How to Layout Top module using SOC Encounter with 40nm

Masayuki KIMURA

平成 23 年 3 月 29 日

1 はじめに

本稿は，Renesas-40nm プロセスを用いてトップレベルモジュールをレイアウトするための手順を示したものである．SLD-2 の SMA_2_TOP 用いた場合を説明する．本モジュールは，以下の SRAM マクロを含んでいる．

- WDREG110PAA128W14C1.LEFLIB
- WDSRAM002PAA512W25C2.LEFLIB

2 準備

2.1 必要なファイル

必要なファイル群は次のとおりである．

lef	トップ lef ファイル
lef_proute	電源配線用 lef ファイル
WDREG110PAA128W14C1.LEFLIB	メモリ用 lef ファイル
WDSRAM002PAA512W25C2.LEFLIB	メモリ用 lef ファイル
SMA_2_TOP.v	論理合成済みネットリスト
SMA_2_TOP.sdc	論理合成後 sdc

2.2 ディレクトリ構成

作業ディレクトリの構成は次のようになっている．

```
$ ls
 FECTS_saveDIR  Makefile  Makefile~  conf  lib  script  sdc  tdf  vnet
```

FECTS_saveDIR クロックの情報ディレクトリ

conf VDEC_soce.conf が入っている

lib lef ファイルが入っている

script SOC Encounter 用のスクリプトが入っている

sdc 論理合成後の sdc が入っている (SMA_2_TOP.sdc)

vnet ネットリストが入っている (SMA_2_TOP-compile.v)

3 設定読み込みからフロアプランニングまで

SOC Encounter を立ち上げる．

3.1 conf/VDEC_soce.conf

まず，各種設定用のファイルを読み込む.

```
source "./conf/VDEC_soce.conf"
```

VDEC_soce.conf では，ライブラリの場所などの基本的な設定を行っている．これは必ず読み込ませる.

VDEC_soce.conf(抜粋)

```
...
set OPC_PATH          /home/vdec/lib/ux8l
set OPC_ADD           /home/vdec/lib/ux8l/IO/OPC_ADD2
set OPC_PLL           /home/vdec/lib/ux8l/PLL/linux

set LibertyADD        $OPC_ADD/blib/UX8L/wide1_1.1V/liberty
set LibertyPLL        $OPC_PLL/blib/UX8L/wide1_1.1V/liberty
set LibertyMax        $OPC_PATH/blib/UX8L/wide1_1.1V/liberty/max
set LibertyMin        $OPC_PATH/blib/UX8L/wide1_1.1V/liberty/min
set capTable          $OPC_PATH/lib/UX8L/wide1_1.1V/soce/captable/UX8L_7L.captable
...
```

3.2 script/load_design.tcl

デザインを読み込む.

script/load_design.tcl

```
global loadLEF

source "./script/var.tcl"
setUIVar rda_Input ui_timingcon_file sdc/${DESIGN}.sdc
# setUIVar rda_Input ui_leffile ${lib_file}
setUIVar rda_Input ui_leffile $loadLEF
setUIVar rda_Input ui_timelib [ list ${LibertyTyp}/UX8L_wide1_1.1V_TYP_primitive_hvt.lib \
    ${LibertyTyp}/UX8L_wide1_1.1V_TYP_primitive_mvt.lib ]
setUIVar rda_Input ui_netlist vnet/${DESIGN}-compile.v
setUIVar rda_Input ui_timelib,min [ list ${LibertyMin}/UX8L_wide1_1.1V_MIN_primitive_mvt.lib \
${LibertyMin}/UX8L_wide1_1.1V_MIN_primitive_hvt.lib ]
setUIVar rda_Input ui_timelib,max [ list ${LibertyMax}/UX8L_wide1_1.1V_MAX_primitive_mvt.lib \
${LibertyMax}/UX8L_wide1_1.1V_MAX_primitive_hvt.lib ]
setUIVar rda_Input ui_topcell ${DESIGN}
commitConfig

set designName [dbgDesignName]
```

3.3 script/floorplan_SMA_2_TOP.tcl

フロアプランを行う．フロアプランでは，マクロやパッドなどの配置を決める.

3.3.1 マクロを配置する際の注意事項

1. OnGrid 上に配置すること

これはマニュアルにも書いてある．OnGrid 上に配置されているかどうかは,

```
checkMacroLLOnTrack -useM2M3Track
```

で確認が可能である．ただ，Grid の表示方法が分からないので，ここは試行錯誤となる.

3.3.2 フロアプランニング

1. パッドを置く あらかじめ記述した csv ファイルにより，PAD の配置を行う．これには，`mkdef.tcl` を用いる．

```
#####################################################
# Set Input Parameter
#####################################################
set design         SMA_2_TOP
set pinassign_list ./pin/PAD_UX8L_SMA_2_TOP_50.csv
set pad_file       ./pin/chip_5000x5000_FPBGA_pad.list
set output_def    io.def.gz

# set chipsize 2.5
set chipsize 5.0
#set chipsize 5025
```

上記部分を適切に書き換えることにより，PAD の csv ファイルを読み込ませ，チップサイズを決める．

2. マクロを配置する `placeInstance` を利用する．

```
placeInstance SMA_0_CONF_CONST_CTRL_0  [expr 0.132 * -3935 + 0.066] [expr ${cell_height} * -950]   R0 -f
placeInstance SMA_0_CONF_CONST_CTRL_1  [expr 0.132 * -3935 + 0.066] [expr ${cell_height} *  1000]   R0 -
placeInstance SMA_0_DMEM_ACCESS_CTRL_0 [expr 0.132 *  5050  + 0.066] [expr ${cell_height} * -470]   R0 -
placeInstance SMA_0_DMEM_0             [expr 0.132 *  7870  + 0.066] [expr ${cell_height} * -420]   R0 -
placeInstance SMA_0_PE_ARRAY_0         [expr 0.132 * -14987 + 0.066] [expr ${cell_height} * -620]   R0 -
```

3. Row カットと stop セルを配置する 全てのマクロを検索し，カットを行い各マクロの境界 stop セルを配置する．

```
dbForEachCellIo [dbgTopCell] ioPtr {
    selectInst [dbInstName [dbIoInst $ioPtr]]
}
# cutRow -selected -leftGap 4.455 -rightGap 4.455 -bottomGap 4.455 -topGap 4.455
# cutRow -selected -leftGap 8.91 -rightGap 8.91 -bottomGap 8.91 -topGap 8.91
# cutRow -selected -leftGap 22.225 -rightGap 22.225  -bottomGap 22.225  -topGap 22.225
cutRow -selected -leftGap 22.225 -rightGap 22.225  -bottomGap 35.64  -topGap 35.64
deselectAll

selectInst SMA_0_PE_ARRAY_0
selectInst SMA_0_CONF_CONST_CTRL_0
selectInst SMA_0_CONF_CONST_CTRL_1
selectInst SMA_0_DMEM_ACCESS_CTRL_0
selectInst SMA_0_DMEM_0
cutRow -selected -topGap ${powerGapH} -bottomGap ${powerGapH} -leftGap ${powerGapV} -rightGap ${powerGap

addEndCap -preCap LDL_POWERSTOPL -postCap LDL_POWERSTOPR -prefix LDL_POWERSTOP
```

3.4 フロアプランの結果

図 1，図 2，図 3 に，`./script/floorplan.tcl` を実行したときの結果を示す．

ここで，一旦 SOC Encounter を再起動する．これは，電源配線用に`./lib/lef_proute` を読み直す必要があるからである．

4 電源配線

自動レイアウトの中でもっとも注意を要するところである．ここを間違えないように注意する．

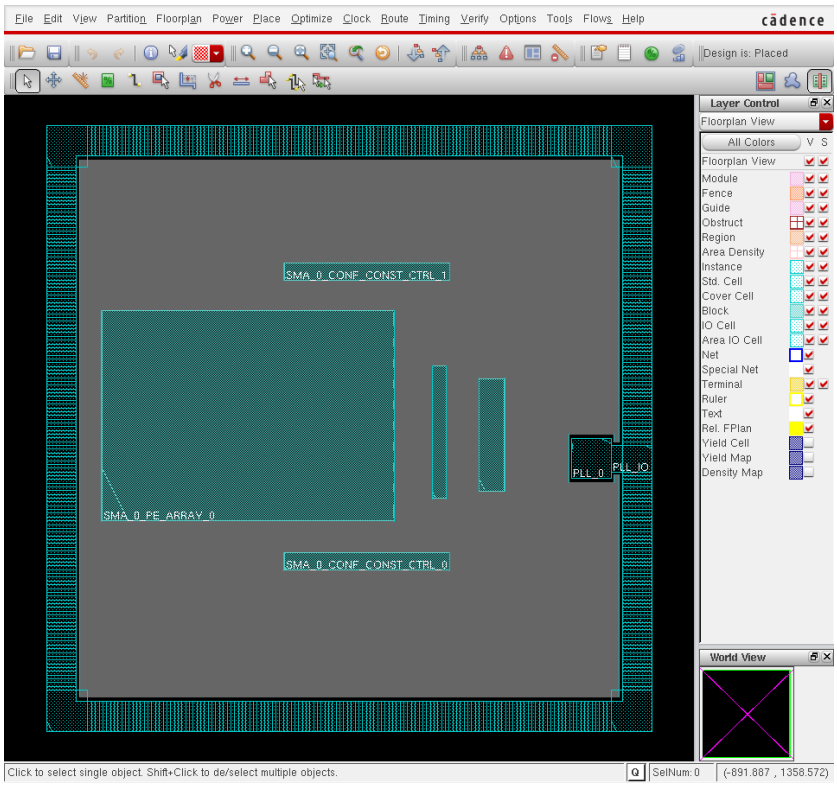

図 1: フロアプラン結果

4.1 ./script/snap.tcl レイアウト，ビアのスナップ

各操作の前に，snap.tcl を読み込ませて，OnGrid にレイアウトされるように注意する．
電源配線，配置配線の時には必ず source ./script/snap.tcl を実行して OnGrid レイアウトされるようにすること！

./script/snap.tcl

```
setSnapGrid -layer { 1 }               -pitch 0.011 0.011
setSnapGrid -layer { 2 3 4 5 }         -pitch 0.033 0.033
setSnapGrid -layer { V12 V23 V34 V45 } -pitch 0.066 0.066
```

4.2 前回デザイン読み込み

ここで，使用する lef ファイルを./lib/lef_proute に切り替える．

```
source "./conf/VDEC_soce.conf"
source "./script/load_design_proute.tcl"
```

./script/load_design_proute.tcl(抜粋)

```
...
set inputVerilog "./SMA_2_TOP_floorplan.v"
set inputSDC     "./sdc/SMA_2_TOP.sdc"
set inputDef     "./SMA_2_TOP_floorplan.def"

setUIVar rda_Input ui_timingcon_file $inputSDC
setUIVar rda_Input ui_leffile        { ./lib/lef_proute ./lib/WDREG110PAA32W16C1.LEFLIB }
...
```

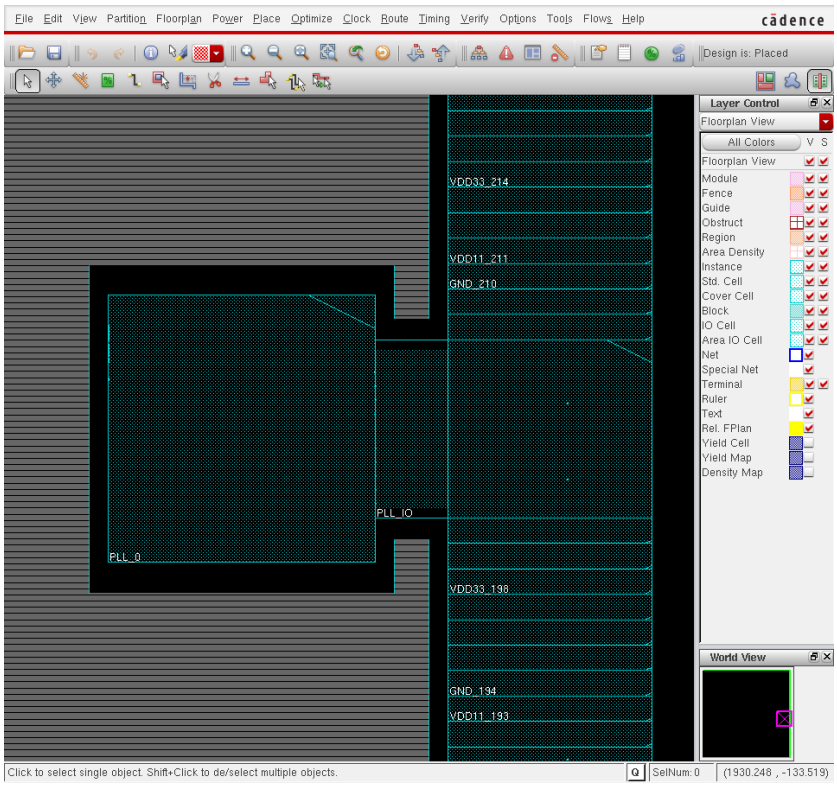

図 2: PLL フロアプラン結果

4.3 電源配線 tcl./script/proute_top.tcl

電源レールを張る．M1L 層と，M2L 層に張る必要がある．M2L 層を張らなかった場合，FILLER のいくつかで open のエラーが出てしまうので，かならず張ること．

./script/proute.tcl(抜粋)

```
# M2L へのレール
sroute -noBlockPins -noPadRings -noPadPins -noStripes \
     -crossoverViaTopLayer 5 \
     -jogControl { preferWithChanges differentLayer } \
     -nets { VDD VSS } -targetViaTopLayer 5 \
     -corePinLayer 2 \
     -extraConfig align.cfg
###

# M1L へのレール
sroute -noBlockPins -noPadRings -noPadPins -noStripes \
     -crossoverViaTopLayer 5 \
     -jogControl { preferWithChanges differentLayer } \
     -nets { VDD VSS } -targetViaTopLayer 5 \
     -corePinLayer 1 \
     -extraConfig align.cfg
```

確認事項

- M1L，M2L の両方に，まんべんなくレールが張られているか？
- マクロのまわりで，レールが止まっているか？

マクロに電源リングを巻くため，全てのマクロを選択する．

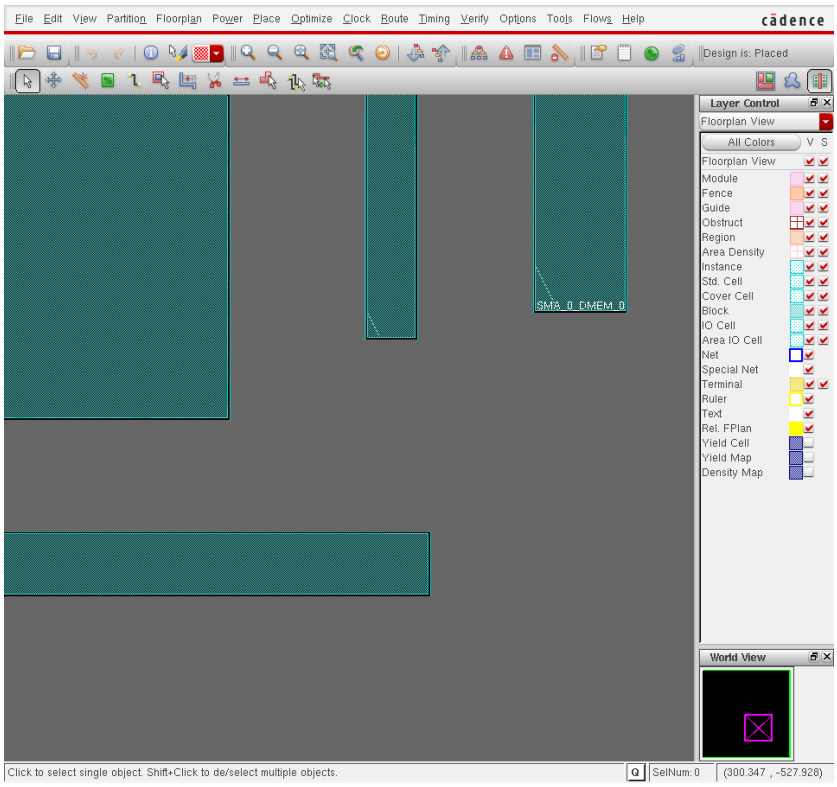

図 3: マクロフロアプラン結果

./script/proute_top.tcl(抜粋)

```
selectInst SMA_0_PE_ARRAY_0
selectInst SMA_0_CONF_CONST_CTRL_0
selectInst SMA_0_CONF_CONST_CTRL_1
selectInst SMA_0_DMEM_ACCESS_CTRL_0
selectInst SMA_0_DMEM_0
```

マクロに電源リングを巻く．リングは M5 層と M4 層に張る．-around each_block を設定することにより，メモリマクロの周囲にリングを張っている．

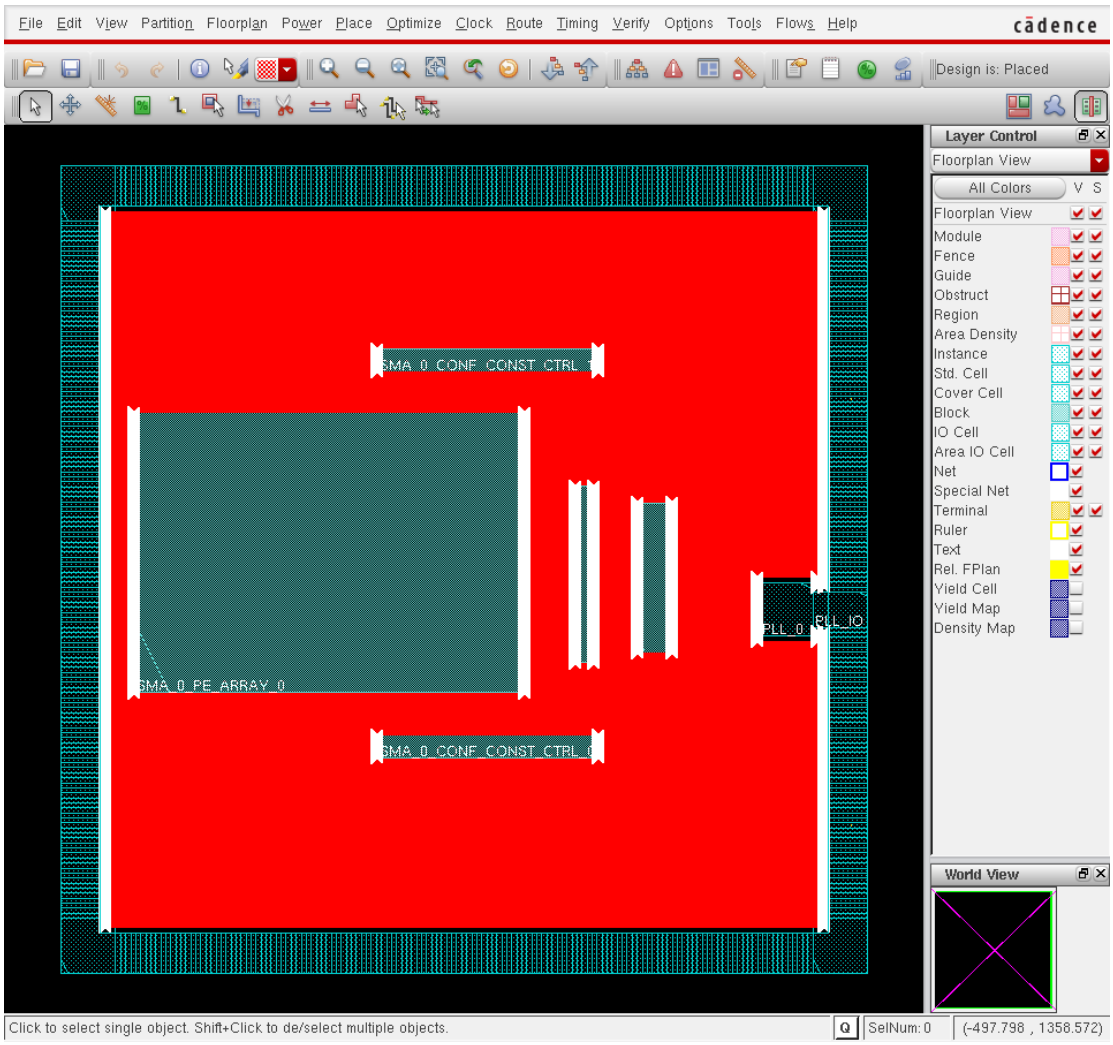

図 4: トップ階層へレールを張った結果

./script/proute_top.tcl(抜粋)

```
addRing \
    -spacing_top     [expr 0.132 * 6]\
    -spacing_bottom [expr 0.132 * 6]\
    -spacing_left    [expr 0.132 * 6]\
    -spacing_right   [expr 0.132 * 6]\
    -width_top       [expr 0.066 * 15] \
    -width_bottom    [expr 0.066 * 15] \
    -width_left      [expr 0.066 * 15] \
    -width_right     [expr 0.066 * 15] \
    -layer_top       M4L \
    -layer_bottom    M4L \
    -layer_left      M4L \
    -layer_right     M4L \
    -offset_top      [expr 0.165 * 6]\
    -offset_bottom   [expr 0.165 * 6]\
    -offset_left     [expr 0.264 * 6]\
    -offset_right    [expr 0.264 * 6]\
    -stacked_via_top_layer     M3L \
    -stacked_via_bottom_layer M3L \
    -snap_wire_center_to_grid Grid \
    -around          selected \
    -type            block_rings \
    -nets            {VDD VSS}
deselectAll
```

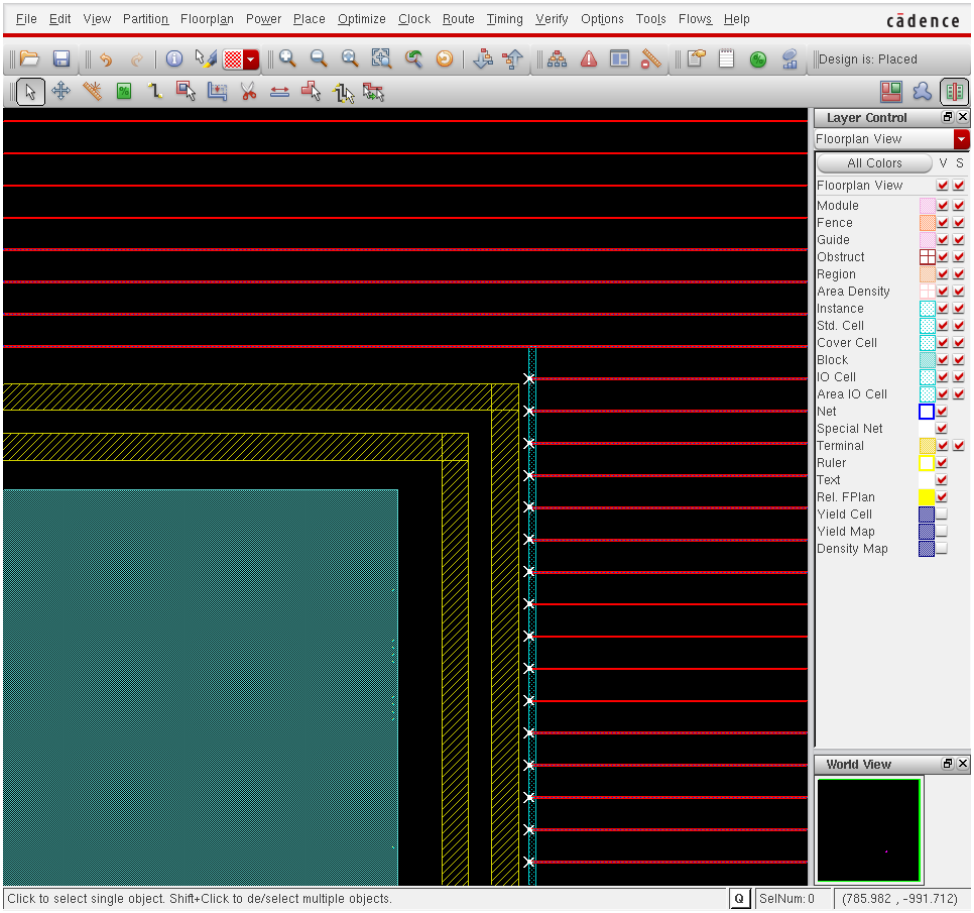

図 5: マクロに電源リングを張った結果

図 5 に，マクロに対してリングを張った結果を示す．
マクロの電源を接続する．

./script/proute_top.tcl(抜粋)

```
sroute -nets { VSS VDD } \
    -connect { blockPin floatingStripe} \
    -connectInsideArea \
    -layerChangeRange { 1 8 } \
    -blockPinTarget { nearestRingStripe nearestTarget } \
    -padPinPortConnect { allGeom } \
    -blockPin useLef \
    -allowJogging 1 \
    -crossoverViaTopLayer 8 \
    -crossoverViaBottomLayer 1 \
    -allowLayerChange 1 \
    -targetViaTopLayer 8 \
    -checkAlignedSecondaryPin 1 \
    -targetViaBottomLayer 1
```

図 6 に，マクロに対して電源を接続した結果を示す．
PAD の近くに VIA を打たれるとエラーになるので，blockage を張って VIA を防ぐ．

./script/pblockage.tcl(抜粋)

```
createRouteBlk -layer { 1 2 3 5 6 7 } -box -2181.828 -2190.796  2181.828 -2170.796
createRouteBlk -layer { 1 2 3 5 6 7 } -box -2181.828  2190.796  2181.828  2170.796
```

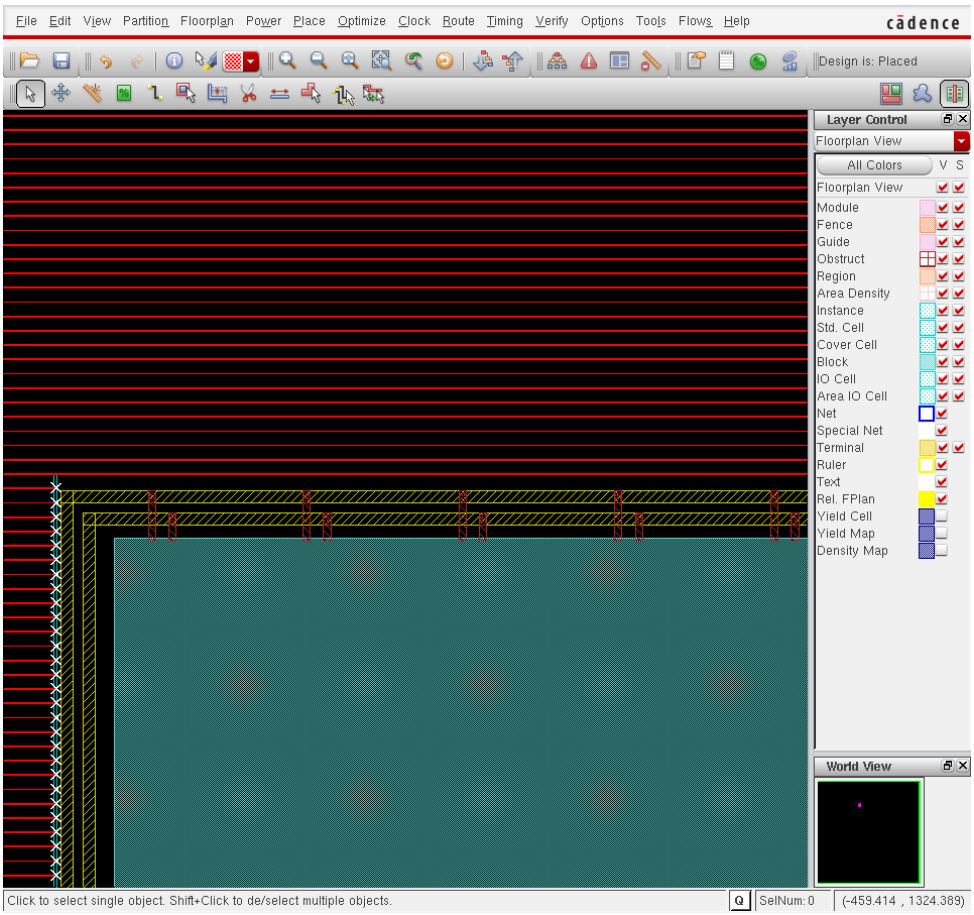

図 6: マクロに対し電源を接続

図 7, 図 8 に，ブロッケージを張った結果を示す.
トップ階層の電源リングを巻く．今回は 2 電源設計のため，VDD,VDDL,VSS の 3 つのリングを張る.

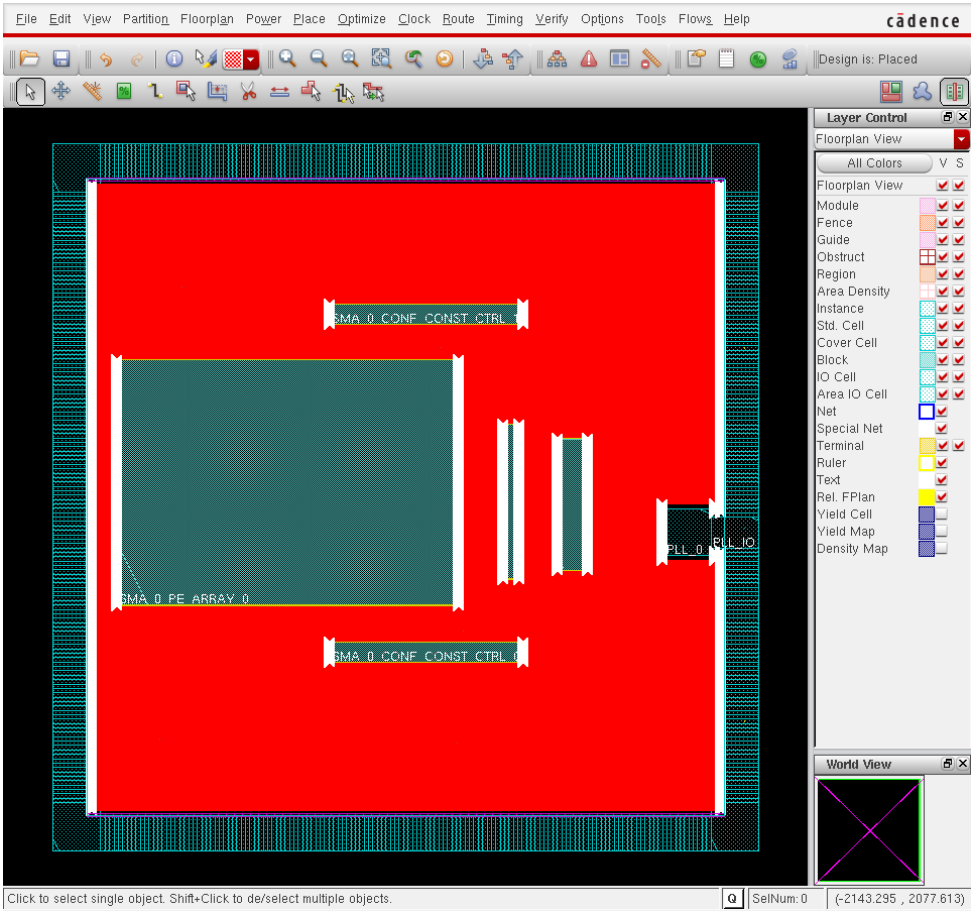

図 7: PAD の周りにブロッケージを張る (1)

./script/proute_top.tcl(抜粋)

```
addRing -nets { VDD VDDL VSS } \
    -io_offset 1 \
    -follow io \
    -layer_top ${hlayer} \
    -layer_bottom ${hlayer} \
    -width_top ${h_width} \
    -width_bottom ${h_width} \
    -width_right ${v_width} \
    -width_left ${v_width} \
    -layer_right ${vlayer} \
    -layer_left ${vlayer} \
    -spacing_top ${h_spacing} \
    -spacing_bottom ${h_spacing} \
    -spacing_right ${v_spacing} \
    -spacing_left ${v_spacing} \
    -offset_top ${h_offset} \
    -offset_bottom ${h_offset} \
    -offset_right ${v_offset} \
    -offset_left ${v_offset} \
    -stacked_via_top_layer M6S \
    -stacked_via_bottom_layer M1L \
    -snap_wire_center_to_grid Grid
```

図 9 に，リングを張った結果を示す．
トップ階層のストライプを張る．

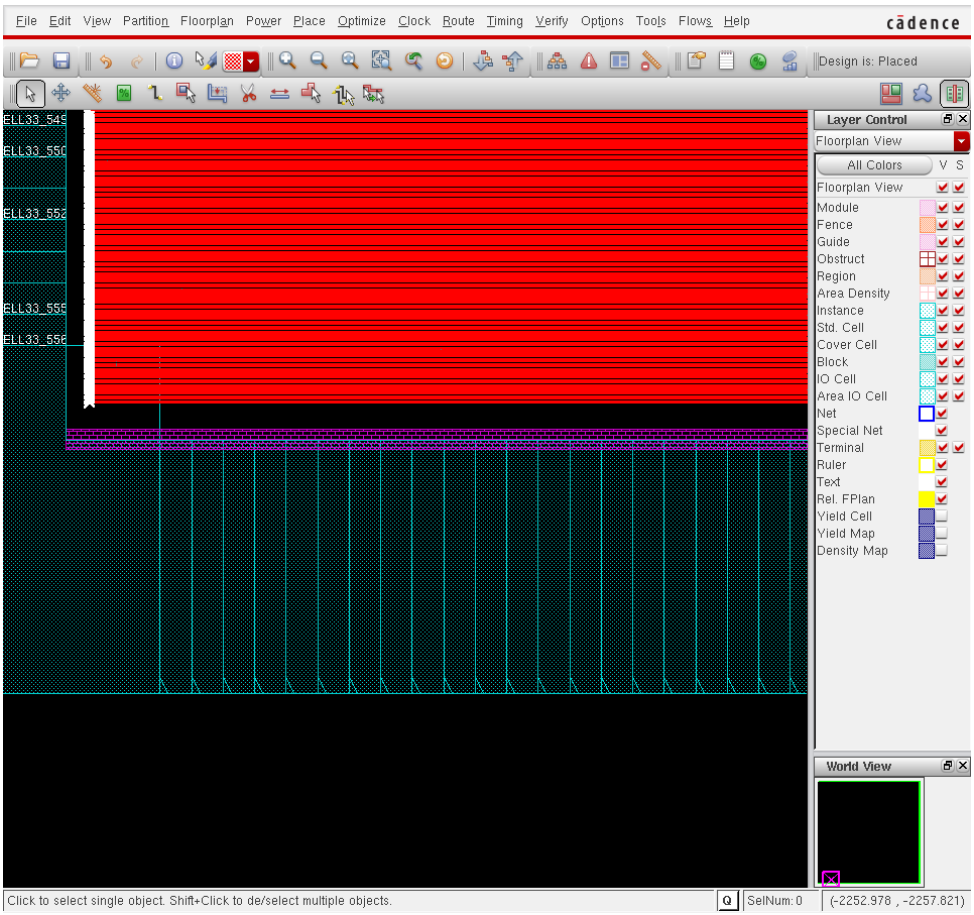

図 8: PAD の周りにブロッケージを張る (2)

```
deselectAll
setAddStripeOption -remove_floating_stripe_over_block 0
addStripe -nets {VSS VDD} \
    -layer M6S \
    -direction vertical \
    -spacing [expr 0.264 * 20] \
    -width 0.99 \
    -set_to_set_distance [expr 0.264 * 98] \
    -xleft_offset [expr 0.264 * 150] \
    -block_ring_top_layer_limit PM \
    -block_ring_bottom_layer_limit M1L \
    -padcore_ring_top_layer_limit PM \
    -padcore_ring_bottom_layer_limit M1L \
    -stacked_via_top_layer PM \
    -stacked_via_bottom_layer M1L \
    -merge_stripes_value [expr 0.264*5] \
    -snap_wire_center_to_grid Grid \
    -via_using_exact_crossover_size 1
```

図 10，図 11，図 12 に，トップ階層のストライプを張った結果を示す．
パッドの電源を接続する．

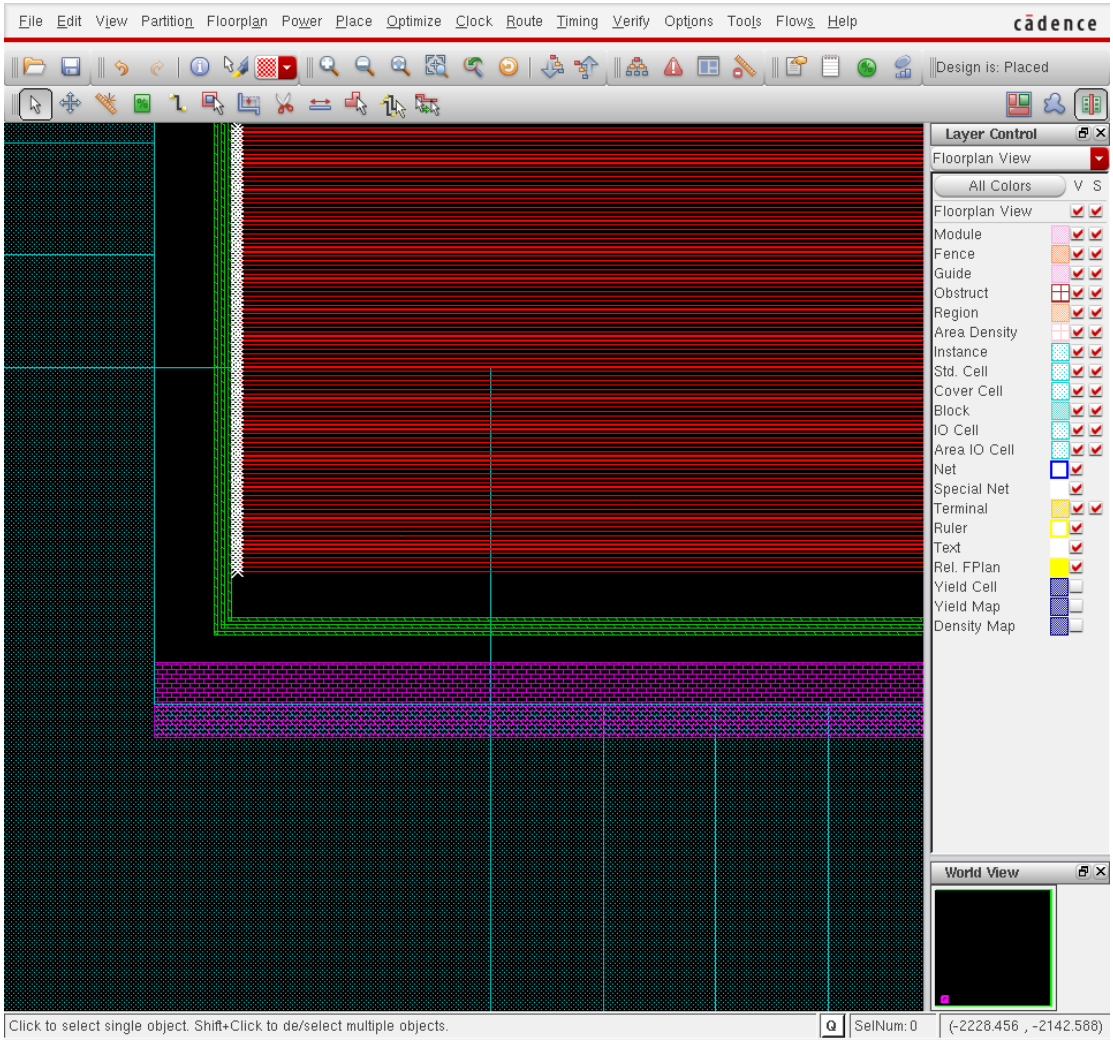

図 9: トップ階層にリングを張った結果 (一部)

```
set connectlayer 4
set con_name VSS
sroute -nets { VSS } \
    -connect { padPin } \
    -padPinAllGeomsConnect \
    -padPinLayerRange { 4 5 } \
    -padPinWidth 0.99 \
    -split_long_via {0.264 0.264 0.264}

set connectlayer 4
set con_name VDD
sroute -nets { VDD } \
        -connect { padPin } \
     -padPinAllGeomsConnect \
        -blockPinTarget { nearestTarget } \
        -padPinLayerRange { 4 5 } \
     -padPinWidth 0.99 \
    -split_long_via {0.264 0.264 0.264}
```

ここで，一旦 SOC Encounter を再起動する．これは，電源配線用の./lib/lef_proute から，普通の./lib/lef に切り替える必要があるからである．

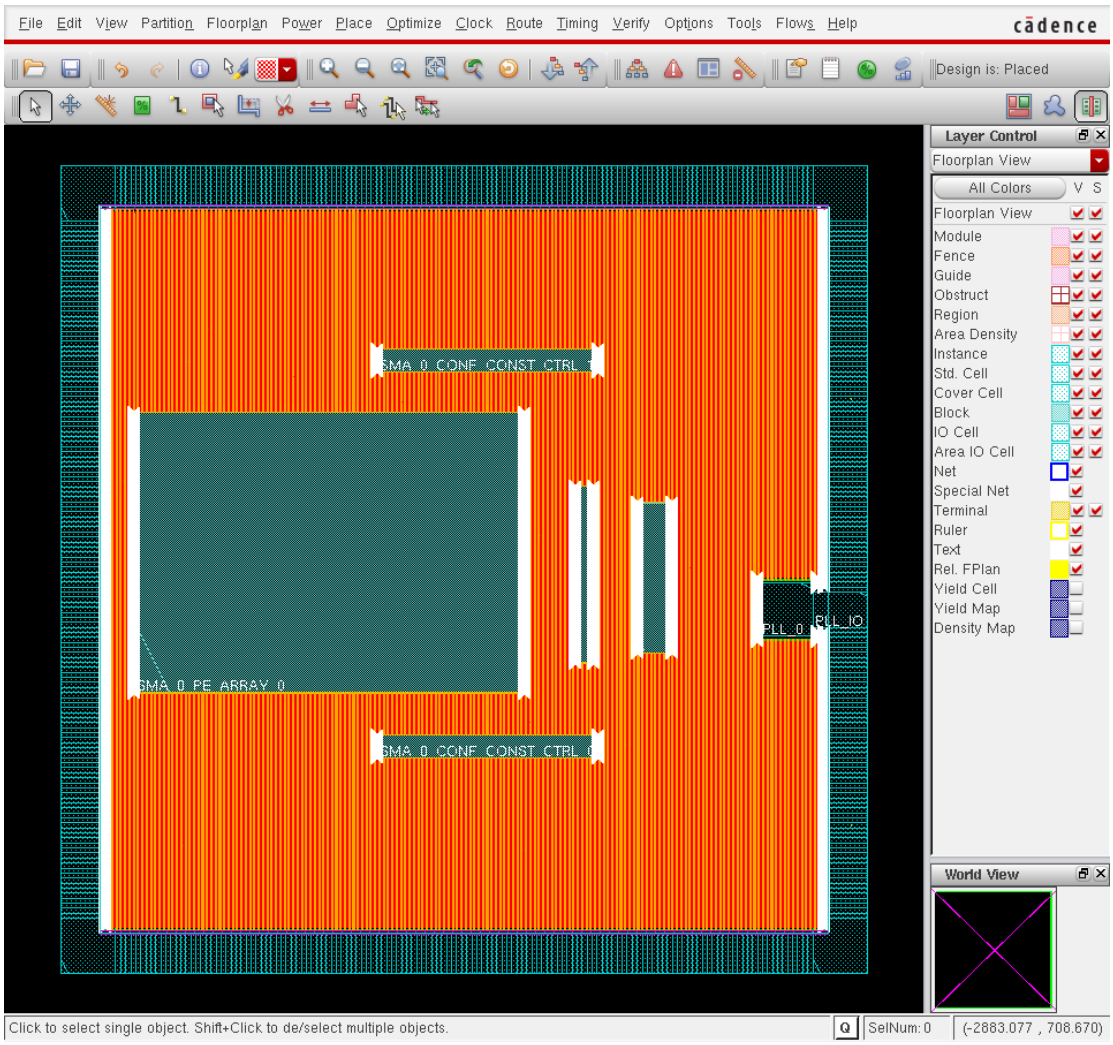

図 10: トップ階層にストライプを張った結果 (1)

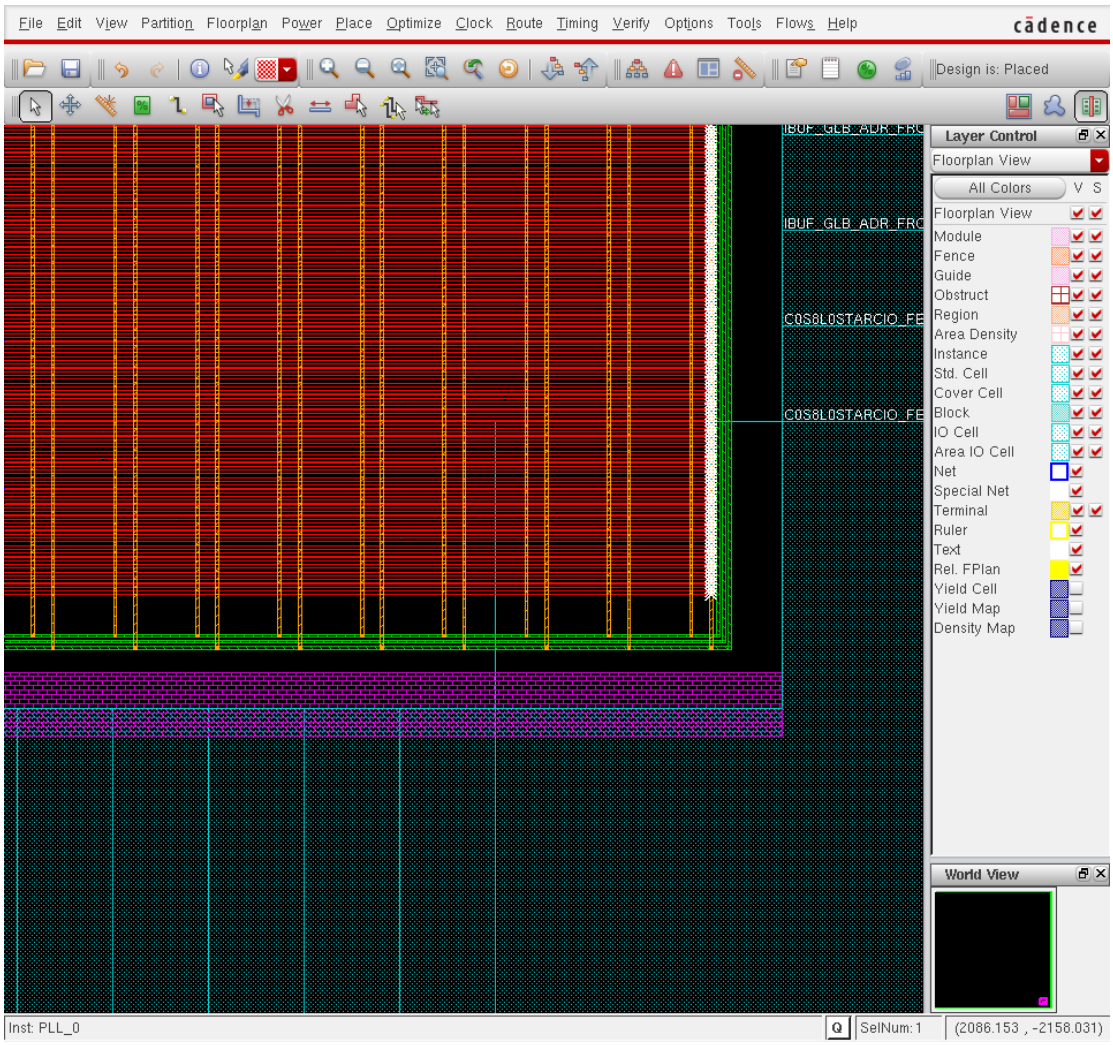

図 11: トップ階層にストライプを張った結果 (2)

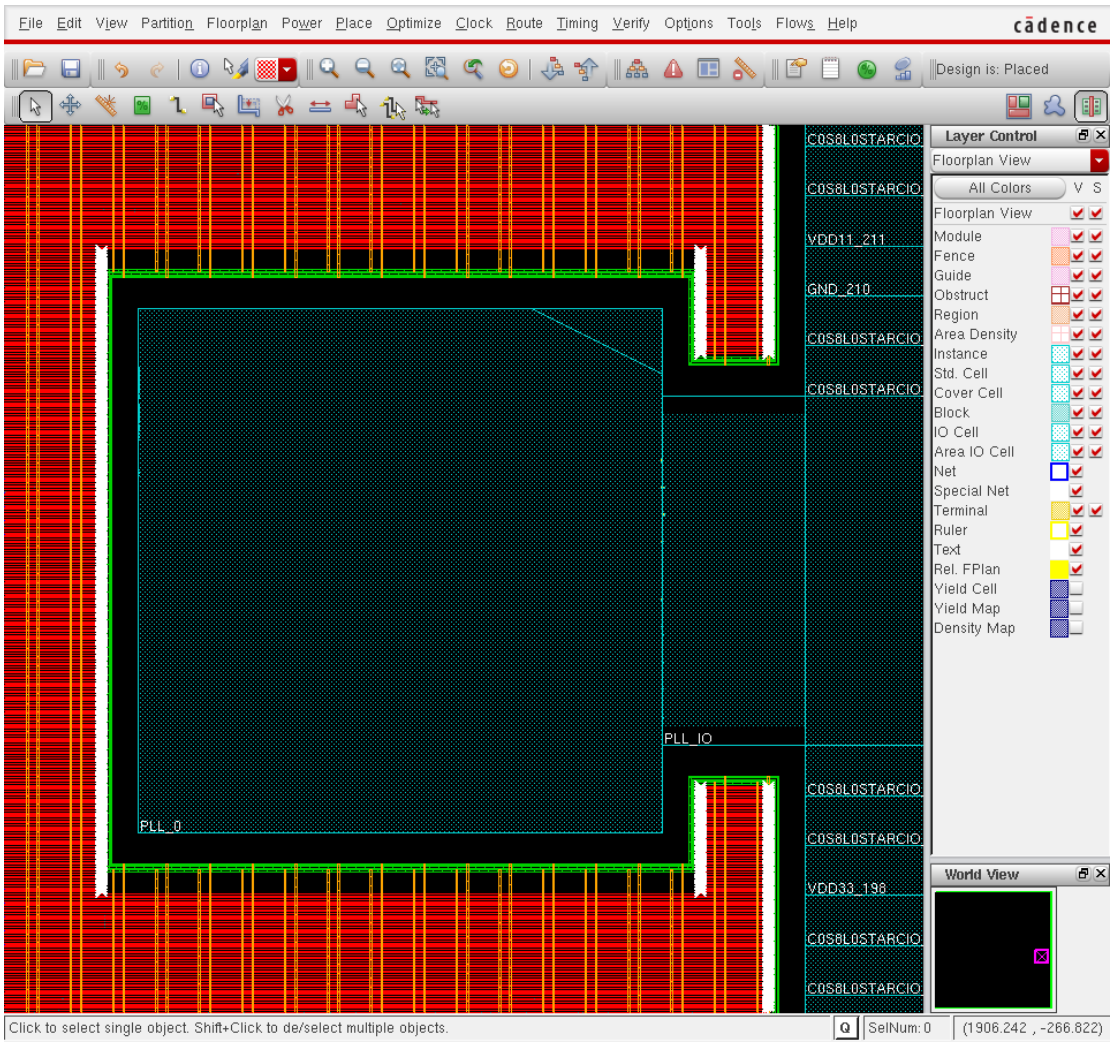

図 12: トップ階層にストライプを張った結果 (PLL)

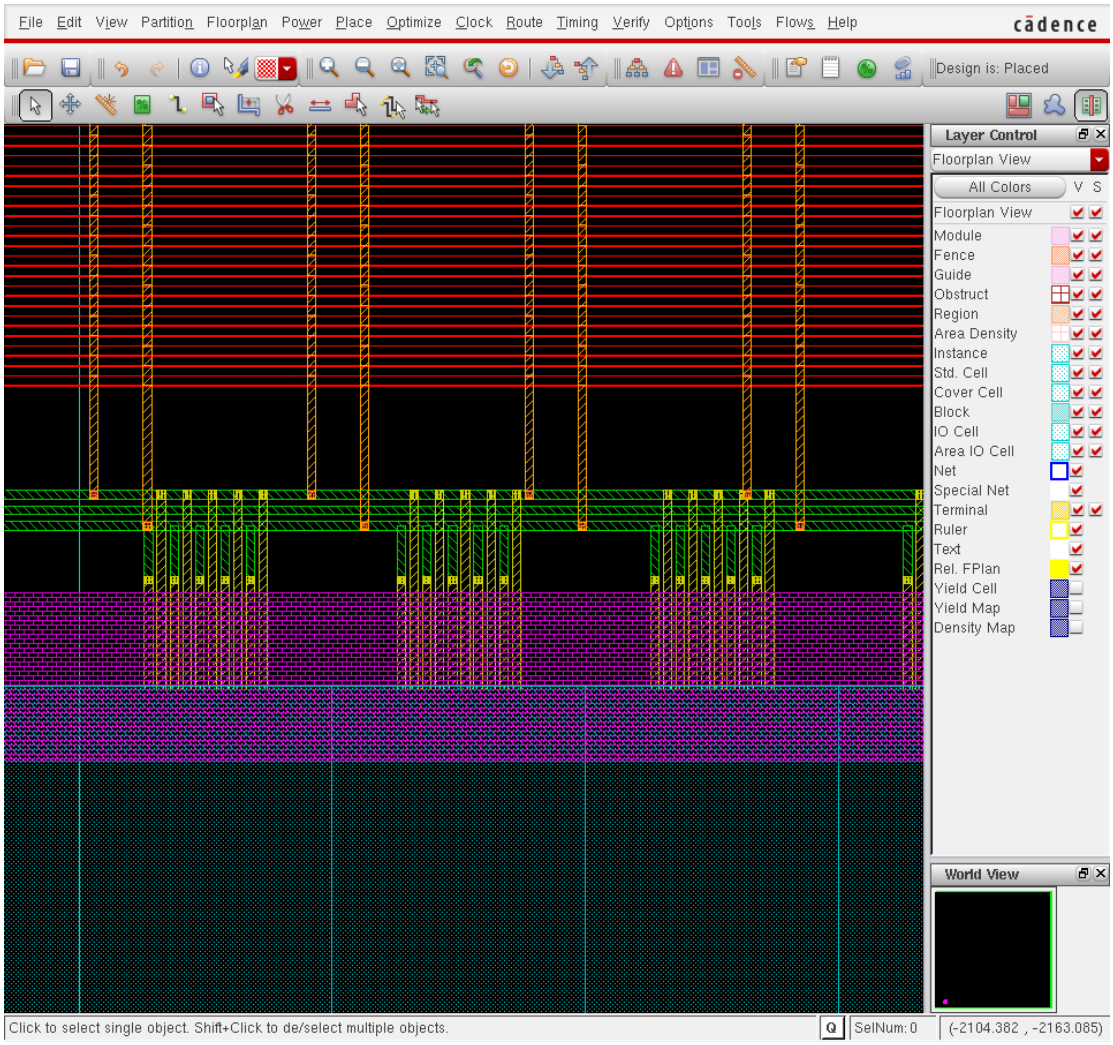

図 13: PAD に電源を接続した結果

図 14: ブロッケージの配置結果

5 配置配線

5.1 デザインの読みなおし

SOC Encounter を再起動したので，デザインを読み直す.

```
source "./conf/VDEC_soce.conf"
source "./script/load_design_pr.tcl"
```

5.2 ./script/blockage.tcl

配置禁止領域にブロッケージを設定する．ブロッケージを張る領域は大きく 3 種類に分けられる.

- 電源ストライプ領域
- マクロの境界領域

5.2.1 電源ストライプ領域

電源ストライプ領域にスタセルが置かれると，ショートの可能性があるため，あらかじめブロッケージを張ってスタセルが置かれることを防ぐ.

./script/blockage.tcl(抜粋)

```
for {set x [expr $floorplan_X1 + [expr 0.264 * ${STRIPE_X}] + 0.099]} \
    {$x < $floorplan_X2} \
    {set x [expr $x + [expr 0.264 * 49]] } {
createObstruct $x $floorplan_Y1 \
    [expr $x + 0.594] $floorplan_Y2
createObstruct [expr $x + 1.716] $floorplan_Y1 \
    [expr $x + 0.594 + 1.716 ] $floorplan_Y2
    }
```

5.2.2 マクロの境界

マクロの境界付近にスタセルが配置されることを防ぐ.

./script/blockage.tcl(抜粋)

```
createRouteBlk -box $floorplan_X1 $floorplan_Y1      [expr $floorplan_X1 + 0.33 - 0.066] [expr $floorpla
createRouteBlk -box $floorplan_X2 $floorplan_Y1      [expr $floorplan_X2 - 0.33 + 0.066] [expr $floorpla
createRouteBlk -box $floorplan_X1 $floorplan_Y1      $floorplan_X2  [expr $floorplan_Y1 + 0.33 - 0.066]
createRouteBlk -box $floorplan_X1 [expr $floorplan_Y2 + [expr $cell_height * 2]] $floorplan_X2  [expr $f
```

図 14 に，ブロッケージを配置した結果を示す.

5.3 ./script/place.tcl

./script/place.tcl(抜粋)

```
# ================================================================================ #
# --- Setting for trialRoute/IPO --- #
# ================================================================================ #
setTrialRouteMode -highEffort true -maxRouteLayer $maxLayerNumber -handlePreroute true
setAnalysisMode -checkType setup
setOptMode -optimizeAssignNet true -optimizeConstantNet true -rebuffer true -bufferAssignNets true
setOptMode -minimizeArea true -setupTargetSlack 0.0 -preserveModuleFunction false -reclaimArea true

...

# ================================================================================ #
# --- placeDesign  --- 
# -------------------------------------------------------------------------------- #
# - If you want to execute non-timingDriven placement,
#   please set -noTimingDriven instead of -timingDriven.
# - Please execute scan reordering after placement/optimization.
#   So setPlaceMode setting is -noReorderScan.
# - If placeDesign or OptDesign stop in the step of checkPlace related on Macro
#   pin, please set "setPlaceMode -allowBorderPinAbut true"
# ================================================================================ #
setPlaceMode -timingDriven true -reorderScan false -maxRouteLayer $maxLayerNumber

if { [ info var clkSpec ] != "" } {
    setPlaceMode -clkGateAware true
} else {
    Puts "WARNING : -clkGateAware Option Not appended to setPlaceMode because no CTS Spec Files are spec
}

##setPlaceMode -congEffort high
setPlaceMode -congEffort medium
```

./script/place.tcl(抜粋続き)

```
# -------------------------------------------------------------------------------- #
# Notice : If your design has ERRORs in GateDRC (ERROR 0410 or 7030),
#          please Set setOptMode -neverAddPort
#          However, this option may be deteriorate the performance of
#          timing optimization
# -------------------------------------------------------------------------------- #

setOptMode -neverAddPort

placeDesign

trialRoute -highEffort
saveDesign ${encounterDBS}/place.enc
```

図 15 に，配置の結果のレイアウトを示す．図上では，trialRoute を行ったため，簡易的に配線が行われている．

5.4 ./script/cts.tcl

クロックツリーの生成 (Clock Tree Synthesis:CTS) を行う．これも，Renesas のスクリプトがほとんどそのまま使える．
一部自分でも解決していないのだが，top 階層ではないため，CLK と RST_N がインスタンスとして宣言されておらず，うまく動かない命令が存在する．これは現時点ではコメントアウトしているが，それでも簡易 DRC フリーになるため，放置している．

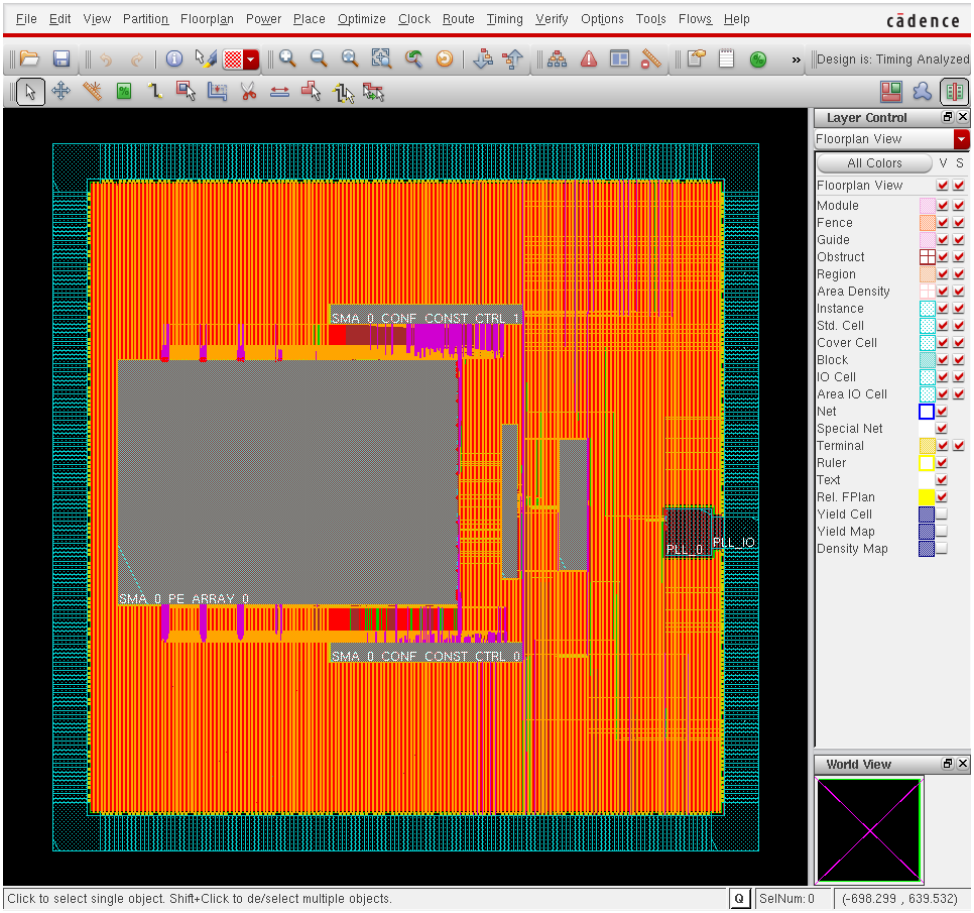

図 15: 配置の結果

./script/cts.tcl(抜粋)

```
# コメントアウト
# dbSetInstPlacementStatus [dbGetInstByName $clk_pin_name] dbcPlaced
# ecoChangeCell -inst "$reset_pin_name" -cell "LDH_BUF_S_10"
```

また，それに伴う CLK，RST_N のインスタンスを取得するコマンドはすべてコメントアウトしてしまっている．これは問題があれば御教授を願いたい．

./script/cts.tcl(抜粋)

```
# コメントアウト
# dbSetInstPlacementStatus [dbGetInstByName "core/clk_blk/cts_clk1/CTS_ROOT"] dbcPlaced
# ecoChangeCell -inst "core/clk_blk/cts_clk1/CTS_ROOT" -cell "LDH_BUF_S_10"

# dbSetInstPlacementStatus [dbGetInstByName "core/clk_blk/cts_clk2/CTS_ROOT"] dbcPlaced
# ecoChangeCell -inst "core/clk_blk/cts_clk2/CTS_ROOT" -cell "LDH_BUF_S_10"

# dbSetInstPlacementStatus [dbGetInstByName "core/cts_reset/CTS_ROOT"] dbcPlaced
# ecoChangeCell -inst "core/cts_reset/CTS_ROOT" -cell "LDH_BUF_S_10"
```

また，実行に際し，./script/cts_spec というファイルを用意する．これには，クロックとリセットの制約条件を記述するが，ほとんど Renesas が提供したものと変わらない．

図 16 に，CTS を実行後のレイアウトを示す．配線が除去されるが，多分大丈夫 ???

確認事項

./FECTS_saveDIR/cts.ctsrpt に，CTS のレポートが生成されている．

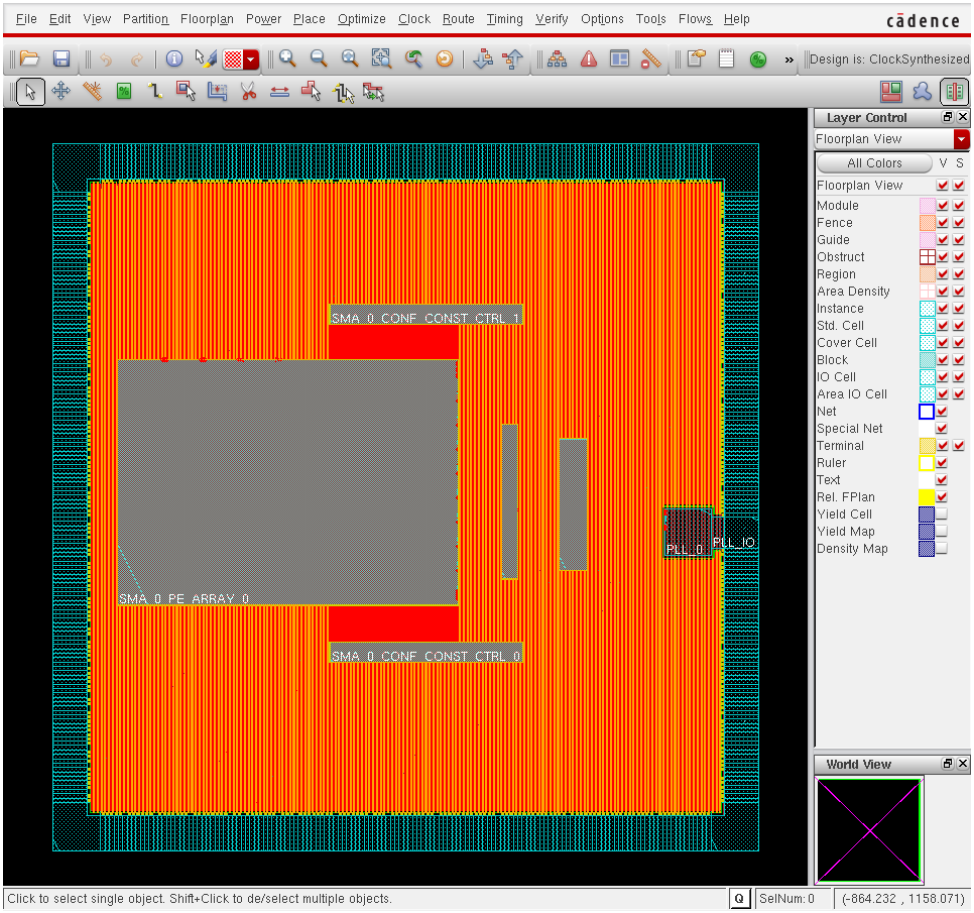

図 16: CTS の結果

クロックのスキュー値が想定外に大きくなっていないか確認する.

./FECTS_saveDIR/cts.ctsrpt(抜粋)

```
Nr. of Subtrees                : 1
Nr. of Sinks                   : 258
Nr. of Buffer                  : 7
Nr. of Level (including gates) : 2
Root Rise Input Tran           : 200(ps)
Root Fall Input Tran           : 200(ps)
Max trig. edge delay at sink(R): CONTEXT_CONT_SW_B/CONTEXT_MEM0/CLKB 545.9(ps)
Min trig. edge delay at sink(R): RFILE0/RFILE_CORE0/RFILE_reg_4__3_/CK 505.2(ps)

                                 (Actual)              (Required)
Rise Phase Delay               : 505.2~545.9(ps)       100~20000(ps)
Fall Phase Delay               : 621.4~664(ps)         100~20000(ps)
# この値が大きくなっていないかを確認する
Trig. Edge Skew                : 40.7(ps)              250(ps)
Rise Skew                      : 40.7(ps)
Fall Skew                      : 42.6(ps)
Max. Rise Buffer Tran          : 234.8(ps)             500(ps)
Max. Fall Buffer Tran          : 318.4(ps)             500(ps)
Max. Rise Sink Tran            : 203.3(ps)             500(ps)
Max. Fall Sink Tran            : 286.4(ps)             500(ps)
Min. Rise Buffer Tran          : 232.5(ps)             0(ps)
Min. Fall Buffer Tran          : 317.4(ps)             0(ps)
Min. Rise Sink Tran            : 128(ps)               0(ps)
Min. Fall Sink Tran            : 179.8(ps)             0(ps)
```

5.5 配線

Nanoroute を立ち上げず，SOC Encounter から Nanoroute を操作する．
といっても，スクリプトはほとんど変更の必要がない．
心配事項だが，いくつか SOC Encounter で使えないコマンドが存在する．

./script/nanoroute.tcl(抜粋)

```
pdi deselect
pdi delete_wire -violated
```

これらに変わるコマンドが見つからず，コメントアウトしてしまった．
配線では，Nanoroute の設定が非常に重要である．これらは Renesas のものをそのまま使うこと．

./script/nanoroute.tcl(抜粋)

```
##
## 1st initial option setting
##
pdi clear_all

# --- Multithread Option ( Number Change ) --- #
pdi set_option env_number_processor $NumOfCPU
# --- Multithread Option ( Number Change ) --- #

pdi set_option droute_on_grid_only true
#pdi set_option droute_on_grid_only false
pdi set_option routeWithViaInPin true
pdi set_option routeWithViaOnlyForStandardCellPin true
pdi set_option dbKeepFillWires true

pdi set_option routeAutoPinAccessUseViaOnly "1:1"

pdi set_option drouteExpIgnoreEolSideSpacing true
pdi set_option droute_exp_use_advanced_tapering true
##
## 2nd initial option setting
##
pdi set_option droute_exp_fix_violation_for_close_pin_connection true
pdi set_option droute_exp_relax_via_line_samenet_notch true
pdi set_option drouteUseConservativeCutSpacingForPin true
pdi set_option route_strictly_honor_non_default_rule true
pdi set_option droute_use_conservative_cut_spacing_for_special_via false
pdi set_option droute_honor_obs_around_pin true
pdi set_option droute_use_min_spacing_for_blockage false
pdi set_option droute_allow_merged_wire_at_pin false
pdi set_option droute_check_min_enclosed_area true
pdi set_option droute_auto_stop false
pdi set_option route_top_routing_layer $TopLayer
pdi set_option env_honor_track true
pdi set_option droute_fast_area_route_mode true
pdi set_option db_detect_wire_crossing true

pdi set_option drouteTopMultiCutViaRoutingLayer 5
pdi set_option drouteBottomMultiCutViaRoutingLayer 1
```

配線を行う．globalRoute をした後，detailRoute を行う．

./script/nanoroute.tcl(抜粋続き)

```
pdi set_option route_use_guide true
pdi set_option groute_trunk_pin_first true

pdi set_option droute_start_iteration default
pdi set_option droute_end_iteration default
pdi global_route
pdi detail_route

pdi set_option route_use_guide false
pdi set_option groute_trunk_pin_first false

pdi set_attribute -net * -skip_routing false
```

図 5.5 に，配線を行った後のレイアウトを示す．

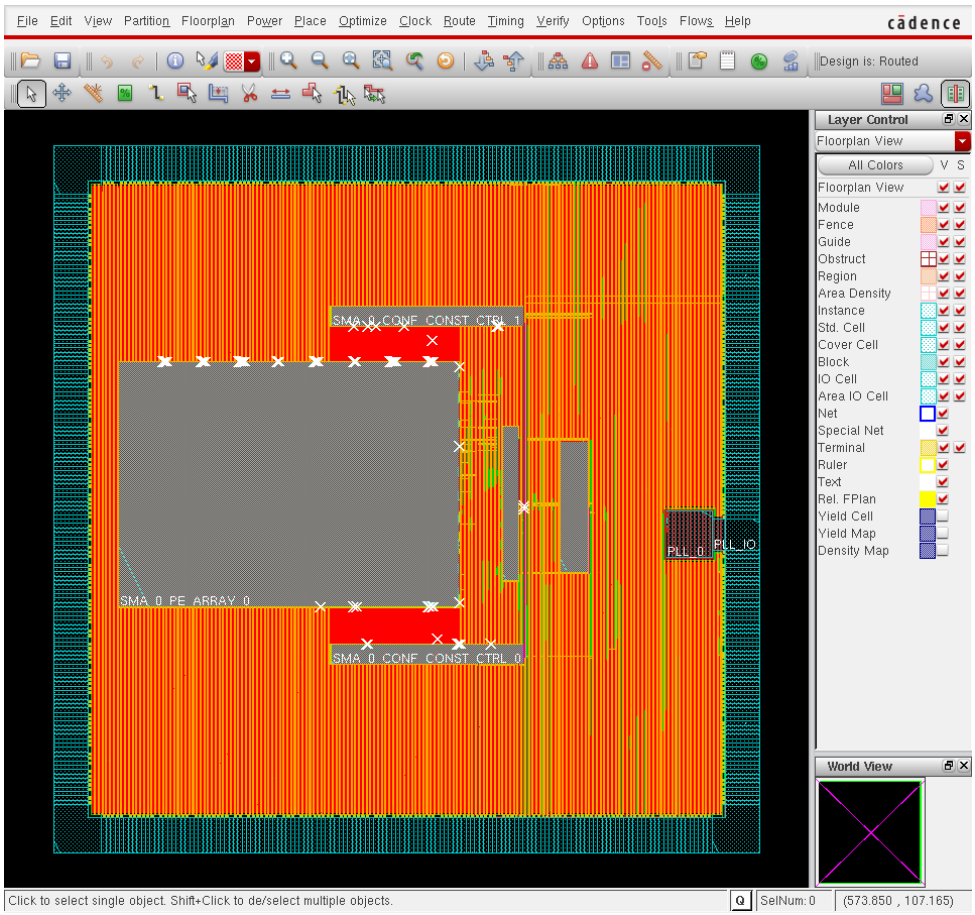

図 17: 配線後のレイアウト

確認事項

レポートの DRC の数を確認する．

```
#Total number of DRC violations = 12
#Total number of violations on LAYER M1L = 0
#Total number of violations on LAYER M2L = 0
#Total number of violations on LAYER M3L = 0
#Total number of violations on LAYER M4L = 1
#Total number of violations on LAYER M5L = 0
#Total number of violations on LAYER M6S = 0
#Total number of violations on LAYER M7T = 0
#Total number of violations on LAYER PM = 0
```

M4L 層にエラーが 12 個残っている．

エラーが残った場合，ecoRoute を行う．

5.6 ./script/ecoroute.tcl

ecoRoute モードで再度配線を行う．

```
source "./script/ecoroute.tcl"
```

Nanoroute モードを ecoRoute にすることにより，ecoRoute が開始される．

./script/ecoroute.tcl(抜粋)

```
setNanoRouteMode -routeInsertAntennaDiode true
setNanoRouteMode -routeWithEco true
setNanoRouteMode -drouteStartIteration 0
globalDetailRoute
```

ecoRoute 後，レポートのエラーが 0 になっていることを確認する．

```
#Total number of DRC violations = 0
#Total number of net violated process antenna rule = 0
#Total number of violations on LAYER M1L = 0
#Total number of violations on LAYER M2L = 0
#Total number of violations on LAYER M3L = 0
#Total number of violations on LAYER M4L = 0
#Total number of violations on LAYER M5L = 0
#Total number of violations on LAYER M6S = 0
#Total number of violations on LAYER M7T = 0
#Total number of violations on LAYER PM = 0
```

画面上に DRC のマーカーが残るが，エラーは消えているので大丈夫だと思う．

6 容量セル埋め/フィラー挿入等

マクロ設計時と同一の処理である．

6.1 フィラーセル埋め ./script/filler.tcl

マクロ設計時と同一の処理である．

7 簡易 Verify と GDS 出力

SOC Encounter 上にて簡易的な Verify を行い，GDS を出力する．

7.1 簡易 verify ./script/verify.tcl

簡易 verify は，余計な読み込み部分などをのぞき，Renesas のサンプルスクリプトをそのまま実行させる．

```
source "./script/verify.tcl"
```

確認事項

- DRC が 0 になっている．
- ログにエラーが無い
- 正しく SMA_2_TOP.gds2.gz が出力されている．