How to Layout 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 プロセスを用いてマクロ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 DMEM_ACCESS_CTRL マクロを用いた場合を説明する．本マクロは，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
DMEM_ACCESS_CTRL.v	論理合成済みネットリスト
DMEM_ACCESS_CTRL.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 (PE.sdc)

vnet ネットリストが入っている (PE-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]
```

図 1 に，script/load_design.tcl を実行した後の結果を示す.

3.3 script/floorplan.tcl

フロアプランを行う.
フロアプランでは，マクロ (今回はメモリ) の配置などを決める.

3.3.1 メモリ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,

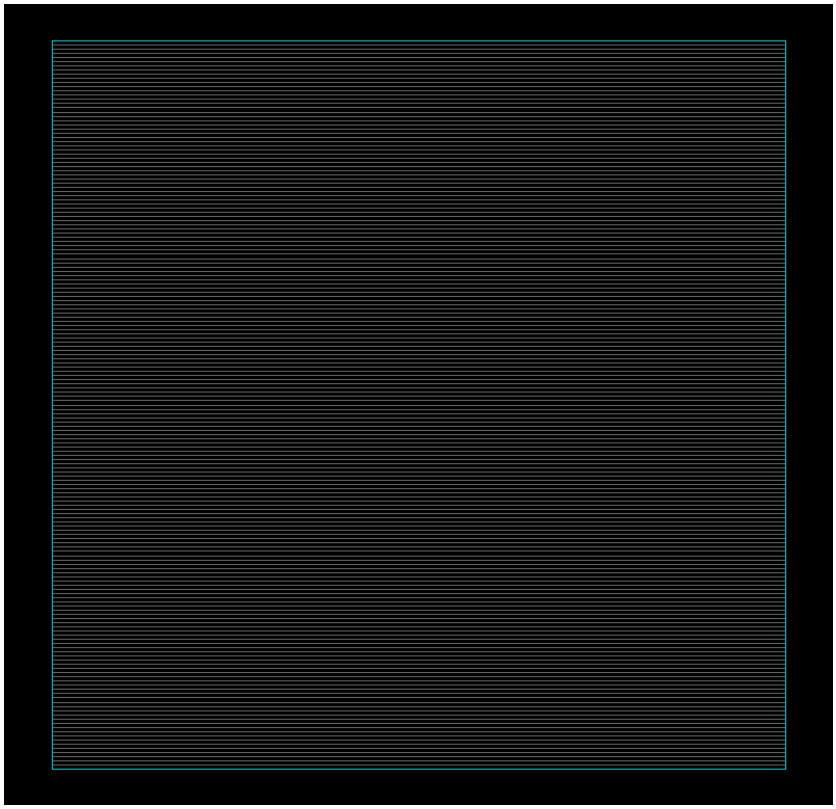

図 1: デザイン読み込み結果

```
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

```
floorPlan -site nc40_dsc -s $floorplan_X2 $floorplan_Y2 \
0.0 [expr $cell_height] 0.0 [expr $cell_height]
```

メモリを配置する．

```
placeInstance INST_I_WDREG110PAA128W14C1 [expr 0.132 * ${IMEM1X} + 0.066] \
        [expr ${cell_height} * ${IMEM1Y}]   R${IMEM1R} -fixed
placeInstance IMEM_I_WDREG110PAA128W14C1 [expr 0.132 * ${IMEM2X} + 0.066] \
        [expr ${cell_height} * ${IMEM2Y}]   R${IMEM2R} -fixed
```

Row カットと stop セルを配置する．これにより，メモリの横方向から電源が配線が入り込み，Geometry エラーになることを防ぐ．

```
selectInstByCellName ${MEMORY_CELL_NAME}
cutRow -selected -topGap [expr ${powerGapH} * 2] -bottomGap [expr ${powerGapH} * 3] \
    -leftGap [expr ${powerGapV} * 2] -rightGap [expr ${powerGapV} * 2]
deselectAll

addEndCap -preCap LDL_POWERSTOPL -postCap LDL_POWERSTOPR \
    -prefix LDL_POWERSTOP_
```

図 2 に，Row カットと stop セルを配置した結果を示す．

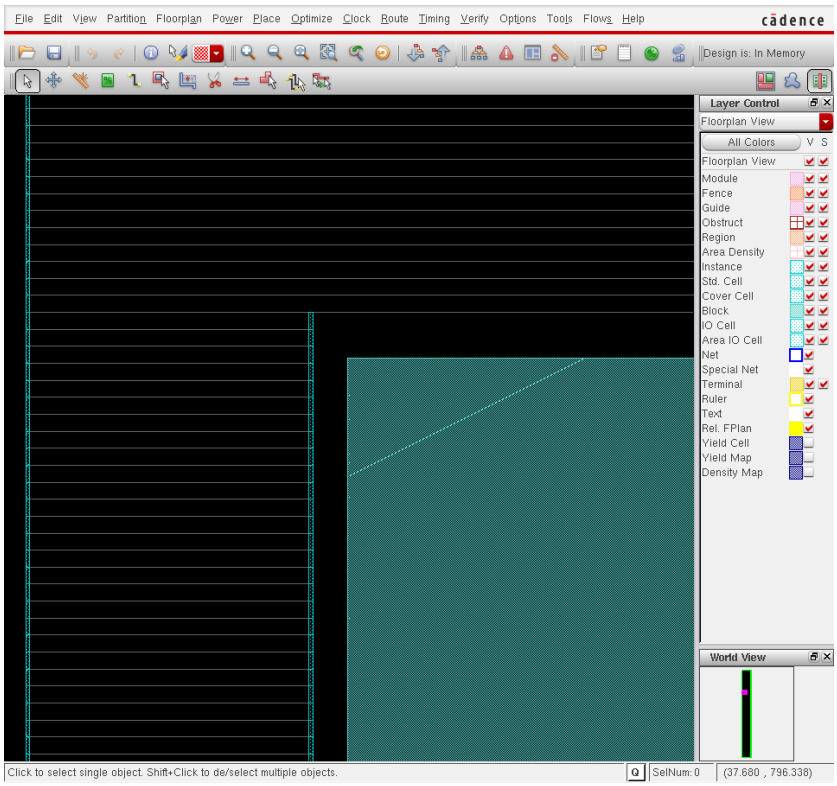

図 2: Row カットと stop セルの配置結果

3.4 フロアプランの結果

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

確認事項

- 正しいサイズでコアエリアが生成できているか？
- メモリは正しい位置に配置されているか，メモリの角度 (R270) は正しいか？
- 各モジュールは OffGrid となっていないか？
- メモリ領域には，M4L 層に Routing Blockage は張ってあるか？
- メモリのまわりは，Row カットがなされているか？POWERSTOP は入っているか？

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．

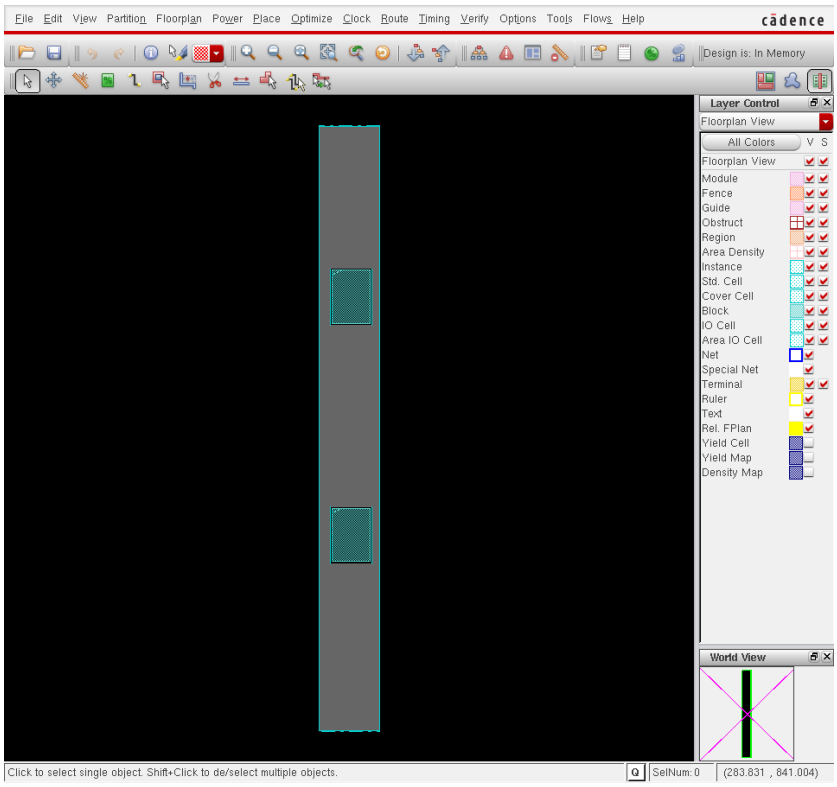

図 3: フロアプラン結果

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 "./PE_floorplan.v"
set inputSDC     "./sdc/PE.sdc"
set inputDef     "./PE_floorplan.def"

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

4.3 電源配線 tcl./script/proute.tcl

まず，via が太らないように各種設定を行っている．これは内藤電誠からもらったスクリプトをそのまま流用したので，あまり意味が分かっていない．

./script/proute.tcl(抜粋)

```
setAddRingOption -extend_stripe_search_distance 0.0
setViaGenOption -optimize_cross_via 1

set srouteExtraConfig_cfg [open ./script/align.cfg {WRONLY CREAT TRUNC}]
   puts $srouteExtraConfig_cfg "srouteAlignViaOnStripeOffsetFromCenter 264"
   puts $srouteExtraConfig_cfg "srouteNoViaOnWireShape NONE"
   puts $srouteExtraConfig_cfg "srouteTargetVerticalMargin 660"
#   puts $srouteExtraConfig_cfg "srouteLayerNormalCost.5  8"
#   puts $srouteExtraConfig_cfg "srouteLayerWrongWayCost.5  8"
close $srouteExtraConfig_cfg
##
```

メモリ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．

./script/proute.tcl(抜粋)

```
addRing \
    -spacing_top    0.132 \
    -spacing_bottom 0.132 \
    -spacing_left   0.132 \
    -spacing_right  0.132 \
    -width_top      0.396 \
    -width_bottom   0.396 \
    -width_left     0.264 \
    -width_right    0.264 \
    -layer_top      M4L \
    -layer_bottom   M4L \
    -layer_left     M5L \
    -layer_right    M5L \
    -offset_top     [expr 0.165 * 8] \
    -offset_bottom  [expr 0.165 *10] \
    -offset_left    [expr 0.264 * 4] \
    -offset_right   [expr 0.264 * 4] \
    -stacked_via_top_layer    M5L \
    -stacked_via_bottom_layer M4L \
    -snap_wire_center_to_grid Grid \
    -around         each_block \
    -type           block_rings \
    -nets           {VDD VSS}
```

図 4 に，メモリに対してリングを張った結果を示す．
メモリに対して電源レールを張る．メモリのポートに対して水平にレールを張る．これは富士通の 65nm と同一である．

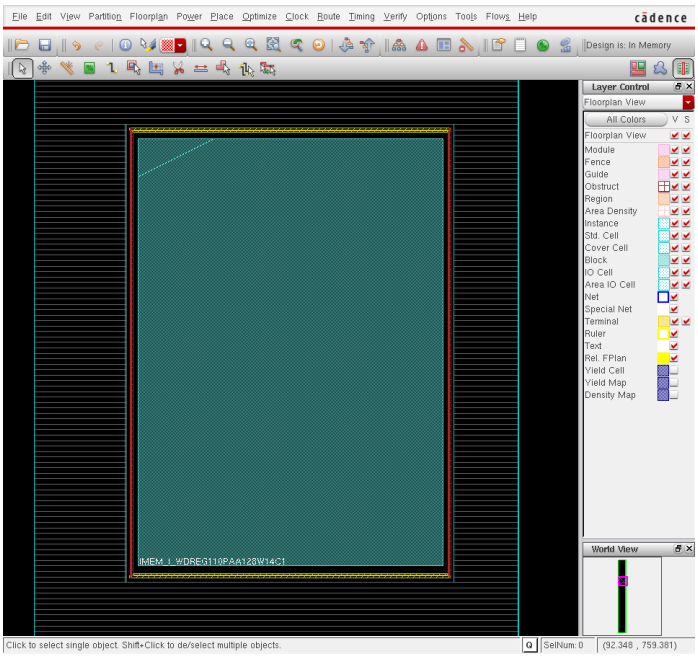

図 4: メモリに電源リングを張った結果

./script/proute.tcl(抜粋)

```
selectInstByCellName ${MEMORY_CELL_NAME}
addStripe \
    -layer      M4L \
    -width      pin_width \
    -spacing    0.066 \
    -pin_layer  M4L \
    -over_pins  1 \
    -same_sized_stack_vias  1 \
    -stacked_via_top_layer M4L \
    -stacked_via_bottom_layer M4L \
    -over_power_domain      1 \
    -orthogonal_only        0 \
    -direction horizontal \
    -nets {VSS VDD}
deselectAll
```

図 5 に，メモリ電源に対してレールを張った結果を示す．
縦方向にストライプを引く．メモリのレールに対して自動的に電源の接続が行われ，メモリの電源接続が完了する．

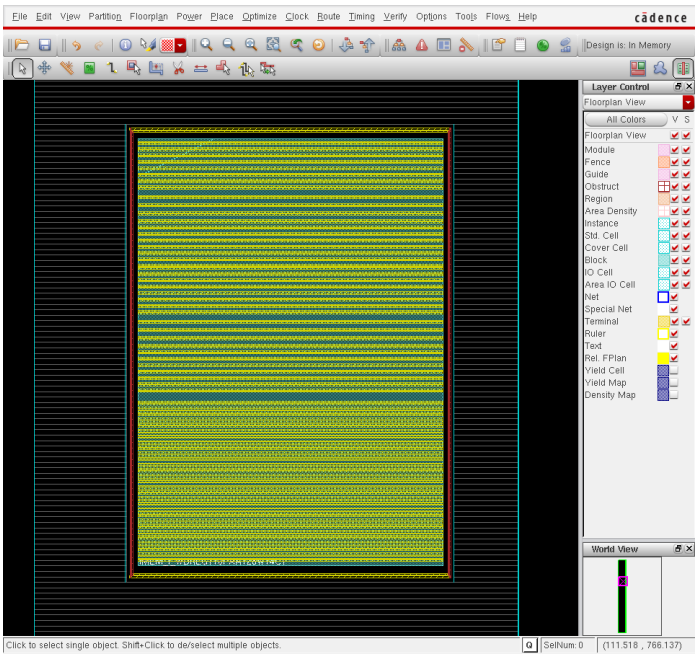

図 5: メモリ電源にレールを張った結果

./script/proute.tcl(抜粋)

```
addStripe -nets {VSS VDD } \
    -layer M5L \
    -spacing [expr 1.056] \
    -width [expr 0.066 * 9] \
    -set_to_set_distance [expr 0.264 * 49] \
    -xleft_offset [expr 0.264 * ${STRIPE_X} + 0.066 + 0.033] \
    -block_ring_top_layer_limit M7T \
    -block_ring_bottom_layer_limit M1L \
    -stacked_via_top_layer M5L \
    -stacked_via_bottom_layer M1L \
    -padcore_ring_top_layer_limit M5L \
    -padcore_ring_bottom_layer_limit M1L \
    -extend_to design_boundary \
    -merge_stripes_value ${cell_height} \
    -split_vias 0 \
    -snap_wire_center_to_grid Grid
```

図 6 に，メモリ電源に対してストライプを張った結果を示す．

メモリのレールと，電源ストライプが接続されていることを確認する．

次に電源レールを張る．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．

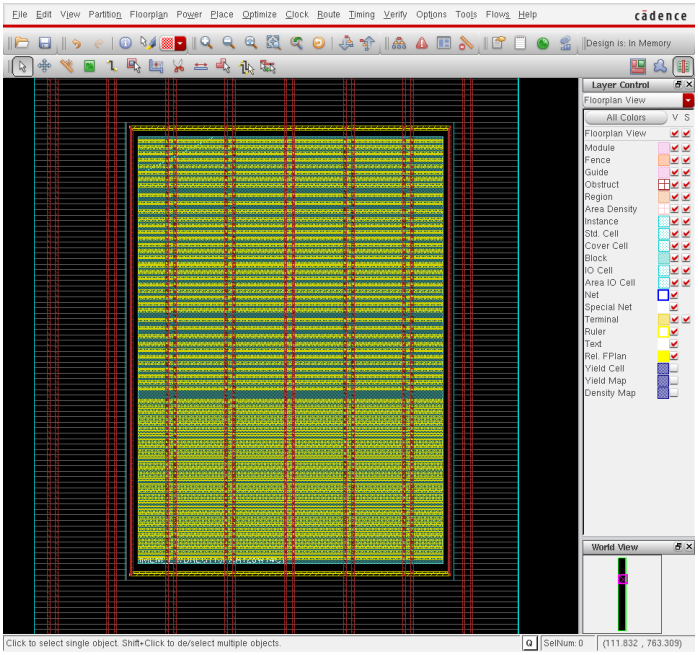

図 6: メモリ電源にストライプを張った結果

./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？
- メモリのまわりで，レールが止まっているか？

図 7 に，ここまでの電源配線の結果のレイアウトを示す．
図 8 に，メモリまわりの電源の様子を示す．メモリのピンに電源が接続されていることが分かる．

確認事項

- メモリのまわりにリングは正しく巻かれているか (他のレールとくっついていないか)？
- 明らかに DRC になりそうと思われる複雑な電源接続線はできていないか？

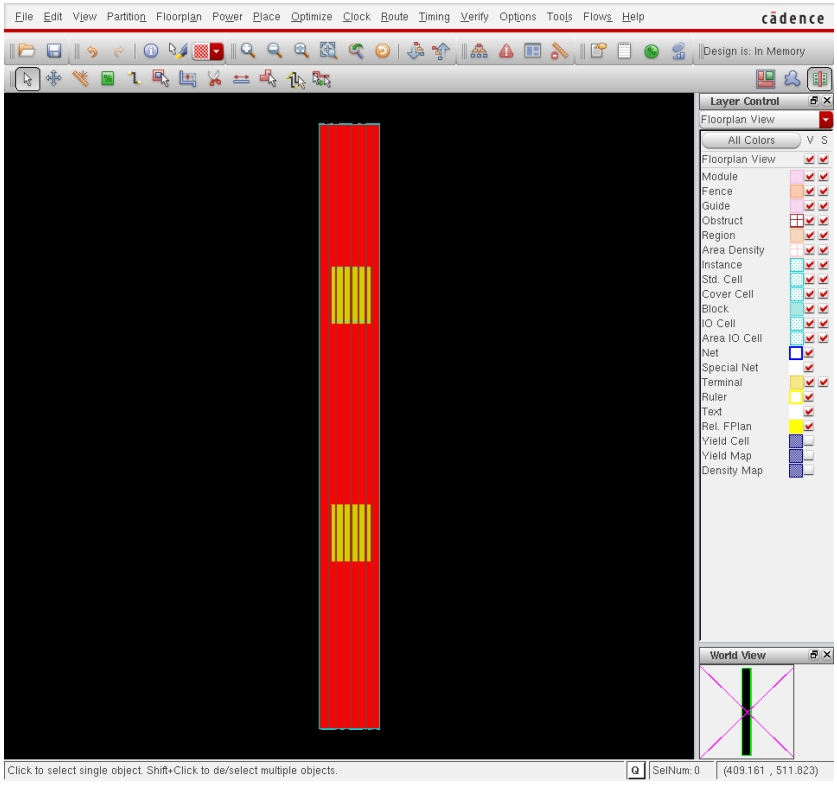

図 7: 最終電源レイアウト

- すべてのメモリのピンに電源が接続されているか (これを目視で確認するのは非常に難しいが…)？
- POWERSTOP により，メモリのリングが上下のストラップ以外と接続されていることはないか (経験上これが DRC になりやすい)？
- ストライプは，メモリリングの境界で正しく切れているか？

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．

5 配置配線

レイアウト上もっとも時間のかかる部分である．

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
- メモリの境界領域

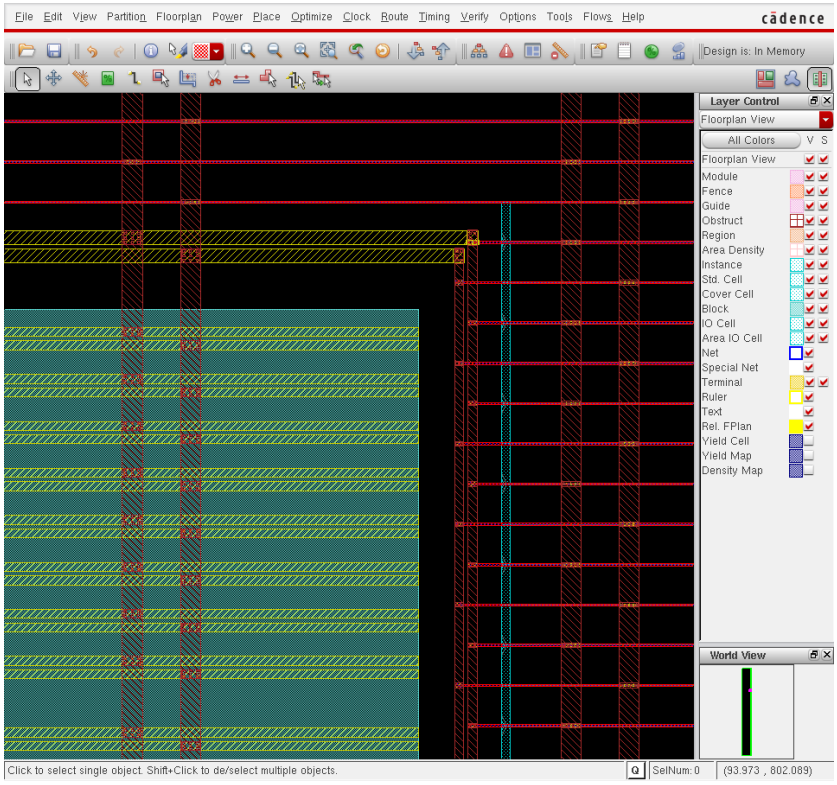

図 8: メモリ周辺の電源レイアウトの様子

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
```

5.2.3 メモリの境界

マクロと同様，メモリの境界にもスタセルが配置されることを防ぐ．

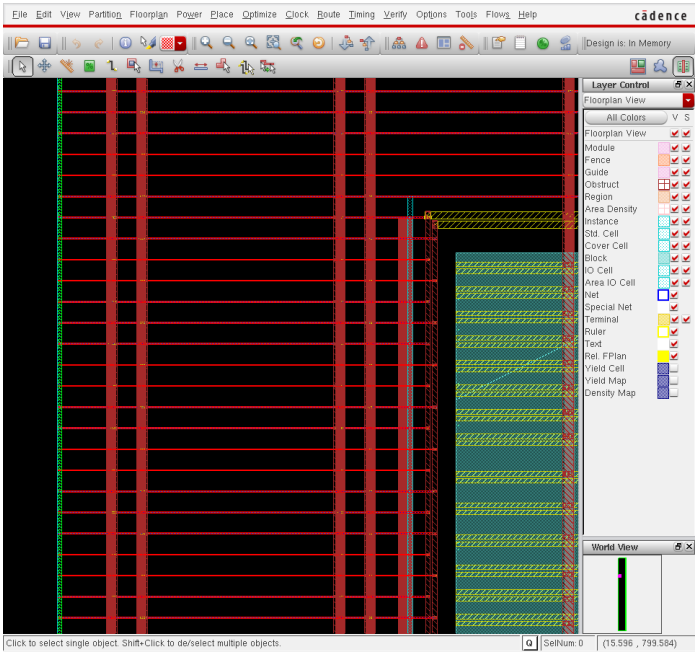

図 9: ブロッケージの配置結果

./script/blockage.tcl(抜粋)

```
# IMEM1 border constraint
set obstX [expr 0.132 * ${IMEM1X} + 0.066]
set obstY [expr ${cell_height} * ${IMEM1Y} - $powerGapV]
createObstruct [expr $obstX - $powerGapH - 1.919] $obstY \
    [expr $obstX - $powerGapH] [expr $obstY + $sram_width + 2 * $powerGapV]
createObstruct [expr $obstX + $sram_height + $powerGapH] $obstY \
    [expr $obstX + $sram_height + $powerGapH + 1.919] [expr $obstY + $sram_width + 2 * $powerGapV]

# IMEM2 constraint
set obstX [expr 0.132 * ${IMEM2X} + 0.066]
set obstY [expr ${cell_height} * ${IMEM2Y} - $powerGapV]
createObstruct [expr $obstX - $powerGapH - 1.919] $obstY \
    [expr $obstX - $powerGapH] [expr $obstY + $sram_width + 2 * $powerGapV]
createObstruct [expr $obstX + $sram_height + $powerGapH] $obstY \
    [expr $obstX + $sram_height + $powerGapH + 1.919] [expr $obstY + $sram_width + 2 * $powerGapV]
```

図 9 に，ブロッケージを配置した結果を示す.

5.3 ./script/DMEM_ACCESS_CTRL_pin.tcl

ピン制約を行う.

./tdf/DMEM_ACCESS_CTRL_pin.tdf(抜粋)

```
...
setPinConstraint -cell PE  -pin WEST_PE_ALU_16_     -layer {M4L}   -edge 0
setPinConstraint -cell PE  -pin WEST_PE_ALU_15_     -layer {M4L}   -edge 0
setPinConstraint -cell PE  -pin WEST_PE_ALU_14_     -layer {M4L}   -edge 0
setPinConstraint -cell PE  -pin WEST_PE_ALU_13_     -layer {M4L}   -edge 0
setPinConstraint -cell PE  -pin WEST_PE_ALU_12_     -layer {M4L}   -edge 0
...
```

-edge id の id の数字の意味は，マニュアルを見られたし.

重要な点として，ピンの大きさの変更を行う必要がある．通常マクロの配置配線を行うと，各 (入出力) ピンの長さが 0.329 という非常に謎めいた値となる．このまま勧めると MinStep エラーを起こすため，0.330 の切りの良い長さにピンの大きさを変更する．

./tdf/DMEM_ACCESS_CTRL_pin.tcl(抜粋)

```
setPinDepth -cell PE  -pin * -depth 0.330
```

5.4 ./script/place.tcl

これは Renesas 提供の tcl をほとんどそのまま走らせて問題ない．

./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 10 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．

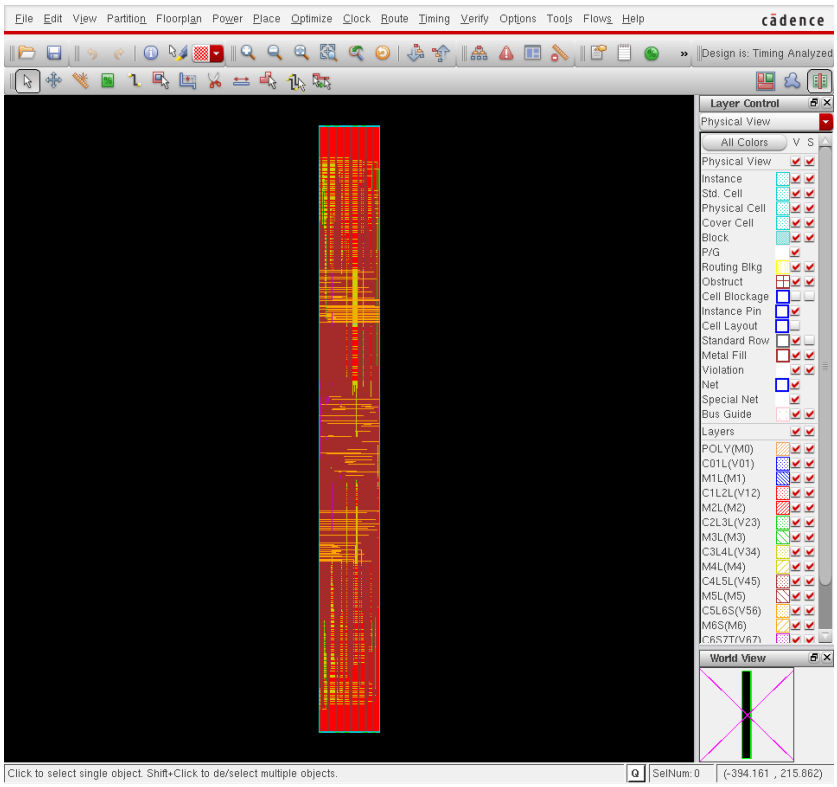

図 10: 配置の結果

確認事項

- フロアプランの中心に，不可解なセルが配置されていないか (メモリなどのライブラリの読み込みに失敗している場合，このような症状がおき，DRC が集中する)？
- 各入出力ピンはフロアプランの正しい側面に配置されているか (制約が効いていない場合，フロアプラン上部にピンが集中する)？
- 各入出力ピンの長さが，0.330 になっているか？ピン長制約をかけなければ，0.329 になるはずである．これをチェックする (図 5.4)

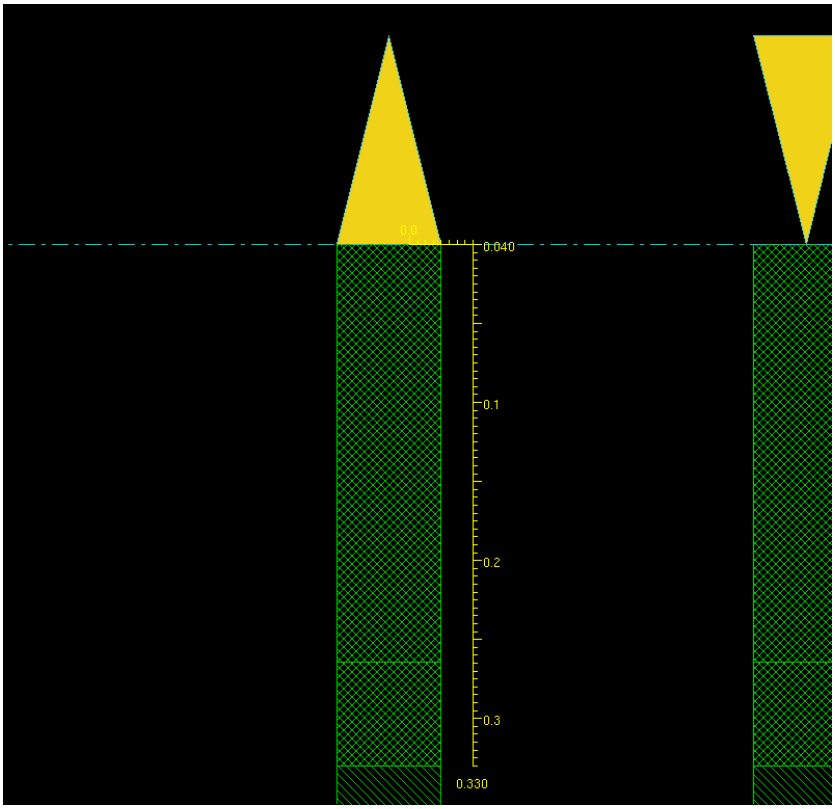

図 11: ピンの長さは 0.33 の倍数でなければならない

5.5 ./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．

./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 12 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 ???

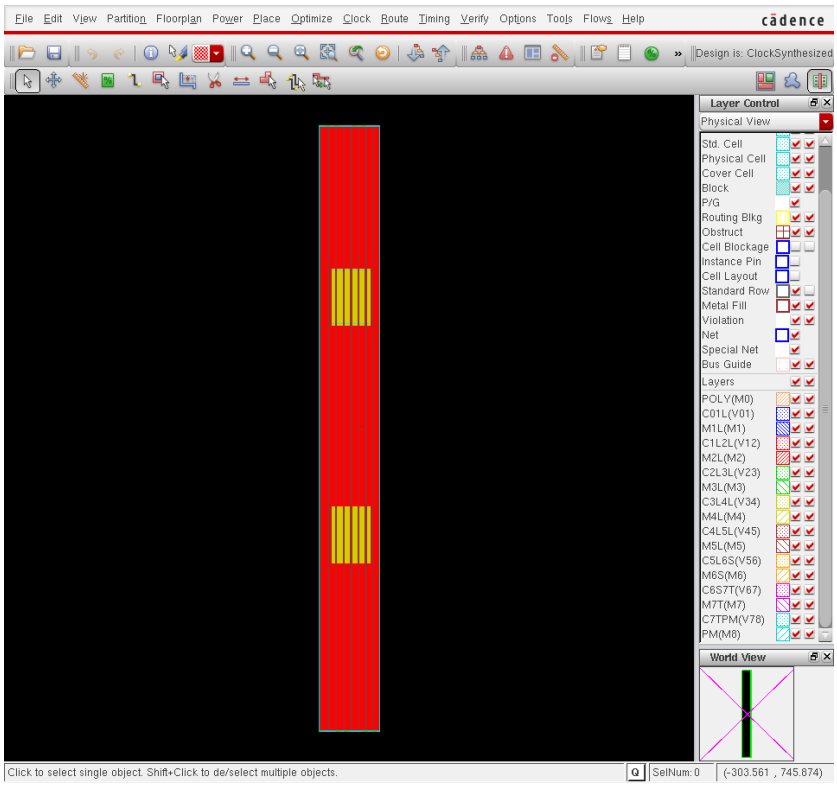

図 12: CTS の結果

確認事項

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

5.6 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 5.6 に，配線を行った後のレイアウトを示す．

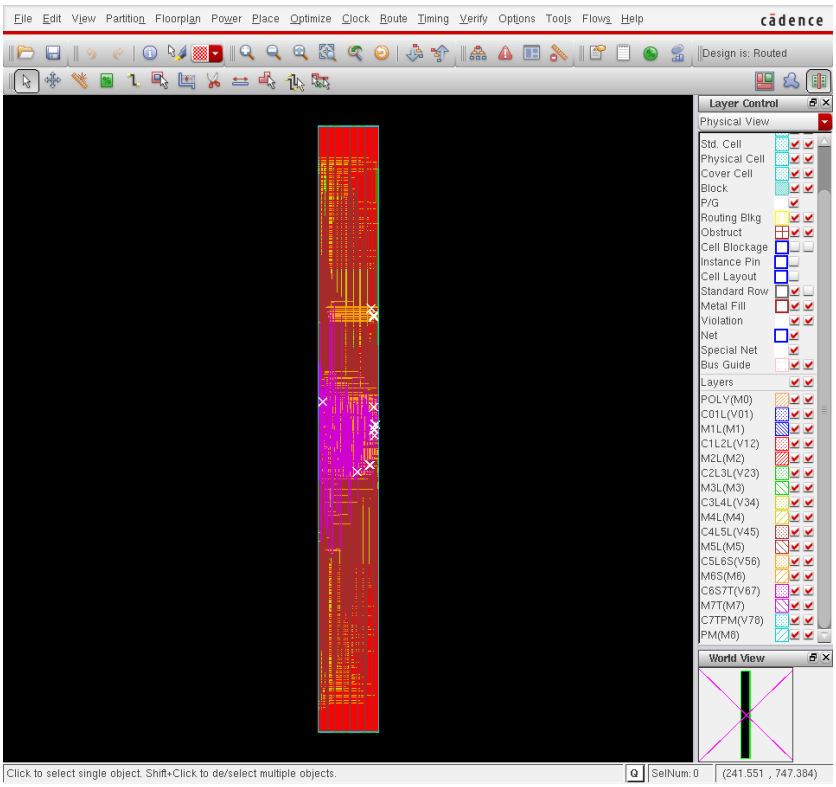

図 13: 配線後のレイアウト

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．

5.7 ./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

配置配線はここまでで大方終了し，最後にセル埋め，メタル埋めなどの後処理を行う.

6.1 容量セル埋め./script/cdfill.tcl

容量セルを埋めることにより，配置で余った部分を満たしている.

```
source "./script/cdfill.tcl"
```

./script/cdfill.tcl(抜粋)

```
addDeCap -totCap $targetCap -maxNrIter $max_nr_iter \
        -addDRCCheckOption {-merge} \
        -addDRCCheckOption {-minHole} \
        -log deCapLog/addDeCap.log \
        -prefix $decap_prefix

rptDeCap add_decap

#****************************
# Inserting Normal FillCells
#****************************
addFiller -cell $normal_fill -prefix $normal_prefix -noDRC
```

図 6.1 に，容量セルを埋めた後のレイアウトを示す．図 6.1 に，ネットを隠したときのレイアウトを示す.
フロアプランの全域に容量セルが埋まっていることが分かる.

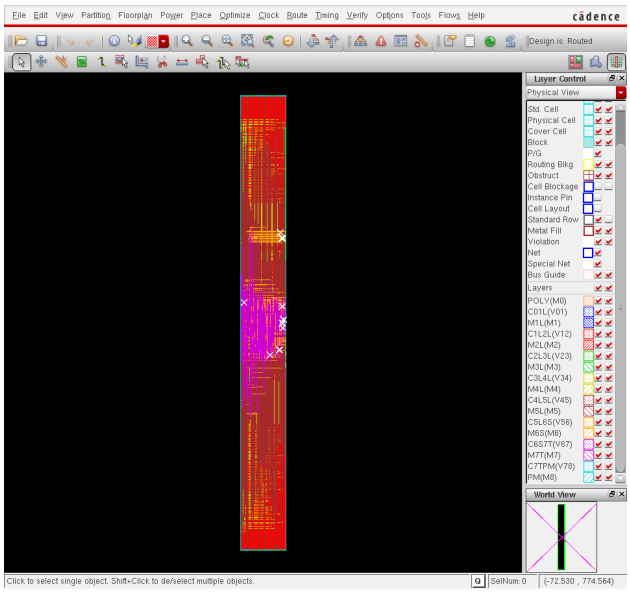

図 14: 容量セル埋め結果 (ネットあり)

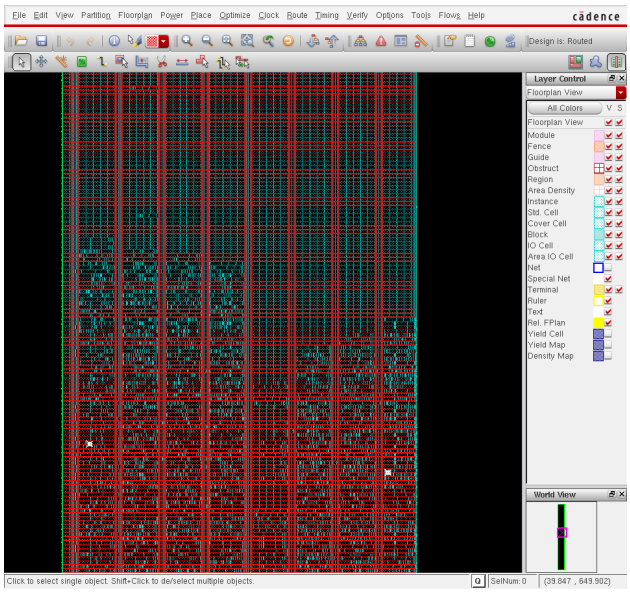

図 15: 容量セル埋め結果 (ネット非表示)

6.2 フィラーセル埋め./script/filler.tcl

容量セルで埋めることができなかった場所を，フィラーセルで埋める．

```
source "./script/filler.tcl"
```

./script/filler.tcl(抜粋)

```
addFiller -cell $normal_fill \
          -prefix $normal_prefix \
          -markFixed \
          -noDRC

checkPlace
```

今回は，本スクリプトを実行しても大きな変化は起こらない．

6.3 メタル埋め./script/metal_fill.tcl

メタル密度を満たすために，メタル層にダミーメタルを挿入する．

```
source "./script/metal\_fill.tcl"
```

./script/metal_fill.tcl(抜粋)

```
## -------------------------
##  Add MetalFill
## -------------------------

addMetalFill -iterationNameList { step1 step2 } -extraConfig cont.conf \
    -layer { 1 2 3 4 5 6 7 } -area $floorplan_X1 $floorplan_Y1 $floorplan_X2 $floorplan_Y2 \
    -timingAware on -snap -onCells -stagger off
#    -useNonDefaultSpacing
```

図 6.3 に，ダミーメタル挿入後のレイアウトを示す．

確認事項

メタル密度制約違反が存在しないか確認する．

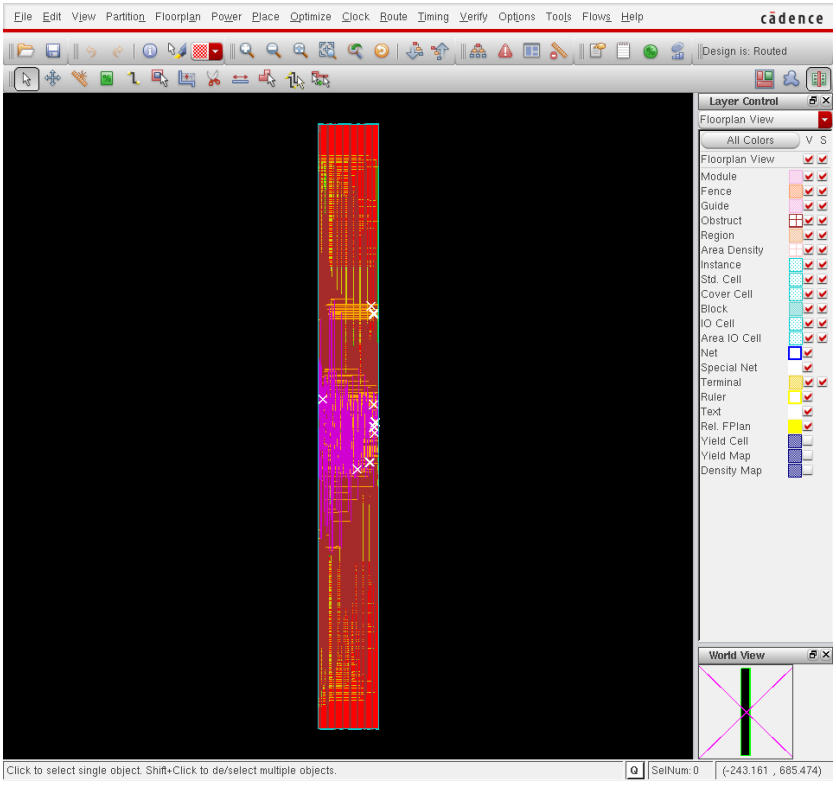

図 16: ダミーメタル挿入後のレイアウト

レポート内に，DRC が存在しないことを確認する．

```
******** Start: VERIFY DENSITY ********
Multi-cpu acceleration using 2 CPU(s).
Density calculation ...... Slot :   0 of   1 Thread : 0

No density violations were found.

******** End: VERIFY DENSITY ********
VMD: elapsed time: 2.00
     (CPU Time: 0:00:02.1  MEM: 22.086M)

Saving Drc markers ...
... 0 Drc markers are saved ...
```

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．

Geometry Check

```
 *** Starting Verify Geometry (MEM: 1175.0) ***

  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 1056
                  ..... maxBinCols: 129, maxBinRows: 129
Multi-cpu acceleration using 2 CPU(s).
 VERIFY GEOMETRY ..... SubArea : 1 of 36  Thread : 0
...
 VERIFY GEOMETRY ..... SubArea : 33 of 36  Thread : 0
 VERIFY GEOMETRY ..... SubArea : 35 of 36  Thread : 0
VG: elapsed time: 16.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
```

Connectity Check

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Tue Sep 28 21:22:52 2010

Design Name: PE
Database Units: 1000
Design Boundary: (0.0000, 0.0000) (244.2000, 239.9760)
Error Limit = 1000; Warning Limit = 50
Check all nets
Multi-cpu acceleration using 2 CPU(s).
*** Processing net VDD in Job 1
*** Processing net VSS in Job 2
**** 21:22:54 **** Processed 5000 nets (Total 6255) in Job 3

VC Elapsed Time: 0:00:02.0

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Tue Sep 28 21:22:54 2010
******** End: VERIFY CONNECTIVITY ********
```

Anntena Check

```
******* START VERIFY ANTENNA ********
Report File: PE.antenna.rpt
LEF Macro File: PE.antenna.lef
5000 nets processed: 0 violations
Verification Complete: 0 Violations
******* DONE VERIFY ANTENNA ********

Saving Drc markers ...
... 0 Drc markers are saved ...
```

なっていない場合，原因を探してレイアウトをやり直す．
成功した場合，図 7.1 のように，Violation Browser に何もエラーが表示されないはずである．

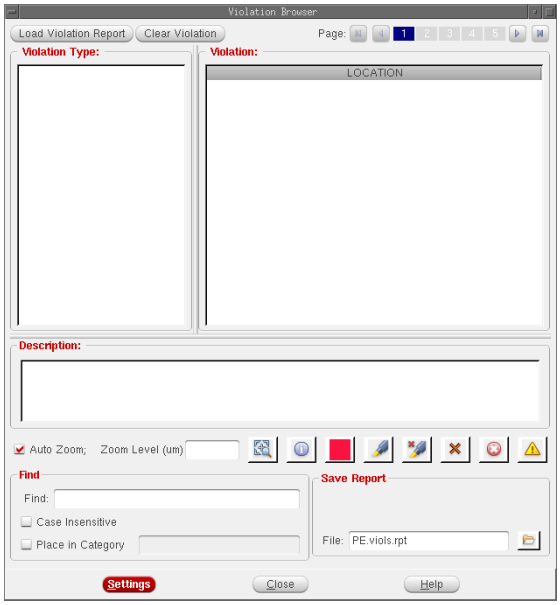

図 17: Violation Browser

7.2 GDS 出力 ./script/gdsout.tcl

GDS を出力する．

```
source "./script/gdsout.tcl"
```

./script/gdsout.tcl(抜粋)

```
setStreamOutMode -SEcompatible true
setStreamOutMode -supportPathType4 false
streamOut $O_GDS -mapFile $I_MAP -units 1000 -uniquifyCellNames -structureName ${DESIGN} -dieAreaAsBound
```

確認事項

- ログにエラーが無い
- 正しく PE.gds2.gz が出力されている．