# Documentation of TraPL 17.10 (2017.08)

Author : Daohang Shi (shidaohang1990@gmail.com)

## Binary name wiscTraPL

Binary name is wiscTraPL with optional flags as below.

| Name | Option type | Value type | Description |
|---|---|---|---|
| -design_info | required | string | input file of all info about the design |
| -tar_gcell_util | optional | string | wiscTraPL output file of gcell utilization |
| -tar_gcell_tov | optional | string | wiscTraPL output file of gcell track overflow |

The formats of input/output files are introduced in the following sections and can also be found in README.

A sample run with all options specified has the following output.

```
-design_info        : /home/dshi/BENCHMARK/mgc_edit_dist_a/mgc_edit_dist_a_simple.badger.txt
-tar_gcell_util     : /home/dshi/BENCHMARK/mgc_edit_dist_a/mgc_edit_dist_a.tar_gcell_util.txt
-tar_gcell_tov      : /home/dshi/BENCHMARK/mgc_edit_dist_a/mgc_edit_dist_a.tar_gcell_tov.txt
<I> Starting wiscTraPL : Fri Aug 18 02:24:25 2017
<I> Benchmark Name: mgc_edit_dist_a
Importing nets...
Grid Size:     134x134
num_tracks_under_layer:
        0       4020  8040  1206016080
num_tracks_per_cell:
        30      30      30      30      30
layer_pitch_size:
        0.2     0.2     0.2     0.2     0.2
num_panels_under_layer:
        0       134     268     402     536
```

```
<I> Elapsed Time for DB Importing: 0.851277 seconds
<I> Starting network analyzer:
<I> Elapsed Time for TraPL Analyzer: 154.732 seconds
collecting total track usage for all g-cells...
<I> Results of wiscTraPL
        metal1 : 16574.8
        metal2 : 16475.7
        metal3 : 15799.8
        metal4 : 16916.4
        metal5 : 16499.4
Total : 82266
Exported to : /home/dshi/BENCHMARK/mgc_edit_dist_a/mgc_edit_dist_a.tar_gcell_util.txt
        metal1 : 647.272
        metal2 : 1769.33
        metal3 : 2022.22
        metal4 : 2109.79
        metal5 : 2137.31
Total : 8685.92
Exported to : /home/dshi/BENCHMARK/mgc_edit_dist_a/mgc_edit_dist_a.tar_gcell_tov.txt
<I> Finishing wiscTraPL : Fri Aug 18 02:27:03 2017
```

## Input File (-design_info)

After sourcing .lef, .def and .v files in Olympus, we can run some physical design steps and then export the DB information needed by wiscTraPL as shown in the following Tcl script. This script will output two files, a ".badger.txt" file, and a ".to-route.db" file. The ".badger.txt" file is used as input to traPL.

```
puts "Benchmark : $::env(BENCHMARK)"
set dir " YOUR DIRECTORY HERE "
# Replace with directory containing benchmark files
# All .lef, .def, and .v files should be in the same directory


set output "$dir/$::env(BENCHMARK).badger.txt"
read_lef -files $dir/tech_45nm.lef
read_lef -files $dir/cells.lef
read_verilog -files $dir/design.v
read_def -files $dir/floorplan.def


set clk_nets [get_nets ispd_clk]
set data_nets [get_nets -type signal -filter "@name != ispd_clk"]


place_global
```

```
place_detail


if { [llength $clk_nets ] > 0 } {


    #   determine clock nets


    #   create and apply nondefault rule for clocks
    create_nondefault_rule -name CLK -min metal4 -max metal5
    set_nondefault_rule -obj [get_obj nond CLK] -spacing 2000 -layers {metal4
metal5}
    set_nondefault_rule -obj [get_obj nond CLK] -layers [get_layers -type
via] -cut 2
    set_property -name nondefault_rule -value CLK -obj $clk_nets
    set_property -name balanced_gr -value true -obj $clk_nets


    #   run global->track->final routing for clocks
    route_global -grid fine -nets $clk_nets -min metal4 -max metal5
    set_rcd_model -stage post_route
    route_track -clock -nets $clk_nets
    route_final -nets $clk_nets


    #   report final routing of clock
    report_routing -nets $clk_nets -name clk_dr
}


#   save the DB right before starting GR
write_db -file $dir/$::env(BENCHMARK).to-route.db


config_route_global -reset_all true


#   global routing of data nets
route_global -nets_only $data_nets
report_route_cong -name data_gr

# customer input file
set fd [open $output w]


# default : small 30 tracks per grid
set num_track_per_grid 30
```

```
set grid_x_offset [ lindex [get_property -name offset [get_layers -type metal
-direction vertical]] 0 ]
set grid_y_offset [ lindex [get_property -name offset [get_layers -type metal
-direction horizontal]] 0 ]
set grid_x_size [expr $num_track_per_grid * [lindex [lreverse [get_property -
name pitch [get_layers -type metal -direction vertical]]] 0] ]
set grid_y_size [expr $num_track_per_grid * [lindex [lreverse [get_property -
name pitch [get_layers -type metal -direction horizontal]]] 0] ]


set area_x_offset [get_property -name xorigin [get_area_objects -type
core_area]]
set area_y_offset [get_property -name yorigin [get_area_objects -type
core_area]]
set area_width [get_property -name width [get_area_objects -type core_area]]
set area_height [get_property -name height [get_area_objects -type
core_area]]


puts $fd "BENCHMARK $::env(BENCHMARK)"


puts $fd "\nAREA_INFO $area_x_offset $area_y_offset $area_width $area_height"
puts $fd "\nGRID_INFO $grid_x_offset $grid_y_offset $grid_x_size
$grid_y_size"


puts $fd "\nLAYER_NUM [llength [get_layers -type metal]]"
puts $fd "\nLAYER_DIR [get_property -name direction [get_layers -type
metal]]"


puts $fd "\nPITCH [get_property -name pitch [get_layers -type metal]]"


# net pin
puts $fd "\nNET_NUM [ llength $data_nets ]"
foreach net $data_nets {
    set pins [get_pins -of_objects [get_nets $net]]
    puts $fd "\nNET_NAME $net \n\nPIN_NUM [llength $pins]"
    set pin_rects [get_objects -type phy_rect -of_objects $pins]
    set pin_x_min [get_property -name xorigin -objects $pin_rects]
    set pin_y_min [get_property -name yorigin -objects $pin_rects]
    set pin_width [get_property -name width -objects $pin_rects]
    set pin_height [get_property -name height -objects $pin_rects]
    set pin_layer [get_property -name layer -objects $pin_rects]
    for {set i 0} {$i < [llength $pins]} {incr i} {
        puts $fd "[lindex $pin_x_min $i] [lindex $pin_y_min $i] [lindex
$pin_width $i] [lindex $pin_height $i] [lindex $pin_layer $i]"  }
```

```
    set wires [get_wires -route_types global_route -quiet true -of_objects
[get_nets $net] ]
    puts $fd "\nWIRE_NUM [llength $wires]"


    if { [llength $wires] == 0 } { continue }


    puts $fd [ get_property -name xstart -objects $wires]
    puts $fd [ get_property -name ystart -objects $wires]
    puts $fd [ get_property -name layer -objects $wires]
    puts $fd [ get_property -name is_via -objects $wires]
    puts $fd [ get_property -name length -objects $wires]
}


# cell loc
puts $fd "\nCELL_NUM [llength [get_cells]]"
foreach cell [get_cells] {
    puts $fd "\n$cell \
    [get_property -name xorigin -objects [get_cells $cell]] \
    [get_property -name yorigin -objects [get_cells $cell]] \
    [get_property -name width -objects [get_cells $cell]] \
    [get_property -name height -objects [get_cells $cell]] \
    [get_property -name is_macro [get_lib_cells [get_property -name lib_cell
[get_cells $cell]]]]"
}


# p/g mesh
set vdd [get_wires -of_objects [get_nets -type power] -filter !@is_via]
puts $fd "\nVDD_NUM [llength $vdd]"


puts $fd [ get_property -name xstart -objects $vdd]
puts $fd [ get_property -name ystart -objects $vdd]
puts $fd [ get_property -name layer -objects $vdd]
puts $fd [ get_property -name is_via -objects $vdd]
puts $fd [ get_property -name length -objects $vdd]
puts $fd [ get_property -name width -objects $vdd]


set gnd [get_wires -of_objects [get_nets -type ground] -filter !@is_via]
puts $fd "\nGND_NUM [llength $gnd]"


puts $fd [ get_property -name xstart -objects $gnd]
puts $fd [ get_property -name ystart -objects $gnd]
puts $fd [ get_property -name layer -objects $gnd]
puts $fd [ get_property -name is_via -objects $gnd]
puts $fd [ get_property -name length -objects $gnd]
puts $fd [ get_property -name width -objects $gnd]
```

```
# clock tree
if { [llength $clk_nets] > 0 } {


    set wires [get_wires -of_objects [get_nets ispd_clk] -route_type detail_route -
filter !@is_via]


    puts $fd "\nCLK_NUM [llength $wires]"
    set clk_x_min [ get_property -name xstart -objects $wires]
    set clk_y_min [ get_property -name ystart -objects $wires]
    set clk_layer [ get_property -name layer -objects $wires]
    set clk_length [ get_property -name length -objects $wires]
    set clk_width [ get_property -name width -objects $wires]


    for {set i 0} {$i < [llength $wires]} {incr i} {
        puts $fd "\
        [lindex $clk_x_min $i] \
        [lindex $clk_y_min $i] \
        [lindex $clk_layer $i] \
        [lindex $clk_length $i] \
        [lindex $clk_width $i]" }
} else {    puts $fd "\nCLK_NUM 0"  }


close $fd

exit
```

## Output files (-tar_gcell_util and -tar_gcell_tov)

Once TraPL finishes its analysis, it can export track utilization / track overflow of all global cells if -tar_gcell_util / -tar_gcell_tov is specified as below.

```
<I> Results of wiscTraPL
      metal1 : 20672.3
      metal2 : 17705.5
      metal3 : 17451.9
      metal4 : 17829.6
      metal5 : 19221.7
Total : 92881.1
Exported to : /home/dshi/BENCHMARK/mgc_des_perf_a/mgc_des_perf_a.tar_gcell_util.txt
      metal1 : 578.167
      metal2 : 673.9
```

```
        metal3 : 792.267
        metal4 : 712.478
        metal5 : 910.289
Total : 3667.1
Exported to : /home/dshi/BENCHMARK/mgc_des_perf_a/mgc_des_perf_a.tar_gcell_tov.txt
```

In the top of the output file there are a few lines commented out by '#'.

After that, it will specify the dimensions of the global routing grid graph.

For example, a design has 2 routing layers, each of which has 3x4 global cells. The output file will be as below.

```
# wiscTraPL output file
dimX : 20
dimY : 30
dimZ : 3

0.8 0.8 0.6
0.6 0.7 0.5
0.8 0.8 0.6
0.5 0.7 0.6

0.8 0.8 0.6
0.6 0.7 0.5
0.8 0.8 0.6
0.5 0.7 0.6
```

## Appendix

An easy-to-modify testing script in bash is attached to save your time.

```
if [ $# -eq 0 ] ; then
    echo "Error!"
    echo "<script_filename> -t <design_list> -b <bin_name> -a <annotation>"
    exit
fi
```

```
#   INIT OF ENV VARS
project_dir=??
log_dir=??
traPL_bin=??
annotation=??
# Replace these question marks with desired values


while [[ $# -gt 0 ]] ; do
    case "$1" in
        -t|--test_case)
            case "$2" in
                "all")  TEST_LIST=(
                        design_1 \
                        design_2 \
                        design_3 ) ; shift 2 ;;
                    *)  TEST_LIST+=$2 ; shift 2 ;;
            esac ;;
        -b|--bin)
            traPL_bin=($2) ; shift 2 ;;
        -a|--annotation)
            annotation=($2) ; shift 2 ;;
        *)
            echo "Internal error!" ; exit 1 ;;
    esac
done

for design in ${TEST_LIST[*]}; do
    benchmark_dir=??
    input_design_info=??
    tar_gcell_util=??
    tar_gcell_tov=??
    # Replace these question marks with desired values

    timestamp=$(date +"%Y-%b-%d-%H_%M_%S")
    log_file=${log_dir}/tagra_${design}_${timestamp}.$annotation.log


    echo "####################################"
    echo "RUNNING wiscTraPL"
    echo "  Benchmark Dir  : ${benchmark_dir}"
    echo "  Log File       : ${log_file}"


    echo "  Running command: ${traPL_bin} -design_info ${input_design_info} -
tar_gcell_util ${tar_gcell_util} -tar_gcell_tov ${tar_gcell_tov}"
    ${traPL_bin} -design_info ${input_design_info} -tar_gcell_util
${tar_gcell_util} -tar_gcell_tov ${tar_gcell_tov} | tee ${log_file}
done
```