Horizontal /vertical halo around macros (microns). Used by automatic macro placement.

Sets the halo around macros for automatic placement, in microns.

Horizontal/vertical channel width between macros (microns). Used by automatic macro placement when RTLMP_FLOW is disabled. Imagine channel=10 and halo=5. Then macros must be 10 apart but standard cells must be 5 away from a macro.

Horizontal/vertical channel width between macros (microns). Used by automatic macro placement when RTLMP_FLOW is disabled. Imagine channel=10 and halo=5. Then macros must be 10 apart but standard cells must be 5 away from a macro.

Determines the channel width between macros for automatic placement, in microns.

Blockage width overridden from default calculation.

Blockage width overridden from default calculation.

Overrides the default blockage width calculation.

This variable specifies the file path which has a set of power grid policies used by pdn to be applied to the design, such as layers to use, stripe width and spacing to generate the actual metal straps.

This variable specifies the file path which has a set of power grid policies used by pdn to be applied to the design, such as layers to use, stripe width and spacing to generate the actual metal straps.

Specifies the file path for power grid policies used by pdn.

MAKE_TRACKS variable outlines the Tcl file that defines adding routing tracks to a floorplan.

MAKE_TRACKS variable outlines the Tcl file that defines adding routing tracks to a floorplan.

MAKE_TRACKS outlines a Tcl file for adding routing tracks to a floorplan.

The metal layer on which to place the I/O pins horizontally (top and bottom of the die).

The metal layer on which to place the I/O pins horizontally (top and bottom of the die).

Specifies the metal layer for horizontal I/O pin placement.

The metal layer on which to place the I/O pins vertically (sides of the die).

The metal layer on which to place the I/O pins vertically (sides of the die).

Specifies the metal layer for vertical I/O pin placement.

Skip loading timing for a faster GUI load.

Skip loading timing for a faster GUI load.

Skips loading timing for faster GUI loading.

Skip the initial non-IO based global placement if IO constraints are present.

Skip the initial non-IO based global placement if IO constraints are present.

Skips initial non-IO based global placement if IO constraints exist.

Cell padding on both sides in site widths to ease routability during global placement.

Cell padding on both sides in site widths to ease routability during global placement.

Defines cell padding in site widths for global placement routability.

Cell padding on both sides in site widths to ease routability in detail placement.

Cell padding on both sides in site widths to ease routability in detail placement.

Defines cell padding in site widths for detail placement routability.
The desired placement density of cells. It reflects how spread the cells would be on the core area. 1.0 = closely dense. 0.0 = widely spread.
The desired placement density of cells. It reflects how spread the cells would be on the core area. 1.0 = closely dense. 0.0 = widely spread.
Sets the desired placement density of cells on the core area.
Check the lower boundary of the PLACE_DENSITY and add PLACE_DENSITY_LB_ADDON if it exists.

Check the lower boundary of the PLACE_DENSITY and add PLACE_DENSITY_LB_ADDON if it exists.
Adjusts the lower boundary of PLACE_DENSITY with an addon if present.

Remove power grid vias which generate DRC violations after detailed routing.
Remove power grid vias which generate DRC violations after detailed routing.

Removes power grid vias causing DRC violations after detailed routing.

Use additional tuning parameters during global placement other than default args defined in gloabl_place.tcl.

Use additional tuning parameters during global placement other than default args defined in gloabl_place.tcl.

Uses additional parameters for global placement beyond default args.
Enable detail placement with improve_placement feature.
Enable detail placement with improve_placement feature.
Enables detailed placement improvement.

Specifies how far an instance can be moved when optimizing.

Specifies how far an instance can be moved when optimizing.

Limits instance movement distance during optimization.
In OpenROAD Flow Scripts (ORFS), the GPL_TIMING_DRIVEN variable specifies whether the global placer (gpl) should use timing driven-placement. Timing-driven placement will cause gpl to ocassionally run static timing analysis (sta) during placement in order to determine the timing on each net. gpl will then reweight the nets based on how timing-critical they are. This process improves the timing of the netlist by decreasing the distance between timing-critical cells, but it also causes an increase in gpl runtime due to running timing
In OpenROAD Flow Scripts (ORFS), the GPL_TIMING_DRIVEN variable specifies whether the global placer (gpl) should use timing driven-placement. Timing-driven placement will cause gpl to ocassionally run static timing analysis (sta) during placement in order to determine the timing on each net. gpl will then reweight the nets based on how timing-critical they are. This process improves the timing of the netlist by decreasing the distance between timing-critical cells, but it also causes an increase in gpl runtime due to running timing

GPL_TIMING_DRIVEN toggles timing-driven global placement, affecting gpl runtime and netlist timing.

In OpenROAD Flow Scripts (ORFS), the GPL_ROUTABILITY_DRIVEN variable specifies whether the global placer (gpl) should use routability driven-placement. Routability-driven placement will cause gpl to ocassionally run global routing (grt) during placement in order to determine routing congestion hotspots. gpl will then reweight the nets based on how congestion data. This process improves the routability of the netlist by spacing out cells in routing congested areas, but it also causes an increase in gpl runtime due to running global routing.
In OpenROAD Flow Scripts (ORFS), the GPL_ROUTABILITY_DRIVEN variable specifies whether the global placer (gpl) should use routability driven-placement. Routability-driven placement will cause gpl to ocassionally run global routing (grt) during placement in order to determine routing congestion hotspots. gpl will then reweight the nets based on how congestion data. This process improves the routability of the netlist by spacing out cells in routing congested areas, but it also causes an increase in gpl runtime due to running global routing.
GPL_ROUTABILITY_DRIVEN toggles routability-driven global placement, influencing gpl runtime and netlist routability.

Specifies a capacitance margin when fixing max capacitance violations. This option allow you to overfix.
Specifies a capacitance margin when fixing max capacitance violations. This option allow you to overfix.

Sets a capacitance margin for overfixing max capacitance violations.

Specifies a slew margin when fixing max slew violations. This option allow you to overfix.
Specifies a slew margin when fixing max slew violations. This option allow you to overfix.

Sets a slew margin for overfixing max slew violations.

Override clock_tree_synthesis arguments
Override clock_tree_synthesis arguments

Overrides arguments for clock tree synthesis.
The CTS_BUF_CELL variable sets the root buffer cell used in the clock tree during clock tree synthesis (CTS)
The CTS_BUF_CELL variable sets the root buffer cell used in the clock tree during clock tree synthesis (CTS)
CTS_BUF_CELL selects the root buffer cell for clock tree synthesis.
The FILL_CELLS variable holds a list of cell names to use as filler cells. Wildcard patterns (*) are supported. Fill cells are used to fill empty cell sites which aids in satisfying design rules and density rules.
The FILL_CELLS variable holds a list of cell names to use as filler cells. Wildcard patterns (*) are supported. Fill cells are used to fill empty cell sites which aids in satisfying design rules and density rules.
FILL_CELLS lists cell names for filler cells, supporting wildcards.

Specifies a time margin for the slack when fixing hold violations. This option allow you to overfix.

Specifies a time margin for the slack when fixing hold violations. This option allow you to overfix.

Specifies a time margin for fixing hold violations.

Specifies a time margin for the slack when fixing setup violations.

Specifies a time margin for the slack when fixing setup violations.

Specifies a time margin for setup violation fixes.

Do not use gate cloning transform to fix timing violations (default: use gate cloning)

Do not use gate cloning transform to fix timing violations (default: use gate cloning)

Disables gate cloning for timing violation fixes.

Do not use pin swapping as a transform to fix timing violations (default: use pin swapping)

Do not use pin swapping as a transform to fix timing violations (default: use pin swapping)

Disables pin swapping for timing violation fixes.

In OpenROAD Flow Scripts, the TNS_END_PERCENT variable specifies what percent of violating timing paths will be fixed during timing optimization. TNS_END_PERCENT must be a floating point value between 0-100. However, even if TNS_END_PERCENT is 0, the worst path will always be fixed. The purpose of this flag is to allow the user some control over how much runtime and logic area is spent on timing optimization, with a higher value leading to more runtime/area and a lower value leading to less runtime/area.

In OpenROAD Flow Scripts, the TNS_END_PERCENT variable specifies what percent of violating timing paths will be fixed during timing optimization. TNS_END_PERCENT must be a floating point value between 0-100. However, even if TNS_END_PERCENT is 0, the worst path will always be fixed. The purpose of this flag is to allow the user some control over how much runtime and logic area is spent on timing optimization, with a higher value leading to more runtime/area and a lower value leading to less runtime/area.

TNS_END_PERCENT defines the percent of timing path fixes during optimization.

In OpenROAD Flow Scripts, the EQUIVALENCE_CHECK variable toggles whether a logical equivalence check is run after timing optimization to ensure logical correctness of the circuit. If EQUIVALENCE_CHECK is 1, the check is performed. If EQUIVALENCE_CHECK is any other value or unset, the check is not run. The default

In OpenROAD Flow Scripts, the EQUIVALENCE_CHECK variable toggles whether a logical equivalence check is run after timing optimization to ensure logical correctness of the circuit. If EQUIVALENCE_CHECK is 1, the check is performed. If EQUIVALENCE_CHECK is any other value or unset, the check is not run. The default

EQUIVALENCE_CHECK toggles logical equivalence checks post-optimization.

In OpenROAD Flow Scripts, the REMOVE_CELLS_FOR_EQY variable sets the list of cells to remove from the verilog netlist file produced specifically for netlist equivalence checking. Netlist equivalence checking is performed with the Yosys EQY tool. Wildcard patterns (*) are supported. This variable is passed directly to write_verilog -remove_cells <>.

In OpenROAD Flow Scripts, the REMOVE_CELLS_FOR_EQY variable sets the list of cells to remove from the verilog netlist file produced specifically for netlist equivalence checking. Netlist equivalence checking is performed with the Yosys EQY tool. Wildcard patterns (*) are supported. This variable is passed directly to write_verilog -remove_cells <>.

REMOVE_CELLS_FOR_EQY lists cells to exclude from equivalence checking netlist files.

This command creates power domain for a group of modules.

This command creates power domain for a group of modules.

Creates power domain for module groups.

This command creates logic port. Direction must be specified from: in, out, inout.

This command creates logic port. Direction must be specified from: in, out, inout.

Creates logic ports with specified directions.

Restructuring (the rst module) uses logic resynthesis to optimize combinational logic paths. Restructuring can be performed in either area or delay mode. Area mode will optimize the logic cell area of the paths, whereas delay mode will optimize for delay. An Area Mode Example: restructure -liberty_file ckt.lib -target area -tielo_pin ABC -tiehi_pin DEF. For Timing Mode Example: restructure -liberty_file ckt.lib -target delay -tielo_pin ABC -tiehi_pin DEF -slack_threshold 1 -depth_threshold 2.

Restructuring (the rst module) uses logic resynthesis to optimize combinational logic paths. Restructuring can be performed in either area or delay mode. Area mode will optimize the logic cell area of the paths, whereas delay mode will optimize for delay. An Area Mode Example: restructure -liberty_file ckt.lib -target area -tielo_pin ABC -tiehi_pin DEF. For Timing Mode Example: restructure -liberty_file ckt.lib -target delay -tielo_pin ABC -tiehi_pin DEF -slack_threshold 1 -depth_threshold 2.

Restructuring optimizes combinational logic paths in area or delay modes.

Yosys is a logic synthesis tool which is responsible for transforming register transfer-level (RTL) code into a gate-level netlist. OpenROAD is a place and route (P&R) tool which is responsible for implementing a gale-level netlist into a chip layout. Yosys is developed by the YosysHQ organization whereas OpenROAD is developed by The OpenROAD Project. OpenROAD Flow Scripts (ORFS) makes use of both of these tools (as well as KLayout) to form a full RTL-to-GDS flow.

Yosys is a logic synthesis tool which is responsible for transforming register transfer-level (RTL) code into a gate-level netlist. OpenROAD is a place and route (P&R) tool which is responsible for implementing a gale-level netlist into a chip layout. Yosys is developed by the YosysHQ organization whereas OpenROAD is developed by The OpenROAD Project. OpenROAD Flow Scripts (ORFS) makes use of both of these tools (as well as KLayout) to form a full RTL-to-GDS flow.

Yosys transforms RTL code into a gate-level netlist, while OpenROAD handles P&R from netlist to chip layout.

The Resizer module (rsz) can modify the netlist by inserting or removing buffers, as well as increasing or decreasing the drive strength of cells. rsz does not save a log of which instances were modified, because it would create an excessively long log file. A recommended workaround solution would be to save a layout file (such as DEF or ODB) before performing resizing, and then compare it to a layout file after resizing.

The Resizer module (rsz) can modify the netlist by inserting or removing buffers, as well as increasing or decreasing the drive strength of cells. rsz does not save a log of which instances were modified, because it would create an excessively long log file. A recommended workaround solution would be to save a layout file (such as DEF or ODB) before performing resizing, and then compare it to a layout file after resizing.

Resizer modifies the netlist by adjusting buffer sizes and cell drive strengths without detailed modification logs.

In OpenROAD, the runtime of the software is directly related to 1) the modeling accuracy and 2) the circuit optimization effort. Improving runtime is usually a tradeoff of one of these two categories. If you are comfortable with reducing optimization effort, such as when performing design space exploration, you could try the following techniques: 1) Relaxing timing constraints in the SDC file, 2) skipping unnesessary optimization routines, such as setup and hold time fixing, 3) Stopping the design flow early, such as after the placement step or clock tree synthesis (CTS) step.

In OpenROAD, the runtime of the software is directly related to 1) the modeling accuracy and 2) the circuit optimization effort. Improving runtime is usually a tradeoff of one of these two categories. If you are comfortable with reducing optimization effort, such as when performing design space exploration, you could try the following techniques: 1) Relaxing timing constraints in the SDC file, 2) skipping unnesessary optimization routines, such as setup and hold time fixing, 3) Stopping the design flow early, such as after the placement step or clock tree synthesis (CTS) step.

OpenROAD runtime is influenced by modeling accuracy and optimization efforts, with trade-offs for runtime improvement.

In OpenROAD, the runtime of the design can be affected by several factors: 1) the size of the netlist. Designs with a large number of instances (100k or more) can take significantly longer than small designs. 2) the physical area of the design. Designs with large die areas (~1 mm^2 or larger) can take longer because of having to store the die size in memory. 3) improper timing constraints. Designs with excessive timing constaints can cause optimization algorithms, particularly Resizer (rsz) to take excessively long. 4) host machine constraints. Large designs can require a large amount of RAM to run. If the required memory exceeds your machine's available memory, the runtime will be significantly increased. Additionally, OpenROAD scales well with core count, so using a CPU with a greater number of cores can improve runtime.

In OpenROAD, the runtime of the design can be affected by several factors: 1) the size of the netlist. Designs with a large number of instances (100k or more) can take significantly longer than small designs. 2) the physical area of the design. Designs with large die areas (~1 mm^2 or larger) can take longer because of having to store the die size in memory. 3) improper timing constraints. Designs with excessive timing constaints can cause optimization algorithms, particularly Resizer (rsz) to take excessively long. 4) host machine constraints. Large designs can require a large amount of RAM to run. If the required memory exceeds your machine's available memory, the runtime will be significantly increased. Additionally, OpenROAD scales well with core count, so using a CPU with a greater number of cores can improve runtime.

Netlist size, design area, timing constraints, and machine capabilities affect OpenROAD's runtime.

In OpenROAD, database units (DBU) are an integer representation of distance on a chip. In the LEF/DEF format, each technology specifies a conversion factor in terms of DBU/micron. For example, if a technology used a conversion factor of 2000 DBU/micron, that would mean that each DBU represents 0.0005 micron or 0.5 nm. The purpose for using DBU is that it enables faster calculation and no loss of precision compared to floating point representations.

In OpenROAD, database units (DBU) are an integer representation of distance on a chip. In the LEF/DEF format, each technology specifies a conversion factor in terms of DBU/micron. For example, if a technology used a conversion factor of 2000 DBU/micron, that would mean that each DBU represents 0.0005 micron or 0.5 nm. The purpose for using DBU is that it enables faster calculation and no loss of precision compared to floating point representations.

Database units (DBU) represent distance on a chip, converting to microns for precision without floating point errors.

In OpenROAD, an integer DBU value can be converted to a floating point micron value by using the ord::dbu_to_microns function

In OpenROAD, an integer DBU value can be converted to a floating point micron value by using the ord::dbu_to_microns function

DBU values convert to microns using ord::dbu_to_microns function.

OpenROAD's GUI can be started in two ways. One is to use the -gui flag when invoking OpenROAD from the command line (e.g. openroad -gui). The other option is to use the gui::show function from inside the openroad command interpreter.

OpenROAD's GUI can be started in two ways. One is to use the -gui flag when invoking OpenROAD from the command line (e.g. openroad -gui). The other option is to use the gui::show function from inside the openroad command interpreter.

OpenROAD's GUI is accessible via command line flag or internal show function.

OpenROAD Flow Scripts (ORFS) includes Makefile targets to open the GUI after each step. You can use the command make gui_*, where * is the name of the flow step (e.g. make_floorplan).

OpenROAD Flow Scripts (ORFS) includes Makefile targets to open the GUI after each step. You can use the command make gui_*, where * is the name of the flow step (e.g. make_floorplan).

OpenROAD Flow Scripts include Makefile targets for GUI access after flow steps.

OpenROAD is run using Tcl scripts. The following commands are used to read and write design data.

```
read_lef [-tech] [-library] filename
read_def filename
write_def [-version 5.8|5.7|5.6|5.5|5.4|5.3] filename
read_verilog filename
write_verilog filename
read_db filename
write_db filename
write_abstract_lef filename
```

Messages with the MPL prefix are from the macro placement (mpl) module. These messages are progress messages, informing the user which step of the HierRTLMP flow is being executed.