# POCOP マクロ、ICCompiler、ClubLayout(ふんが)、2011.4.7 改訂

## 0.1 はじめに

POCOP_TOP は、パイプライン化した POCO のコアのみをレイアウトしたものである。階層化されていて、まず、実体である POCOP をレイアウトし、これを POCOP_TOP のど真ん中に置いて全体をレイアウトする作戦を取っている。本ドキュメントは、このマクロ部分に関するものである。全く同じマクロのレイアウトは先に Astro で行い、これについてのドキュメントは、POCOP マクロ Astro 編に示した。本稿はこれを ICCompiler で行ったものである。

ふんが研用には、/home/hunga/verilog/f65/icc にレイアウト用のディレクトリがある。

## 0.2 ディレクトリセットアップ

以下のファイルはどこに置いておいても良いがやはり適当に整理して置くのが良いと思う。lib などは f65 で共通なので、どこかのをリンクしても良い。

- CS202IO(I/O の Milkyway ライブラリ、リンクする。今回は使わないが POCOP_TOP 部を作る際に必要) : /home/vdec/lib/fujitsu65/milky/data/CS202IO/lib/CS202IO/
- CS202SZ(Cell の Milkyway ライブラリ、リンクする) : /home/vdec/lib/fujitsu65/milky/data/CS202IO/lib/CS202SZ/
- lib: f65.tf, gdsout.map, tlu2mw.map, tlu_plus_mfe.best, tlu_plus_mfe.worst を入れておく
- tdf: POCOP.tdf を入れておく
- vnet: 合成後の verilog 記述、POCOP.vnet, POCOP_TOP.vnet を入れておく
- sdc: 合成時に生成した POCOP.sdc を入れておく
- scripts: tcl ファイルを入れておく。

ここでは log ファイル等は直接 icc の下に吐き出されるがこれが嫌な方は専用のディレクトリを作れば良い。また、tdf は scripts の中に入れてもいい。

# 1 POCOP.tcl の解説

icc_shell -gui で ICC を立ち上げて、source POCOP.tcl とやると以下が実行され、POCOP のレイアウトが生成される。ここでは、2008 年版を使った。ちなみに 2009 年版から tdf が使えなくなったが、これは

これを順に解説する。

## 1.1 初期設定

以下、利用する環境を設定する。

```
lappend serach_path ./
source scripts/set_sz.tcl
set mw_logic0_net VSS
set mw_logic1_net VDD
define_name_rules verilog -allowed "A-Z0-9_"
set_write_stream_options  -map_layer ./lib/gdsout.map -child_depth 0 \
-output_pin {text geometry}  -output_design_intent
```

## 1.2 verilog file の読み込み

ライブラリの生成する。

```
create_mw_lib POCOP -technology lib/f65.tf -mw_reference_library "CS202SZ CS202IO"
```

今回は CS202IO は要らなかったかも、、、
で、オープンする。

```
open_mw_lib POCOP
```

ライブラリは一度作ってしまったら、オープンだけでいい。次に Verilog ソースを読み出す。

```
import_designs vnet/POCOP.vnet -format verilog -top POCOP
```

ここでレイアウト用のウインドウが表われる。

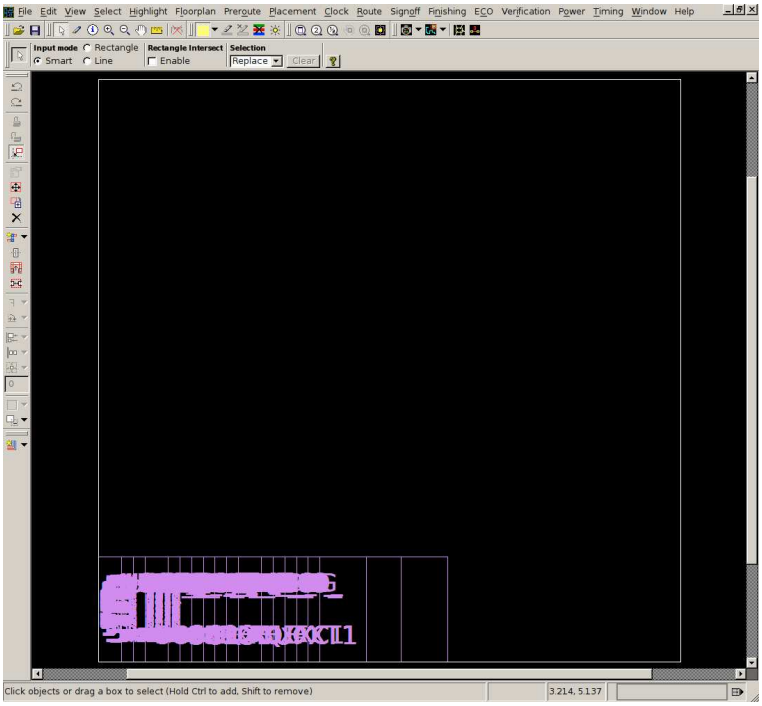


図 1: 読み込んだ後

次に、TLU を読み込む。

```
set_tlu_plus_files -max_tluplus ./lib/tlu_plus_mfe.worst \
        -min_tluplus ./lib/tlu_plus_mfe.best \
        -tech2itf_map ./lib/tlu2mw.map
```

ここで、読み込んだセルの電源、グランドを接続する。これは Astro の connect_pg と同じで、フローのあっちこっちで行っている。また、ピンだけではなく、itie で 0,1 にも値を与える。

```
derive_pg_connection -power_net {VDD} \
        -ground_net {VSS} -power_pin {VDD} -ground_pin {VSS}
derive_pg_connection -tie -power_net VDD \
        -ground_net VSS
```

次に、sdc を読み込んでおく。

```
read_sdc ./sdc/POCOP.sdc
```

ここで、ファンアウトの設定がエラーになるので、削った方がいいのかも、、

## 1.3 フロアプラン

まず、tdf を読み込む

```
read_io_constraints ./tdf/POCOP.tdf
```

tdf とは terminal definition file といってピン配置を指定するファイル。ちなみにこのファイル形式は 2009 年版以降使えなくなってしまった。その代り

```
set_pin_physical_constraints -pin_name BIDATA_0[34] -layer MET2 -width 0.2 -side 2 -order 1
```

という形式を使う。この形式は-side を番号で記載することになっており、左下から時計回りで 1,2,3,4 と付けて行く。つまり tdf の”LEFT” が 1 になり、TOP=2, RIGHT=3, BOTTOM=4 となる。これは分かりにくいのだが、単純な四角形でなくても定義できる点が優れている。
次がフロアプラン

```
source ./scripts/POCOP.fp
```

中身は以下の通りで、サイズと周辺の余裕を定義した。サイズは 100 x 100 としてあるが、もう少し小さめでも良いかもしれない。

```
initialize_floorplan -control_type width_and_height -core_width 100 \
-core_height 100 -start_first_row -bottom_io2core 0.9 -top_io2core 0.9 \
-left_io2core 0.9 -right_io2core 0.9
```

これでフロアプランができる。Astro よりも可愛い感じがする。

## 1.4 配置

次にストラップを作る。今回は小林先生のフローの影響で、VDD/VSS をペアにしてかなり密に作った。MET6 を使った。

```
source ./scripts/POCOP.strap
```

中身は、以下

```
create_power_straps  -direction vertical -start_at 3.6 -num_placement_strap 7 -increment_x_or_y 14.4 \
-nets  {VDD VSS} -layer MET6 -width 1.8 -step 14.4 \
-extend_low_ends force_to_boundary_and_generate_pins \
-extend_high_ends force_to_boundary_and_generate_pins
```

これで、ストラップが生成される。
次にタップセルを入れる。ストラップとの順番は分からないが、他のセルの配置の前にやっておく必要がある。間隔はルールに基づき、はじっこにも入れることにした。

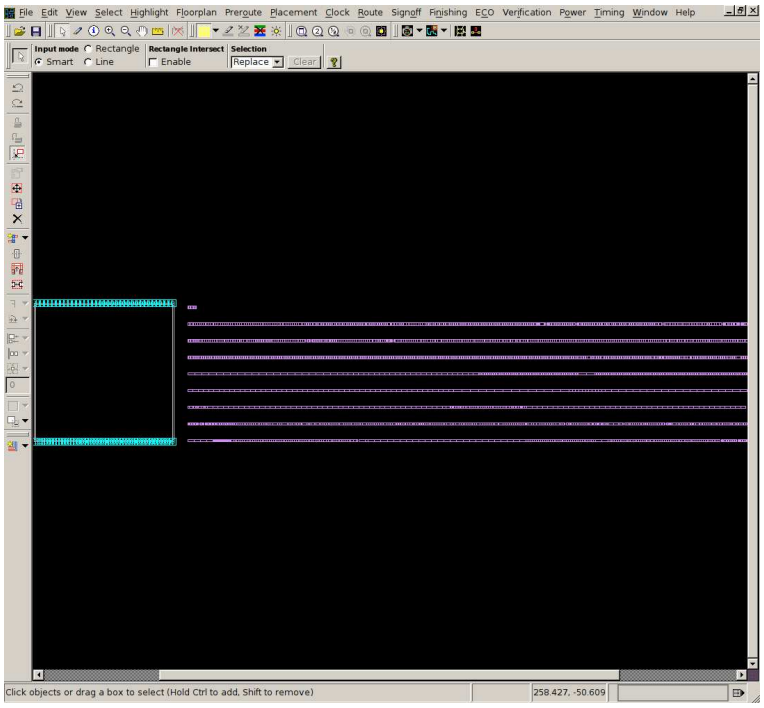


図 2: フロアプラン後

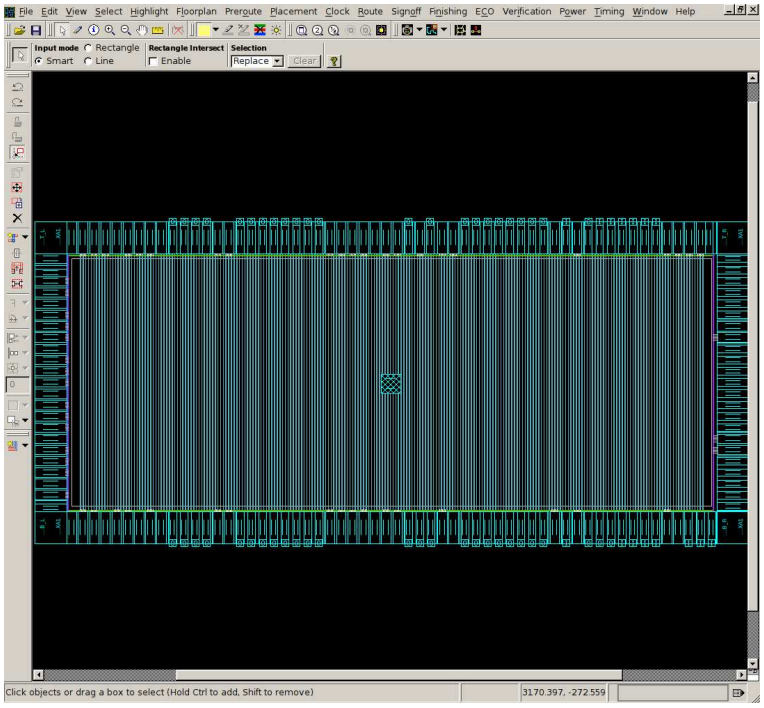


図 3: ストラップ

```
add_tap_cell_array -master_cell_name {SC23YUZTAP021} -distance 36.0 \
-skip_fixed_cells false -no_tap_cell_under_layer {M1 M2} \
-left_macro_blockage_extra_tap must_insert \
-right_macro_blockage_extra_tap must_insert \
-left_boundary_extra_tap must_insert \
-right_boundary_extra_tap must_insert \
-connect_power_name {VDD} -connect_ground_name {VSS}
```

ここで、配置を行う。

```
create_placement -congestion -congestion_effort high
legalize_placement
place_opt
```

今回は動作速度を重視せず、なるべくセルをバラけるように配置して混雑を避ける方針を取っている。
で、例によって電源をつなぐ。

```
derive_pg_connection -power_net VDD \
        -ground_net VSS -power_pin VDD -ground_pin VSS
derive_pg_connection -tie -power_net VDD \
        -ground_net VSS
```

しかし、ここではつなぐものはないみたいだ。

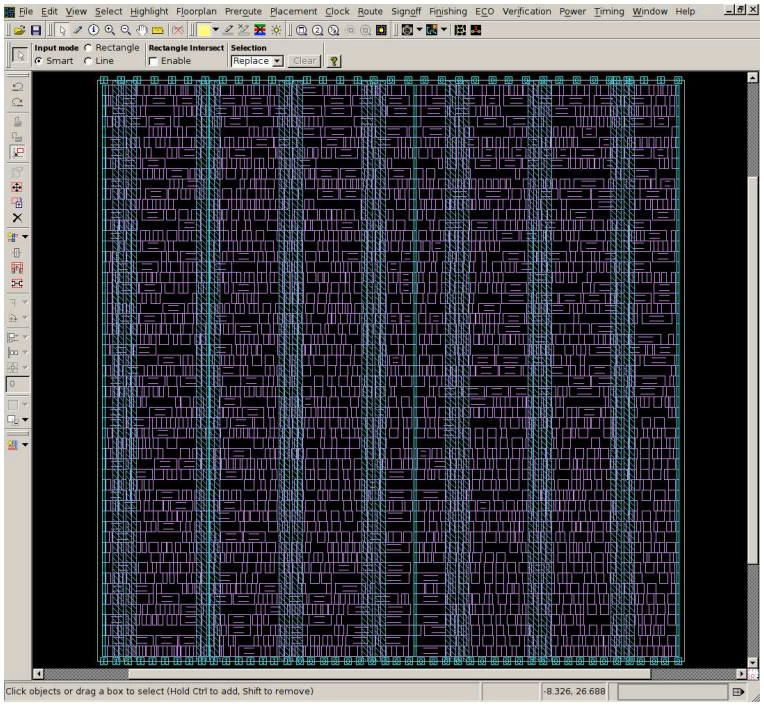


図 4: 配置後

次にセル間のレールを張ってやる。

```
preroute_standard_cells -connect horizontal \
        -extend_to_boundaries_and_generate_pins -port_filter_mode off \
        -cell_master_filter_mode off -cell_instance_filter_mode off \
        -voltage_area_filter_mode off
```

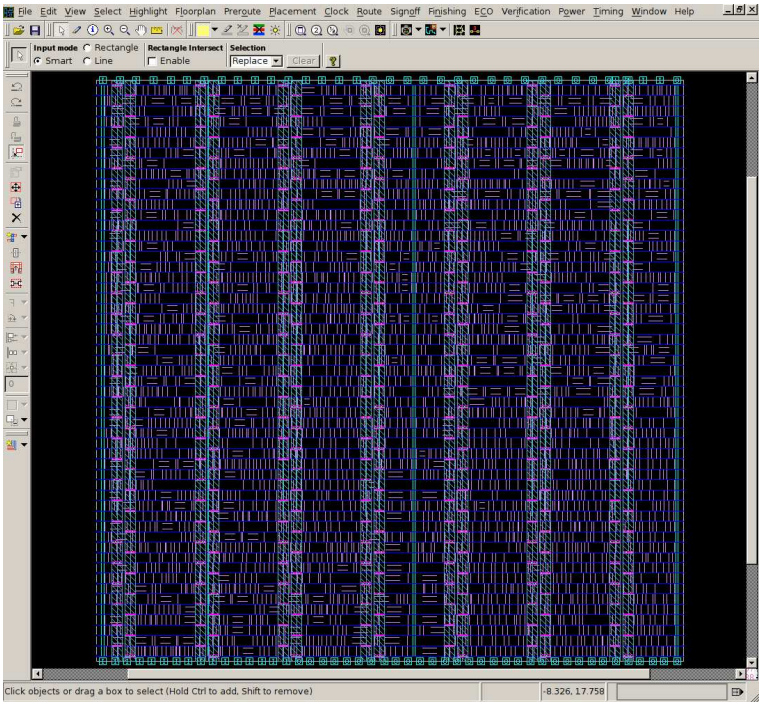


図 5: レールを張った後

## 1.5 配線

まずは配線の準備を行う。

```
set_route_options -groute_timing_driven true \
        -droute_stack_via_less_than_min_area forbid \
        -same_net_notch check_and_fix \
        -fat_wire_check merge_then_check
```

次に、12 層は配線に使用しないことにする。これは、Astro で 12 層を使うと落ちるからなので、ICC ならばできるかもしれない。でもコワいから制約した。勇敢は君はこれなしでやってみよう。

```
set_ignored_layers -max_routing_layer METG2
```

次にアンテナルールを読み込む。ちなみにこれが一番わからなかったのだが、Astro で使っていた制約と小林先生のフローに付いていたのを見ながら適当に作った。

```
set lib [current_mw_lib]
remove_antenna_rules $lib

define_antenna_layer_ratio_scale $lib -layer "MET1"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "MET2"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "MET3"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "MET4"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "MET5"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "MET6"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "METS1"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "METS2"    -layer_scale 1     -accumulate_scale 1
```

```
define_antenna_layer_ratio_scale $lib -layer "METS3"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "METG1"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "METG2"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "METTOP"    -layer_scale 1     -accumulate_scale 1
define_antenna_layer_ratio_scale $lib -layer "CUTS1"    -layer_scale 30.0     -accumulate_scale 30.0
define_antenna_layer_ratio_scale $lib -layer "CUTS2"    -layer_scale 30.0     -accumulate_scale 30.0
define_antenna_layer_ratio_scale $lib -layer "CUTS3"    -layer_scale 30.0     -accumulate_scale 30.0
define_antenna_layer_ratio_scale $lib -layer "CUTG1"    -layer_scale 7.5     -accumulate_scale 7.5
define_antenna_layer_ratio_scale $lib -layer "CUTG2"    -layer_scale 7.5     -accumulate_scale 7.5
define_antenna_layer_ratio_scale $lib -layer "CUTTOP"    -layer_scale 12.0     -accumulate_scale 12.0

define_antenna_rule $lib -mode 2 -diode_mode 4 -metal_ratio 2000 -cut_ratio 2000
define_antenna_layer_rule $lib -mode 2 -layer "MET1"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "MET2"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "MET3"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "MET4"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "MET5"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "MET6"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "METS1"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "METS2"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "METS3"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "METG1"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "METG2"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "METTOP"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUT12"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUT23"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUT34"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUT45"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUT56"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUTS1"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUTS2"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUTS3"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUTG1"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUTG2"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
define_antenna_layer_rule $lib -mode 2 -layer "CUTTOP"   -ratio 2000 -diode_ratio {0.01919 0 1 100000}
```

で、これを読み込む。

```
source scripts/antenna_rule.tcl
```

次は小林先生のフローを真似した。アンテナ避けの設定。

```
set_parameter -name doAntennaConx -value 4
```

で、クロックツリーを生成する。

```
clock_opt
```

図中で縦方向の赤線と横方向の緑線でクロックが形成されているのが確認できる。

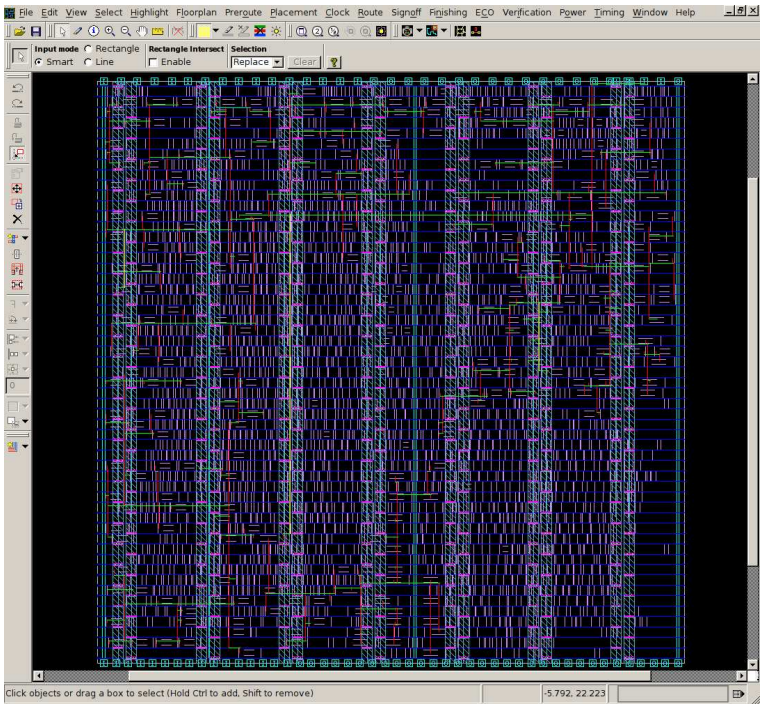


図 6: クロックツリー後

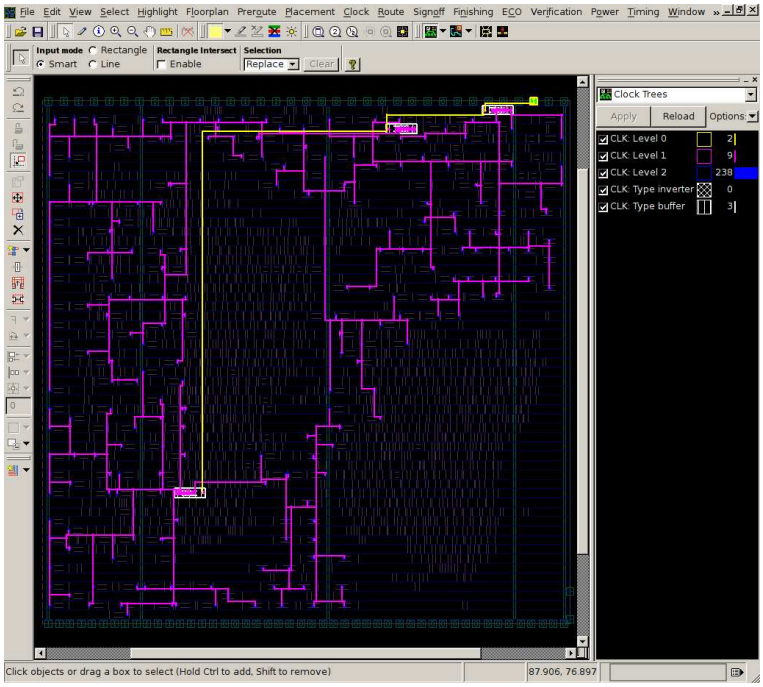


図 7: クロックツリーとバッファの様子

Window メニューから CTS window を表示し、Reload の後、inverter, buffer を表示すると以下のようになる。
次に配線。終了後、リペアを 10 回位回してみた。

```
route_auto
route_search_repair -rerun_drc \
        -trim_antenna_of_user_wires -loop 10
```

Violation が 0 になっているので、次に行く。小林フローにならって、アンテナを確認して、電源とグランドに接続する。

```
verify_route -antenna
derive_pg_connection -power_net VDD \
        -ground_net VSS -power_pin VDD -ground_pin VSS
```

以下が配線後のレイアウトで、結構カラフルである。

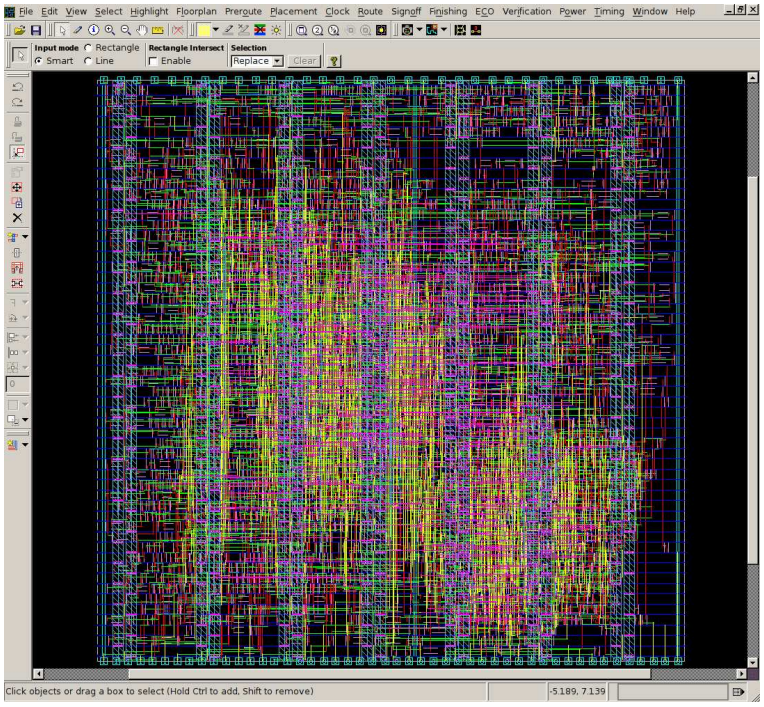


図 8: 配線後

## 1.6 仕上げ

フィラーを挿入する。ここでは CUBA とその他を混ぜて自動で挿入した。Astro で社長が開発した全面 CUBA にして後で削るというテクニックが ICC ではうまくいかなかったためである。これは多分もっとがんばればできるんではないかと思うが、ま、面積に余裕があればこれでも当面問題ないので、良いかと思う。
なんとなく、電源接続もやってみるが、まったく接続してくれなかった。

```
insert_stdcell_filler -connect_to_power VDD \
        -connect_to_ground VSS \
        -cell_without_metal {SC23YUZS021 SC23YUZS011} \
        -cell_with_metal {SC23YUZCUBAS081}
```

```
derive_pg_connection -power_net VDD \
        -ground_net VSS -power_pin VDD -ground_pin VSS
```

次に配線の最適化の設定をしておく。

```
preroute_standard_cells -mode tie -nets {VDD VSS} \
-connect horizontal -port_filter_mode off \
-cell_master_filter_mode off -cell_instance_filter_mode off \
-voltage_area_filter_mode off
```

配線の最適化を行う。

```
optimize_wire_via
```

これで一応出来上がり。

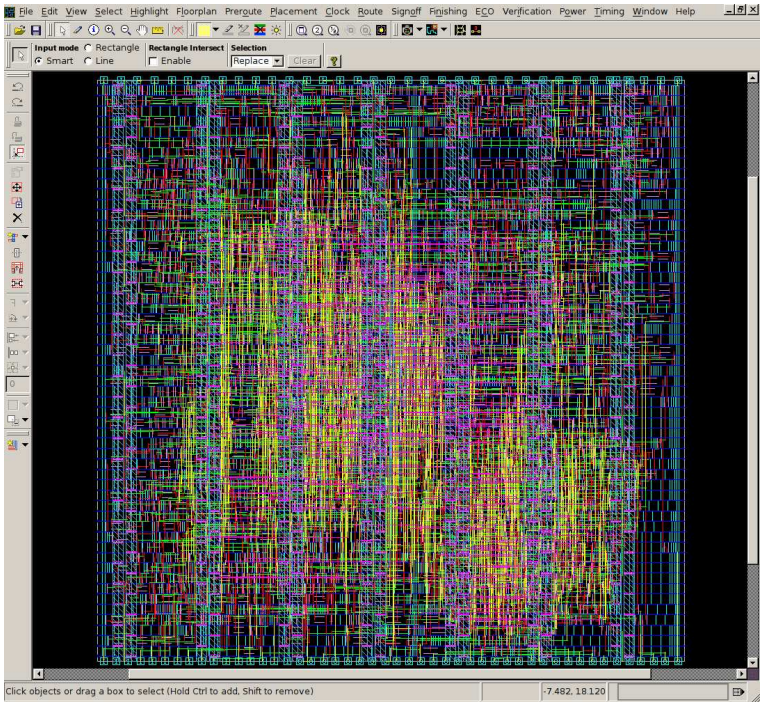


図 9: 最終レイアウト

ここで、レイアウトを保存する。

```
save_mw_cel -design POCOP
```

で、ストリームデータ、LVS で使う Verilog、シミュレーション用 Verilog, sdf を吐き出す。ストリームデータは小林レイアウトと違って最上位階層のみなので注意！

```
set_write_stream_options  -map_layer ./lib/gdsout.map -child_depth 0 \
-output_pin {text geometry}  -output_design_intent
write -format ddc -hierarchy -output POCOP_out.ddc
write_verilog -no_physical_only_cells POCOP_out.v
write_verilog -no_physical_only_cells -pg POCOP_lvs.v
write_stream -format gds -lib_name POCOP -cells {POCOP} POCOP.gds
```

ここで、マクロのフレームを吐き出さないと、上位階層から使えない。次に、Interface Logic Model というのを吐き出す。これが小林先生に見つけていただいた奥義で、これを付けると、上位階層で、モジュール内のタイミングを考えたレイアウトが可能になる。最後にアンテナ情報も吐き出しておく。

```
create_macro_fram \
        -treat_all_blockage_as_thin_wire -library_name {POCOP} \
        -cell_name {POCOP}
create_ilm -include_xtalk
extract_hier_antenna_property -cell_name {POCOP}
```

## 1.7 トラブル

IC Compiler は、コマンドも簡単で、非常に気持ちよくレイアウトを生成してくれる。しかし、最初ルート制約の stackvia の辺の指定をしなかった所、見た目もすごい stack via ができて DRC エラーになった。やはり下記の制約をちゃんとやらないとダメなようだ。

```
set_route_options -groute_timing_driven true \
        -droute_stack_via_less_than_min_area forbid \
        -same_net_notch check_and_fix \
        -fat_wire_check merge_then_check
```