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 TOP 階層に関するものである。方針は Astro、POCOP トップのデザインフローに倣っている。ディレクトリのセットアップ、POCOP マクロの作り方は、ICC、POCOP マクロのデザインフローを参照のこと。 hunga/verilog/f65/icc にレイアウト用のディレクトリがあるのでそこを参照のこと。

これは Astro 編にも書いたが、繰り返しておく。

POCOP_TOP は、I/O を含む e-shuttle 65nm 半区画のチップ上に POCOP を載せるもので、このままテープアウト可能な設計を目指している。レイアウト講習の多くでは、マクロをレイアウトしただけで終わりにしてしまうが、実際にテープアウトしてみると、その設計時間の大半はマクロの組み上げ、I/O 配置、マクロ接合部のエラー等トップ階層に費やされる。したがって、このステップおよび最後の Calibre 四天王を体験しておくことは、テープアウトのために絶対に必要である。

1 POCOP_TOP の Verilog 記述

Astro 版同様、トップ階層の Verilog 記述には、I/O と信号の接続を定義する必要がある。ここでは、入力と出力にそれぞれ以下の I/O を用いている。実はこれ以外使ったことがないので、他はどうだか分からないのだが、CMOS 3.3V 入出力の標準的なものである。

```
IOCB2EITNNMXA02 PAD_IDATAIN_00 (
    .EA(IO_IDATAIN[0]              ),
    .X (IDATAIN[0]                 ));



IOCB2EOT2X2LA02 PAD_DDATAOUT_00 (
    .A (DDATAOUT[0]          ),
    .EX(IO_DDATAOUT[0]       ));
```

この場合、IDATAIN, DDATOUT がマクロへの接続、IO_が付く方が PAD つまりチップの外部への接続に相当する。最上位階層は、このまま配線のみなので、今回は sdc は使わない。使う場合は適当に作ってやれば良い。

2 POCOP_TOP.tcl の解説

icc_shell -gui を立ち上げて、source POCOP_TOP.tcl とやると以下が実行され、POCOP_TOP のレイアウトが生成される。これを順に解説する。

2.1 初期設定

最初は環境設定で、POCOP マクロ生成時と同じ。

```
lappend serach_path ./
source scripts/POCOP_TOP_set_sz.tcl
set mw_logic0_net VSS
```

```
set mw_logic1_net VDD
define_name_rules verilog -allowed "A-Z0-9_"
```

次にライブラリを作って open する。先に作ったマクロもこれに入れておく。一度作ったら、後は open するだけでいい。

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

2.2 Verilog 読み込み

POCOP マクロの時と同様に、Verilog 記述、TLU を読み込み、電源、グランド、0,1 レベルを接続し、sdc を読み込む。

```
import_designs vnet/POCOP_TOP.vnet -format verilog -top POCOP_TOP
set_tlu_plus_files -max_tluplus ./lib/tlu_plus_mfe.worst \
        -min_tluplus ./lib/tlu_plus_mfe.best \
        -tech2itf_map ./lib/tlu2mw.map
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
read_sdc ./sdc/POCOP_TOP.sdc
```

これでレイアウトウインドウが生成される。

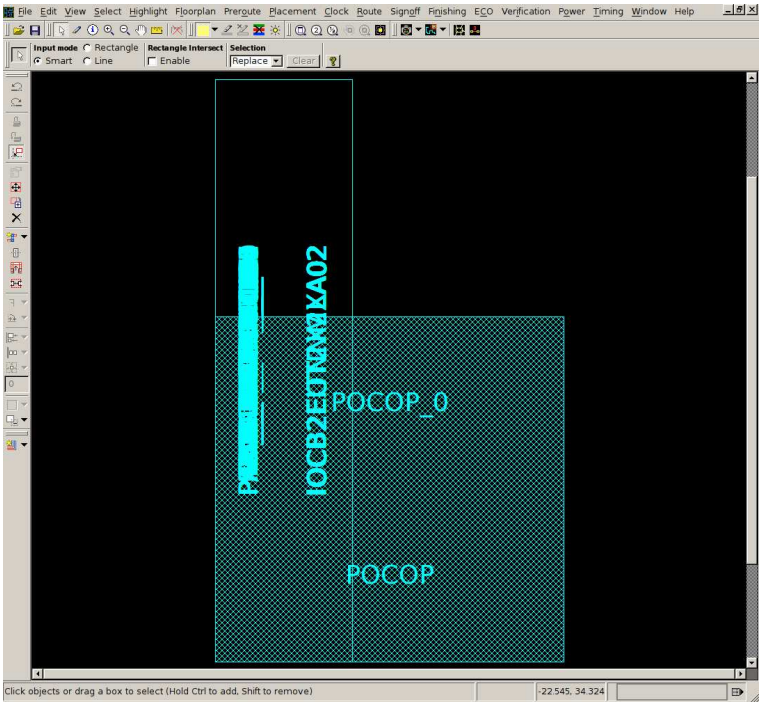

図 1: 読み込んだ後

2.3 フロアプラン

次にフロアプランをする。ここで、コアの大きさ、周辺の余裕を定義する。この大きさは Astro 版と同じにしたのだが、なぜか高さが若干足りず、0.6 だけ補正してある。境界が I/O の縁と合わないと後でトラブルがある。

```
initialize_floorplan -control_type width_and_height \
        -core_width 3420 -core_height 1320 \
        -left_io2core 28 -right_io2core 28 \
        -top_io2core 28.6 -bottom_io2core 28
```

これを実行すると、適当に並べてみてくれる。もちろん、I/O の配置はいい加減であるが、雰囲気が出る。POCOP は右上の箱である。

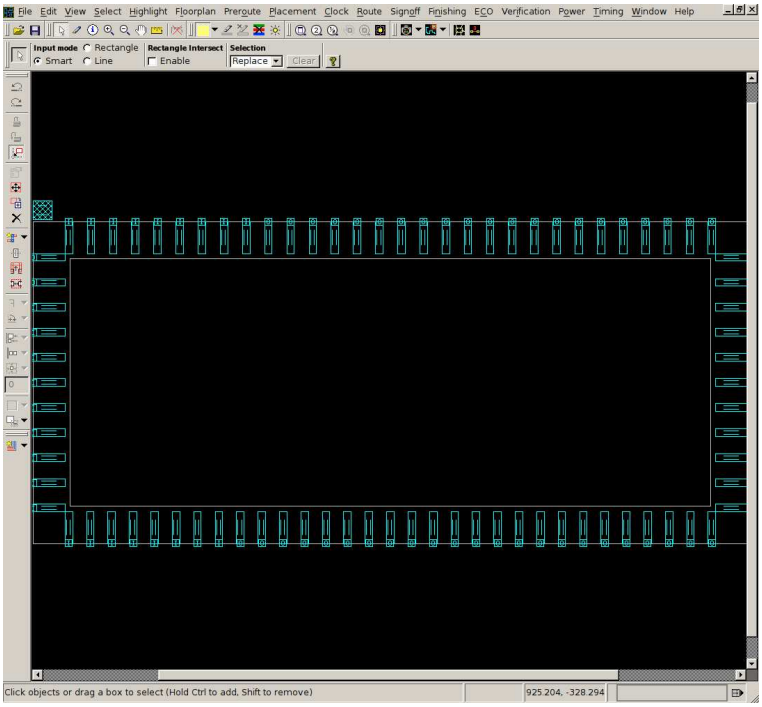

図 2: フロアプラン後

次に Verilog 記述に表れない I/O 関係の部品を生成する。

```
source ./scripts/POCOP_TOP_iogen.tcl
```

生成する部品は、コーナーセル (ZCGCB2E4C0XXA1)、電源用 I/O(VDD 用:IOCB2EPD5PI11, VSS 用:IOCB2EPG5PB11)、VDE:つまり入出力 3.3V 用 (IOCB2EW3AOC0A1) である。

さらに信号線を含むすべての I/O にワイヤパッドが必要である。ワイヤパッドを物理的なパッドにぴったり重ねてやらないと、配線をしてくれない。(最初にテープアウトした際に、この情報を誰も教えてくれず、泣いた。)

VDE 用:IOCB2EW3AOC0A1、VDD 用:IOCB2EWDAOC0A1、VSS 用:IOCB2EWGAOC0A1 信号線用:IOCB2EWSAOC0A1

これを順に使う分だけ番号を付けて生成する。

```
create_cell PAD_CORNER_B_L ZCGCB2E4C0XXA1
create_cell PAD_CORNER_T_L ZCGCB2E4C0XXA1
create_cell PAD_CORNER_B_R ZCGCB2E4C0XXA1
create_cell PAD_CORNER_T_R ZCGCB2E4C0XXA1
```

```
create_cell PAD_VDE01 IOCB2EPE5PE11
create_cell WIRE_PAD_VDE01 IOCB2EW3AOC0A1
create_cell PAD_VDE02 IOCB2EPE5PE11
create_cell WIRE_PAD_VDE02 IOCB2EW3AOC0A1
create_cell PAD_VDE03 IOCB2EPE5PE11
create_cell WIRE_PAD_VDE03 IOCB2EW3AOC0A1
create_cell PAD_VDE04 IOCB2EPE5PE11
create_cell WIRE_PAD_VDE04 IOCB2EW3AOC0A1
create_cell PAD_VDE05 IOCB2EPE5PE11
create_cell WIRE_PAD_VDE05 IOCB2EW3AOC0A1
create_cell PAD_VDE06 IOCB2EPE5PE11
create_cell WIRE_PAD_VDE06 IOCB2EW3AOC0A1
```
... 略

次に、これを配置する。
これが、

```
source ./scripts/POCOP_TOP_ioplace.tcl
```

である。
Astro の所でも書いたが、これは以下のルールを守る必要があり、非常に面倒である。

- コーナーセルの原点は、左隅にあり、例えば左下のコーナーセルとして使うためには、180 度回転して置く必要がある。他のコーナーセルも、回転角度を考えて右下は 270 度、右上は 0 度、左上は 90 度回転させて原点を考慮して置く。
- PAD の位置を考えると、I/O は 60um ピッチで置く必要がある。
- 電源、GND 用の PAD は横幅が 45um だが、信号線用は 40um である。
- I/O の原点は左下。この場所はチップ側 (内側) である。このため、チップの下部に配置する場合は 180 度、右側は 270 度、上部は 0 度、左側は 90 度回転させる必要がある。
- すべての I/O にはワイヤパッドを重ねる。

さて、この記述は、Astro とは全然違っていて、set 文を以下のように使って行う。

```
set obj [get_cells {"PAD_CORNER_B_L"} -all]
set_attribute -quiet $obj orientation S
set_attribute -quiet $obj origin {170 170}
set_attribute -quiet $obj is_placed true
set_attribute -quiet $obj is_fixed true

set obj [get_cells {"PAD_CORNER_B_R"} -all]
set_attribute -quiet $obj orientation E
set_attribute -quiet $obj origin {3646 170}
set_attribute -quiet $obj is_placed true
set_attribute -quiet $obj is_fixed true

set obj [get_cells {"PAD_CORNER_T_R"} -all]
```

```
set_attribute -quiet $obj orientation N
set_attribute -quiet $obj origin {3646 1546}
set_attribute -quiet $obj is_placed true
set_attribute -quiet $obj is_fixed true

set obj [get_cells {"PAD_CORNER_T_L"} -all]
set_attribute -quiet $obj orientation W
set_attribute -quiet $obj origin {170 1546}
set_attribute -quiet $obj is_placed true
set_attribute -quiet $obj is_fixed true

set obj [get_cells {"PAD_CORNER_B_R"} -all]
set_attribute -quiet $obj orientation 270
set_attribute -quiet $obj origin {3646 170}
set_attribute -quiet $obj is_placed true
set_attribute -quiet $obj is_fixed true

set obj [get_cells {"PAD_VDE01"} -all]
set_attribute -quiet $obj orientation S
set_attribute -quiet $obj origin {223 170}
set_attribute -quiet $obj is_placed true
set_attribute -quiet $obj is_fixed true

set obj [get_cells {"WIRE_PAD_VDE01"} -all]
set_attribute -quiet $obj orientation S
set_attribute -quiet $obj origin {223 170}
set_attribute -quiet $obj is_placed true
set_attribute -quiet $obj is_fixed true

set obj [get_cells {"PAD_VSS_01"} -all]
set_attribute -quiet $obj orientation S
set_attribute -quiet $obj origin {283 170}
set_attribute -quiet $obj is_placed true
set_attribute -quiet $obj is_fixed true
```
... 省略

この orientation が分からず、参った。特に小林フローでは FS とかが使われていて、何のことかわからなかった。1 時間程いろいろやってみた結果、F は Flip(反転) であることがわかった。通常反転する必要はないので、

N: 0 度、W:90 度、S:180 度、E:270 度に相当するということがわかった。これで I/O を配置した結果が図のようになる。

ところどころ歯抜けがあるが、これは電源等の一部をテストで取り外して面倒になって入れていない部分である。実害はない。気持ち悪ければ入れて下さい。

さて、次にパドフィラーを入れてこれらをくっつけてやる。間隔がまちまちなので、色々なサイズを指定している。

```
insert_pad_filler -cell {IOCB2ESE4C0A1 IOCB2ESD4C0A1 IOCB2ESA4C0A1 IOCB2ES54C0A1 \
        IOCB2ES24C0A1 IOCB2ES14C0A1 IOCB2ESB4C0A1}
```

次にマクロを配置する。

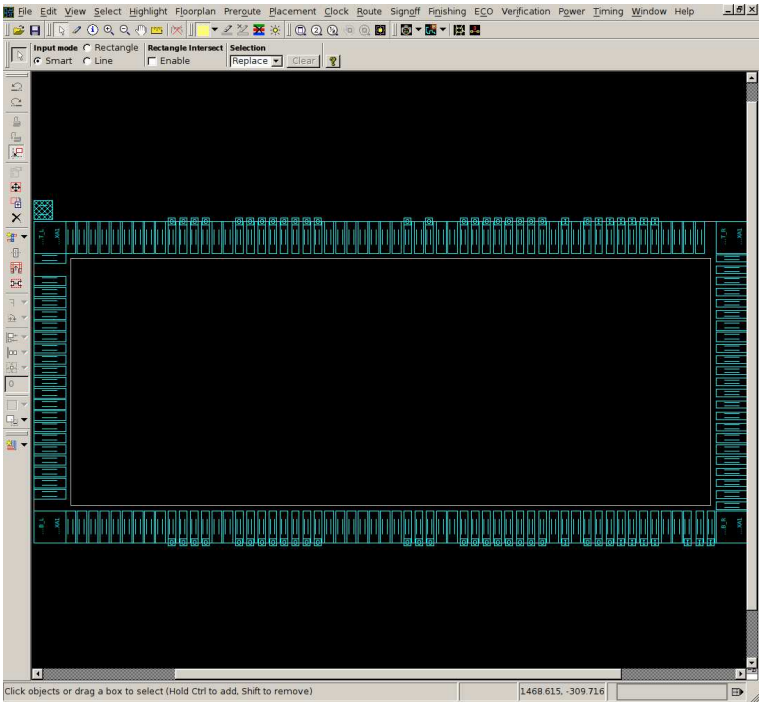

図 3: I/O の配置

```
source ./scripts/POCOP_TOP_place_macro.tcl
```

中身は簡単で、、

```
set obj [get_cells {"POCOP_0"} -all]
set_attribute -quiet $obj orientation N
set_attribute -quiet $obj origin {1853 801.1}
set_attribute -quiet $obj is_placed true
set_attribute -quiet $obj is_fixed false
set_attribute -quiet $obj is_soft_fixed false
set_attribute -quiet $obj eco_status eco_reset
```

I/O の配置と同様である。ここまでの結果は以下のようになる。
次に例によって電源、グランドを繋ぐ。ここでは入出力用の 3.3V(VDE) も繋ぐ。

```
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
derive_pg_connection -power_net VDE -power_pin VDE
```

2.4 電源リング、ストラップ

ここで電源リングを作る。メタルの利用は Astro と同じにした。サイズはコア周辺に収まるように決めた。

```
create_power_straps  -direction vertical  -start_at 243.8 -num_placement_strap 232 \
```

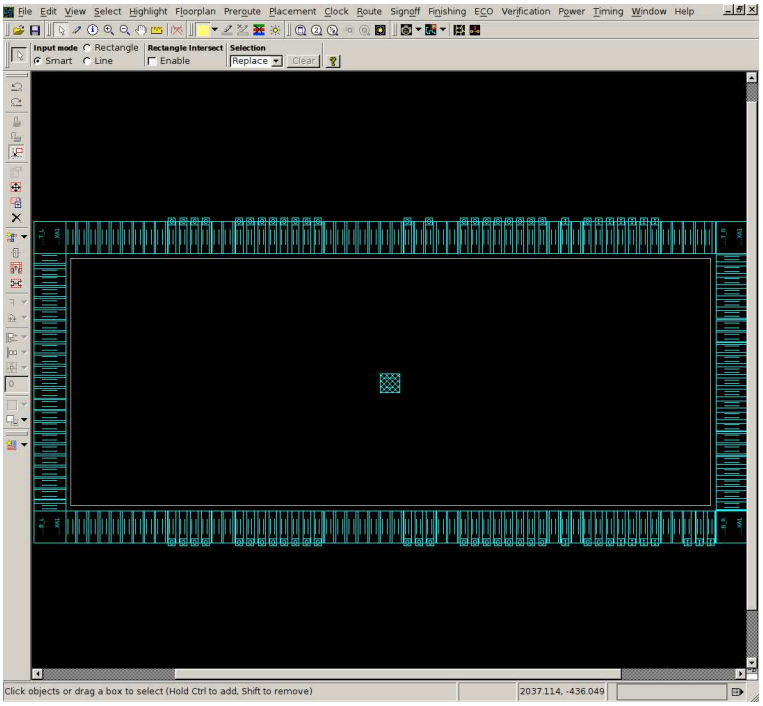

図 4: パドフィラーの挿入とマクロの配置

```
-increment_x_or_y 14.4 -nets  {VDD VSS}  -layer MET6 -width 1.8 \
-start_low_ends coordinate  -start_low_ends_coordinate 174.4 \
-start_high_ends coordinate  -start_high_ends_coordinate 1539.4 \
-extend_low_ends off -extend_high_ends off
```

ちなみに、このコマンドは-undo が効くので、気に入らなかったらやり直して調整が可能である。次に周辺の I/O を繋いでやる。

```
preroute_instances -ignore_macros -ignore_cover_cells
```

次はストラップである。今回はマクロのストラップとピッチを合わせて、これがつながるように配置した。途中うまくつながらなかったので、マクロをずらした。ぴったり合わせるとつながってくれる。

```
create_power_straps  -direction vertical  -start_at 243.8 -num_placement_strap 232 \
-increment_x_or_y 14.4 -nets  {VDD VSS}  -layer MET6 -width 1.8 \
-start_low_ends coordinate  -start_low_ends_coordinate 174.4 \
-start_high_ends coordinate  -start_high_ends_coordinate 1539.4 \
-extend_low_ends off -extend_high_ends off
```

これが下の図である。

ちなみに、マクロには気持ちよくつながるのだが、電源リングでははみ出してしまう。これは実害はないのだが、ちょっと格好悪い。しかし、オプションをいじってもうまくいかなかった。できる人は教えて下さいませ。

レールを作る前に、タップを配置する。その後、電源とグランドを接続するが、ここでも実入りはなかった。

```
add_tap_cell_array -master_cell_name {SC23YUZTAP021} -distance 36.0 \
-skip_fixed_cells false -no_tap_cell_under_layer {M1 M2} \
-connect_power_name {VDD} -connect_ground_name {VSS}
```

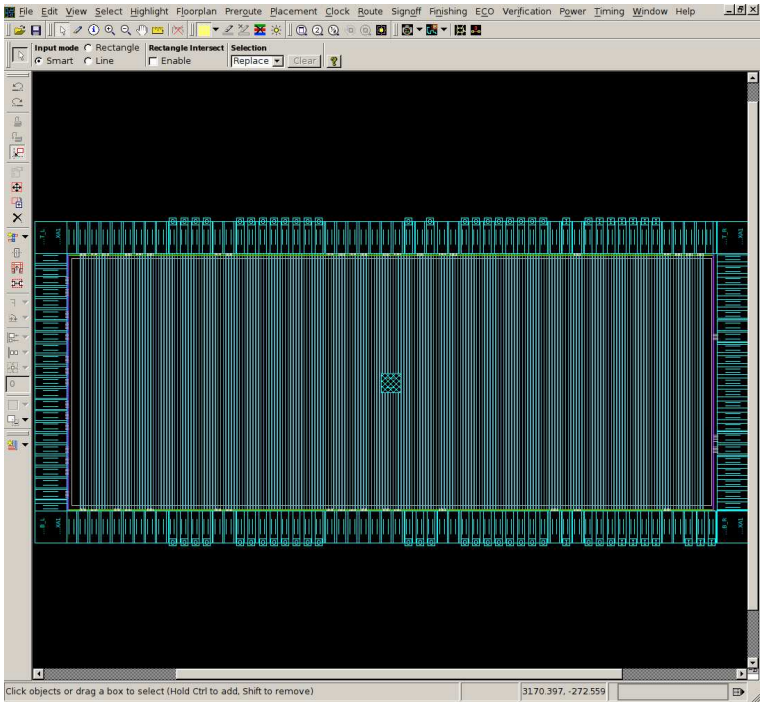

図 5: ストラップ

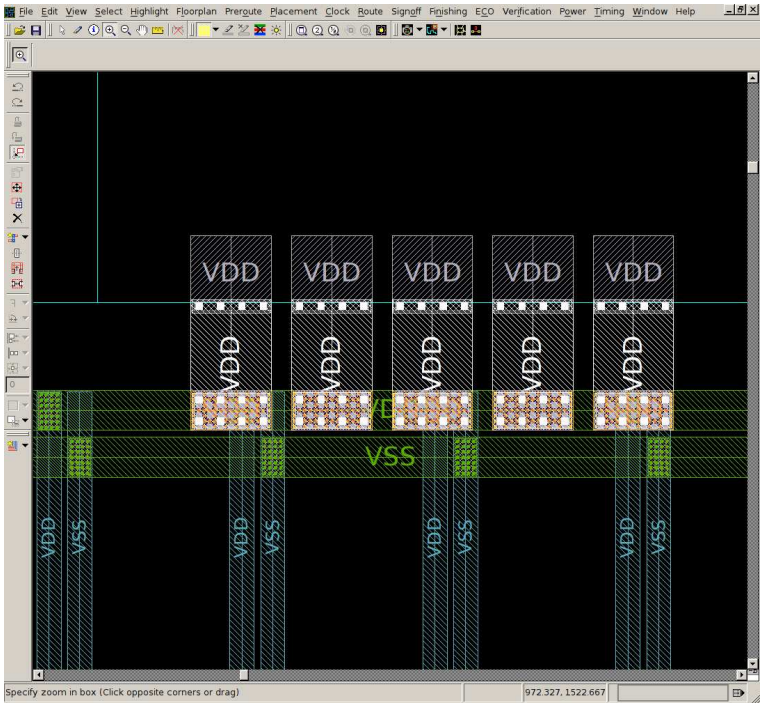

図 6: 格好悪いはじっこ

```
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
```

次にレールを引いた。

```
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
```

ストラップと違って、レールはマクロ内とつながらない。つながった方が良いかと思って位置合わせをやってみたがダメだった。考えてみると、これだけ太いストラップがあるので中と無理につなげる必要はないように思う。

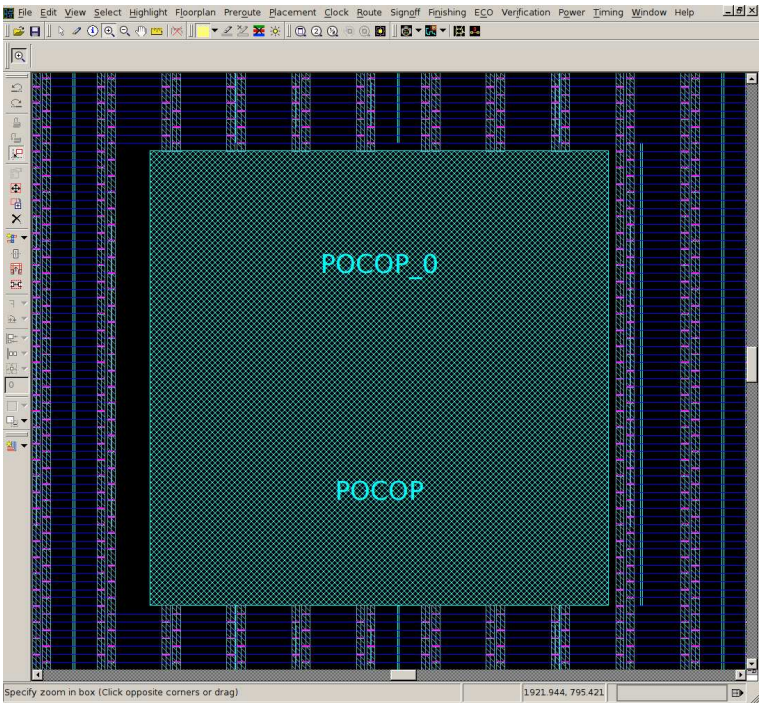

図 7: レールの様子

2.5 配線

次は、アンテナ回避および配線準備である。

```
source scripts/antenna_rule.tcl
set_parameter -name doAntennaConx -value 4
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
set_ignored_layers -max_routing_layer METG2
```

POCOP 同様、12 層は使っていない。これは 12 層を使うと Astro が落ちたためで、ICC なら大丈夫かもしれない。勇敢な君はこれを使わずやってみよう。

で、自動配線をやって、修復を 20 回位やり、アンテナ対策をやってさらに修復を 20 回くらいやる。最後に電源、グランドの接続をやるが、今回も実入りはなかった。

```
route_auto
route_search_repair -rerun_drc \
        -trim_antenna_of_user_wires -loop 20
verify_route -antenna
route_search_repair -rerun_drc \
        -trim_antenna_of_user_wires -loop 20
derive_pg_connection -power_net VDD \
        -ground_net VSS -power_pin VDD -ground_pin VSS
```

2.6 仕上げ

最後に、フィラーを入れる。全面 CUBA にして、一定の間隔で残して残りを消し、それから YUZS で埋めるというテクニックが未熟なため、使えず、お任せモードでやってしまっている。これだと、ほとんど CUBA が入ってしまう。

```
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
```

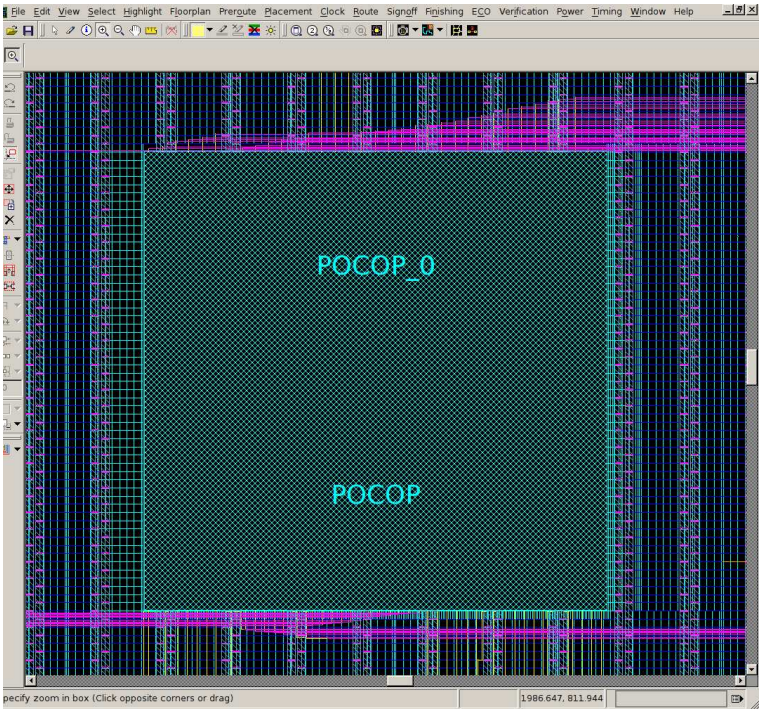

図 8: 最終レイアウト

次にフレームを重ねるために、原点を移動する。

```
move_mw_cel_origin -to {1908 858} [current_mw_cel]
```

最後に必ずセルを保存してから、ストリームアウトする。さらに LVS 用の Verilog 記述を保存する。

```
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_TOP_out.ddc
write_verilog -no_physical_only_cells POCOP_TOP_out.v
write_verilog -pg POCOP_TOP_lvs.v
write_stream -format gds -lib_name POCOP_TOP -cells {POCOP_TOP} POCOP_TOP.gds
write_sdf -version 2.1 POCOP_TOP.sdf
```

後は、Calibre 四天王を参照のこと。今回の設計では、以下の問題が発生した。

- ストラップがなかなかつながらず苦労した。Astro 同様、ウインドウレベルを上げるとマクロの中身が透けて見えるのだが、Astro 以上に見にくい。場所合わせをちゃんとやった所、うまく行った。ストラップやレール等のコマンドには-undo がある。これを利用すると何度もやりなおしが楽にできて便利。(すべてのコマンドに undo があるわけではない)
- I/O セルの配置が、設定した境界に微妙に合わなかった。この場合、I/O と認識されず PAD フィラーを入れてくれない。無理やり合わせたが理由は不明で不気味だ。
- コアのバウンダリ設定を間違い、I/O セルの周辺まで tap を入れてしまった。この状態で、レールを引いたり配線をしようとすると、いきなり icc が落ちてしまう。これは、Centos だと落ちない、とか、きむにいの環境だと落ちずに、ふんがだと落ちる、等色々不思議なことがあったが、バウンダリの設定が適切ならば大丈夫だった。これが結局一番時間を食った。
- 四天王は、LVS 以外は一発で通った。DRC は通るとは思わなかったがさすが ICC。ANT はダイオードを自動で入れてくれる効果が出ている。しかし、その分 LVS で十分てこずった。
- LVS を掛けるための write_verilog がダイオードの接続端子を抽出してくれない。

  ```
  SC23DS01 DS_RST (.VDD ( VDD ) , .VSS ( VSS ) ) ;
  ```

  になってしまう。ここはもちろん以下にならなければならない。

  ```
  SC23DS01 DS_RST (.A(RST_N) .VDD ( VDD ) , .VSS ( VSS ) ) ;
  ```

  レイアウト上は接続されているため、ダイオードについては LVS エラーを生じる。このダイオードは ICC が自動的に入れてくれたものなのになぜこのようになるのだろう？幸い、名前で端子名がわかるので、手で入れて LVS を通すことができた。
- LVS には他にも謎が多い。空の subckt が足りないモジュール (IOCB2ESE4C0A1) があって NOT COMPARED になってしまう。これを吐き出すために、-empty_module を付けるのだが、なぜかトップ階層では通用しない。マクロレベルでは出てくるが、トップでしか使わないモジュールなのでうまく行かない。やむなくこれも手で補った。また、Astro 同様 cdl 内で VDD と VDE をコネクトしている記述があり、これを消去した。

Astro のデザインフローと見比べてもらうと分かるが、ICC は断然優れている。ただ、細かいコマンドが分かっておらず、Layout 画面での操作の経験が乏しいので、この辺を深めないといけない。また、VDEC の IC Compiler 講習会に出てくる ICCompiler ならではの、フロアプランや電源見積り機能を使っておらず、この辺さらに極める必要がある。