Renesas 40nm SoCE PE ARRAY 戦譜

慶應大学 天野 2010.12.15

1 はじめに

本ドキュメントは、自動レイアウトサンプルスクリプト使用説明書 (説明書と略す) WEB-00465513-01 を元に、2010 年 11 月にテープアウトした SMA-2 のレイアウト作成のフローである。本ドキュメントで参照しているのはサンプルスクリプト使用説明書および SoCE フロー補足説明書である。SoCE のバージョンは 9.1 を利用している。

当方で解決できない問題については、ルネサスエレクトロニクスの箕輪さん、内藤電誠工業の新保さんに修正していただいた。深く感謝する。

2 PE のフロー

2.1 概観

PE アレイは PE を 80 個組みあわせて作る。まず部品となる PE の基本的なフローを示す。~hunga/sma/sma2/pr/enc4/script 中にスクリプトファイルがある。

```
set DESIGN PE
source "./script/env.tcl"
source "./conf/VDEC_soce.conf"
set inputLEF "{./lib/lef}"
set inputVerilog "./vnet/${DESIGN}-compile.v"
set inputSDC     "./sdc/${DESIGN}.sdc"
source "./script/load_design.tcl"
source "./script/floorplan.tcl"

# terminate temporary

set DESIGN PE
source "./script/env.tcl"
source "./conf/VDEC_soce.conf"
set inputLEF "{./lib/lef_proute}"
set inputVerilog "./${DESIGN}_floorplan.v"
set inputSDC     "./sdc/${DESIGN}.sdc"
set inputDef     "./${DESIGN}_floorplan.def"
source "./script/load_def.tcl"
source "./script/proute.tcl"

# terminate temporary

set DESIGN PE
source "./script/env.tcl"
source "./conf/VDEC_soce.conf"
set inputVerilog "./PE_proute.v"
```

```
set inputSDC     "./sdc/PE.sdc"
set inputDef     "./PE_proute.def"
set inputLEF "{./lib/lef}"
source "./script/load_def.tcl"
source "./script/PE_blockage.tcl"
source "./script/PE_pin.tcl"
source "./script/place.tcl"
source "./script/nanoroute.tcl"
source "./script/ecoroute.tcl"
source "./script/PE_repeater.tcl"
source "./script/ecoroute.tcl"
deleteAllRouteBlks
source "./script/cdfill.tcl"
deleteObstruction -all
source "./script/filler_no8GC1.tcl"
source "./script/metal_fill.tcl"
source "./script/verify.tcl"
source "./script/gdsout.tcl"
```

このフローではフロアプランが終わった後に SoCE を抜けて LEF を入れ替え、電源配線が終わったら、再び LEF を入れ替えている。面倒だが、電源配線の専用 LEF を利用するためやむを得ない。

1. フロアプラン：floorplan.tcl
2. 電源配線: proute.tcl
3. 配置: place.tcl
4. 配線: nanoroute.tcl, ecoroute.tcl, PE_repeater.tcl
5. フィラー埋め: cdfill.tcl, filler_no8GC1.tcl
6. メタル埋め: metal_fill.tcl
7. 検証：verify.tcl
8. GDS 吐き出し：gdsout.tcl

の順である。通常のフローとやや異なるのは、完全に組みあわせ回路なのでクロックツリーを張る所がない。

2.2 初期設定

レイアウトを行うディレクトリ中には、以下のディレクトリを作っておく

- script: tcl script を入れておく
- conf: VDEC_soce.conf(説明書の最初で作る configuration data) を入れておく
- lib: 説明書の最初で作る lef ファイルを置いておく
- tdf: ピン並びの tdf ファイルを置いておく (script に置いてもよかったかも)
- sdc: 論理合成時に生成した sdc ファイルを置いておく (リンク)
- vnet: 論理合成時に生成した verilog ファイルを置いておく (リンク)

2.3 フロアプラン

ここでは soce の version を 9.1 にすること。

```
set DESIGN PE
source "./script/env.tcl"
source "./conf/VDEC_soce.conf"
set inputLEF "{./lib/lef}"
set inputVerilog "./vnet/${DESIGN}-compile.v"
set inputSDC     "./sdc/${DESIGN}.sdc"
source "./script/load_design.tcl"
source "./script/floorplan.tcl"
```

env.tcl は説明書参照。これを研究室のパスに変えたもの。

load_design で設定したものを読み出す。

```
setUIVar rda_Input ui_timingcon_file $inputSDC
setUIVar rda_Input ui_leffile $inputLEF
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 $inputVerilog
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]
```

次にフロアプランを行う。

```
setIoFlowFlag 0
source script/floorvar.tcl
floorPlan -site nc40_dsc -s $PE_X2 $PE_Y2 \
0.0 [expr $cell_height] 0.0 [expr $cell_height]

generateTracks \
    -m1HOffset 0.0 -m1HPitch 0.132 -m1VOffset 0.066 -m1VPitch 0.132 \
    -m2HOffset 0.0 -m2HPitch 0.132 -m2VOffset 0.066 -m2VPitch 0.132 \
    -m3HOffset 0.0 -m3HPitch 0.132 -m3VOffset 0.066 -m3VPitch 0.132 \
    -m4HOffset 0.0 -m4HPitch 0.132 -m4VOffset 0.066 -m4VPitch 0.132 \
    -m5HOffset 0.0 -m5HPitch 0.132 -m5VOffset 0.066 -m5VPitch 0.132 \
    -m6HOffset 0.0 -m6HPitch 0.264 -m6VOffset 0.066 -m6VPitch 0.264 \
    -m7HOffset 0.0 -m7HPitch 0.792 -m7VOffset 0.066 -m7VPitch 0.792



checkMacroLLOnTrack -useM2M3Track
```

```
set powerGapH 3.838
set powerGapV 3.838
cutRow -selected -topGap ${powerGapH} -bottomGap ${powerGapH} \
-leftGap ${powerGapV} -rightGap ${powerGapV}
deselectAll

addEndCap -preCap LDL_POWERSTOPL -postCap LDL_POWERSTOPR \
    -prefix LDL_POWERSTOP_

source "./script/PE_pin.tcl"

saveDesign ${DESIGN}_floorplan.enc

set outputVerilog ${DESIGN}_floorplan.v
set outputDef     ${DESIGN}_floorplan.def

source "./script/save.tcl"
```

ここでは、floorval.tcl で値を設定している。

```
set cell_height         1.188
set PE_W       166
set PE_H       60
set PE_X1 0.00
set PE_Y1 0.00
set PE_X2 [expr $cell_height * $PE_W]
set PE_Y2 [expr $cell_height * $PE_H]

set powerGapH 3.838
set powerGapV 3.838

set baseAX [expr 0.132 * 1002 + 0.066]
set offsetX [expr 35.640 + 12.936 * 2]
set strideX [expr $PE_X2 + $offsetX]

set baseAY [expr 50 * $cell_height]
set spaceY [expr 40 * $cell_height]
set strideY [expr 2 * $spaceY + $PE_Y2]

set PE_ARRAY_X1 0.00
set PE_ARRAY_Y1 0.00
set PE_ARRAY_X2  [expr 2 * $baseAX + 8 * $strideX]
set PE_ARRAY_Y2  [expr 10 * $cell_height + 10 * $strideY]
```

後のアレイにも使えるように、値を設定している。やや細長いがこれは当初１／２区画で設計していた影響が残っている。それぞれの値はグリッドに載るように 0.132 の倍数で設定されている。

次にピンの設定を行う。東西南北が変わってしまっていて、コメントがおかしいが、EAST 0, WEST 1, NORTH 2, SOUTH 3 だと思う。M4L, M3L を使っている。

```
setPinConstraint -cell PE -spacing ${pinspace} -pin  IN_A_NORTH[0]            -layer {M4L} -edge 0
```

などと設定する。最後に

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

を入れ、ピンで DRC エラーが出ないようにする。

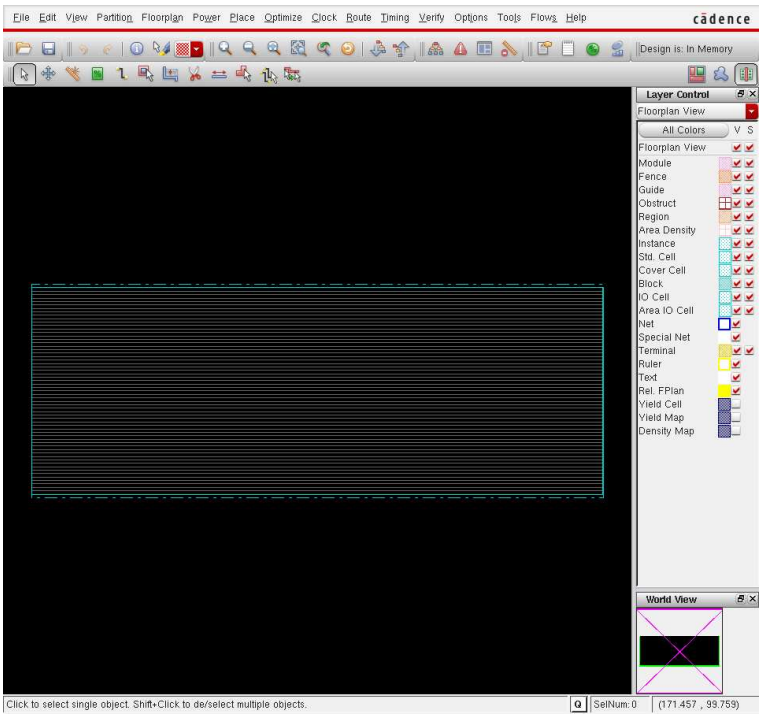

図 1: フロアプラン

2.4 電源配線

一度、SoCE を立ち上げ直してから電源配線を行う。SoCE を立ち上げ直すために、同様の初期設定を行うが、

```
set DESIGN PE
source "./script/env.tcl"
source "./conf/VDEC_soce.conf"
set inputLEF "{./lib/lef_proute}"
set inputVerilog "./${DESIGN}_floorplan.v"
set inputSDC     "./sdc/${DESIGN}.sdc"
set inputDef     "./${DESIGN}_floorplan.def"
source "./script/load_def.tcl"
source "./script/proute.tcl"
```

今度は、lef_proute を使う所がミソである。読み出すファイルは先程フロアプランで作った def である。
proute.tcl は以下のようになっている。
最初にオングリッドのために snap.tcl を実行し、次に floorval を設定しておく。

```
## MAKE RAIL ##
source "./script/snap.tcl"
source "./script/floorvar.tcl"

setAddRingOption -extend_stripe_search_distance 0.0
setViaGenOption -optimize_cross_via 1
...

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
```

この部分は内藤電誠からもらったもので、補助解説書と同じである。

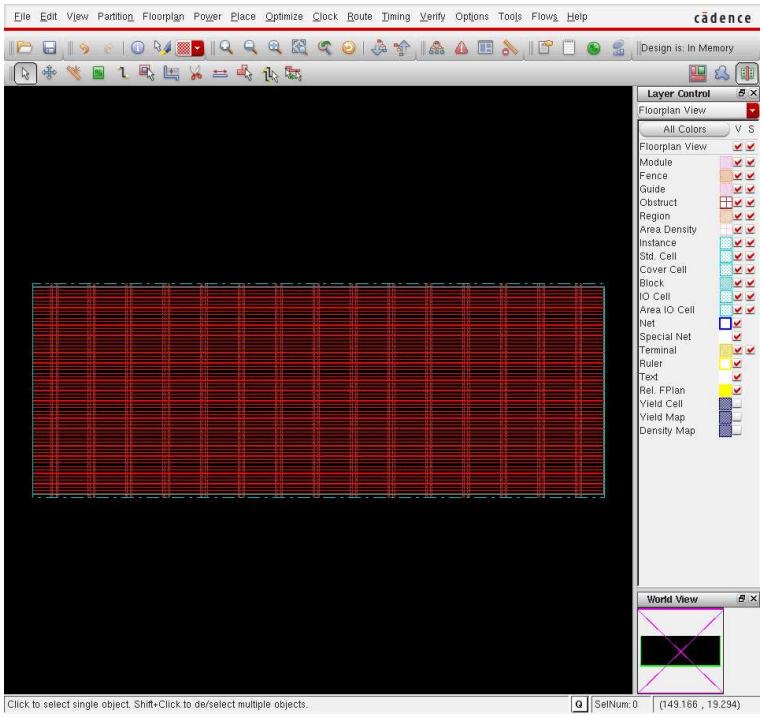

図 2: 電源配線

ここで再び SoCE を終了する。

2.5 配置

次に SoCE を再び立ち上げる。これ以降は落す必要はない。

```
set DESIGN PE
source "./script/env.tcl"
source "./conf/VDEC_soce.conf"
set inputVerilog "./PE_proute.v"
set inputSDC     "./sdc/PE.sdc"
set inputDef     "./PE_proute.def"
set inputLEF "{./lib/lef}"
source "./script/load_def.tcl"
```

ここは、lef を元に戻して同様の設定を行い、proute で保存した def を読み込む。
次にフロアプランに従って配置を行う。まずは、

```
source "./script/PE_blockage.tcl"
source "./script/PE_pin.tcl"
```

で、ブロッケージとピンを宣言する。blockage は、

```
source "./script/floorvar.tcl"

createObstruct $PE_X1 $PE_Y1 [expr $PE_X1 + 1.919] $PE_Y2
createObstruct $PE_X2 $PE_Y1 [expr $PE_X2 - 1.919] $PE_Y2

for {set x [expr $PE_X1 + 6.535]} \
    {$x < $PE_X2} \
    {set x [expr $x + [expr 0.264 * 49]] } {
        createObstruct [expr $x - 0.594 / 2] $PE_Y1 \
            [expr $x + 0.594 / 2] $PE_Y2
        createObstruct [expr $x - 0.594 / 2 + 1.716] $PE_Y1 \
            [expr $x + 0.594 / 2 + 1.716 ] $PE_Y2
    }
```

で、端と、電源ストリップの下に作る。ピンは、フロアプランで使ったのと同じのを使う。
次にいよいよ配置を行う。

```
source "./script/place.tcl"
```

これは補助説明書で使ったのと全く同じであるのでここでは省略する。配置中の trial route の結果は検討しなければならない。ここでは

```
Usage: (41.8%H 36.4%V) = (1.075e+05um 7.727e+04um) = (162250 65044)
Overflow: 89 = 42 (0.23% H) + 47 (0.26% V)
```

なので、行けそうだと判断する。

2.6 配線

次は nanoroute と ecoroute で配線を行う。これは説明書にあった配線のスクリプトが nanoroute, ecoroute を直接起動するものであったのを、キム兄が SoCE からのコマンドとして整理してくれたものである。ここでは詳細な解説は省略する。

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

この部分は小規模な PE といえども多少の時間が掛かる。
次にアンテナ対策のリピータを入れる。

```
source "./script/PE_repeater.tcl"
```

この中身は以下の記述である。

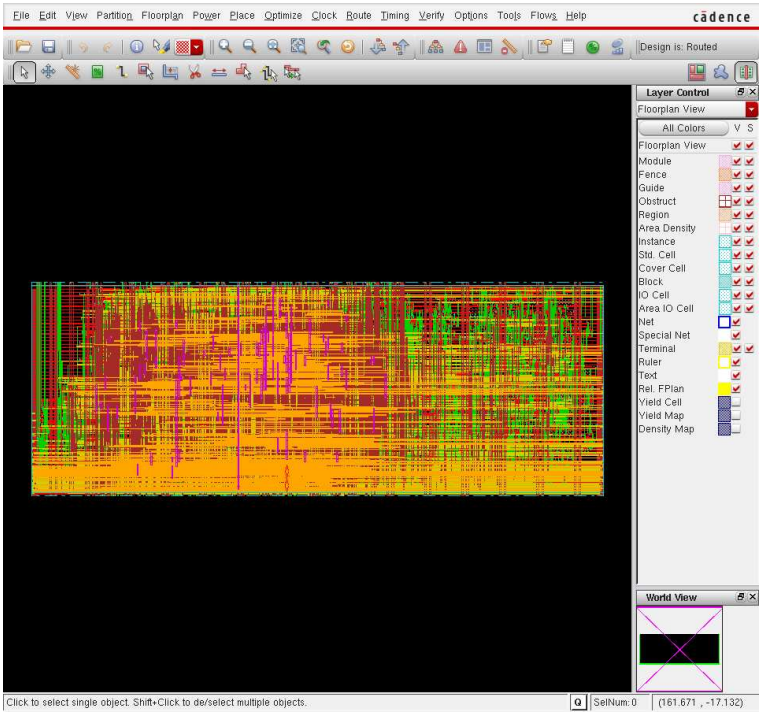
File Edit View Partition Floorplan Power Place Optimize Clock Route Timing Verify Options Tools Flows Help
cadence
Design is: Routed
Layer Control
Floorplan View
All Colors
V S
Floorplan View
Module
Fence
Guide
Obstruct
Region
Area Density
Instance
Std. Cell
Cover Cell
Block
IO Cell
Area IO Cell
Net
Special Net
Terminal
Ruler
Text
Rel. FPlan
Yield Cell
Yield Map
Density Map
World View
Click to select single object. Shift+Click to de/select multiple objects.
Q
SelNum: 0
(161.671 , -17.132)

図 3: 配置

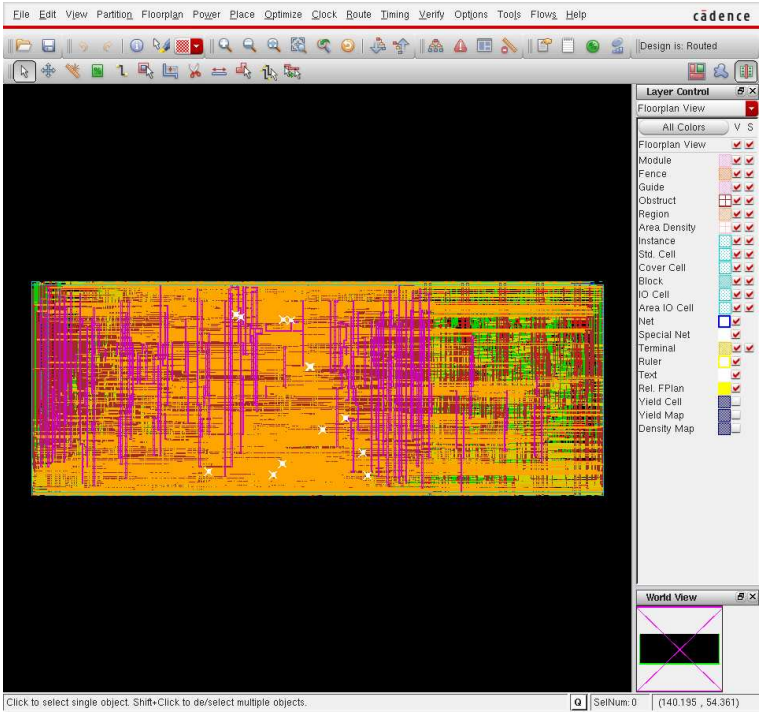
File Edit View Partition Floorplan Power Place Optimize Clock Route Timing Verify Options Tools Flows Help
cadence
Design is: Routed
Layer Control
Floorplan View
All Colors
V S
Floorplan View
Module
Fence
Guide
Obstruct
Region
Area Density
Instance
Std. Cell
Cover Cell
Block
IO Cell
Area IO Cell
Net
Special Net
Terminal
Ruler
Text
Rel. FPlan
Yield Cell
Yield Map
Density Map
World View
Click to select single object. Shift+Click to de/select multiple objects.
Q
SelNum: 0
(140.195 , 54.361)

図 4: 配線

```
setPreference ConstraintUserXOffset 0.066
setPreference BlockSnapRule 2
setPreference ConstraintUserXGrid 0.132
setPreference ConstraintUserYGrid 0.132

insertRepeater -rule "script/ibpo.rule" -selNetFile "script/PE_repair_net.file" -postRoute
```

最初の部分はグリッドに載せるための設定である。ibpo.rule にリピータを入れるルールを指定する。これはここでは、400um の配線長に対して挿入している。

```
SetBufferDrivingStrength LDH_BUF_2 400 0.1
SetDefaultDrivingStrength 50 0.1
```

また、挿入すべき所が分かっていれば、これを PE_repair_net.file で指定する。ここでは以下のネットを指定する。これは階層的に接続した際にアンテナエラーを出すもので、実は上位階層の配線をやってはじめて発見できる。

```
IN_B_SOUTH[12]
\end{document}
```

バッファを挿入したら再び ecoroute を行う。

```
\begin{verbatim}
source "./script/ecoroute.tcl"
```

これでほぼエラーのない配線が出来上がる。

2.7 フィラー埋め、メタル埋め

```
deleteAllRouteBlks
source "./script/cdfill.tcl"
deleteObstruction -all
source "./script/filler_no8GC1.tcl"
```

ここで、2回に分けてフィラーを埋めている。cdfill.tcl の中身は、説明書と同じである。ストラップの下と端に 8GC1 というフィラーが入るとエラーになるため、一度埋めた後にブロッケージを解除して、再び制約を掛けてフィラー埋めをしている。
最後にメタル埋めを行う。これは説明書と同じである。

```
source "./script/metal_fill.tcl"
```

レイアウトができたら検証を行う。これも説明書と同じであり、geometory, connectivity, anntenna, drc がチェックされ、それぞれの rpt ファイルに吐き出される。

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

これでエラーがないことを確認して、gds を吐き出す。

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

まず、説明書で吐き出す標準的な、PE.gds, PE.def.gz, PE.lvs.v が吐き出される。
この中の最後の部分が、

```
lefOut ${DESIGN}.LEF
write_sdf -version 2.1 ${DESIGN}.sdf
extractRC -outfile ${DESIGN}.cap
rcOut -spef ${DESIGN}.spef
```

となっており、これで、LEF, sdf, cap, spef を吐き出す。今回はこれで十分である。

3 PE ARRAY のフロー

3.1 概観

PE ARRAY のフローも概観は PE と同じく、フロアプラン、電源配線、配置、配線、フィラー、メタル埋めのプロセスを辿るが、エラーを避けるために、ブロッケージを張ったり、これを解除したりを繰り返す必要があるので、この辺でやや面倒が増える。

3.2 準備

今回のレイアウトでは、PE の配線に全ての層を使っている。このため、上の PE ARRAY の配線時には、PE の LEF に禁止領域 OBS を挿入する必要がある。これをやらないと、PE 内部の縁付近にメタルや、配線が侵入してエラーを生じてしまう。このためにキム兄が作った ruby 記述が make_obs.rb である。

```
ruby make_obs.rb PE.LEF > tmp
```

で、

```
OBS
  LAYER M2L DESIGNRULEWIDTH 0.066 ;
  RECT 0.000 0.000   197.2080    73.6560 ;
  LAYER M3L DESIGNRULEWIDTH 0.066 ;
  RECT 0.000 0.000   197.2080    73.6560 ;
  LAYER M4L DESIGNRULEWIDTH 0.066 ;
  RECT 0.000 0.000   197.2080    73.6560 ;
  LAYER M5L DESIGNRULEWIDTH 0.066 ;
  RECT 0.000 0.000   197.2080    73.6560 ;
END
```

を発生してくれるので、これを PE.LEF の最後の END の前に挿入する。

3.3 フロアプラン

以下がフロアプランのスクリプトである。lib/PE.LEF も読み込んでやる以外は PE と同じである。

```
set DESIGN PE_ARRAY
source "./script/env.tcl"
source "./conf/VDEC_soce.conf"
set inputLEF { ./lib/lef ./lib/PE.LEF}
set inputVerilog "./vnet/${DESIGN}-compile.v"
set inputSDC     "./sdc/${DESIGN}.sdc"
```

```
source "./script/load_design.tcl"
source "./script/PE_ARRAY_floorplan.tcl"
```

ポイントは PE_ARRAY_floorplan.tcl でこの最初に floorvar.tcl でサイズを指定している。この中身は、

```
set cell_height         1.188

set PE_W       166
set PE_H       60

set PE_X1 0.00
set PE_Y1 0.00
set PE_X2 [expr $cell_height * $PE_W]
set PE_Y2 [expr $cell_height * $PE_H]

set powerGapH 3.838
set powerGapV 3.838

set baseAX [expr 0.132 * 1002 + 0.066]
set offsetX [expr 35.640 + 12.936 * 2]
set strideX [expr $PE_X2 + $offsetX]

set baseAY [expr 50 * $cell_height]
set spaceY [expr 40 * $cell_height]
set strideY [expr 2 * $spaceY + $PE_Y2]

set PE_ARRAY_X1 0.00
set PE_ARRAY_Y1 0.00
set PE_ARRAY_X2  [expr 2 * $baseAX + 8 * $strideX]
set PE_ARRAY_Y2  [expr 10 * $cell_height + 10 * $strideY]
```

で PE ARRAY のサイズを指定している。ここを変えると PE の間隔が変わる。
また、中で、PE の配置を行っているのが、place_pe.tcl である。この中身は、

```
set p7x $baseAX
set p6x [expr $baseAX + $strideX]
set p5x [expr $baseAX + 2 * $strideX]
set p4x [expr $baseAX + 3 * $strideX]
set p3x [expr $baseAX + 4 * $strideX]
set p2x [expr $baseAX + 5 * $strideX]
set p1x [expr $baseAX + 6 * $strideX]
set p0x [expr $baseAX + 7 * $strideX]

set p0y $baseAY
set p1y [expr $baseAY + $strideY]
set p2y [expr $baseAY + 2 * $strideY]
set p3y [expr $baseAY + 3 * $strideY]
```

```
set p4y [expr $baseAY + 4 * $strideY]
set p5y [expr $baseAY + 5 * $strideY]
set p6y [expr $baseAY + 6 * $strideY]
set p7y [expr $baseAY + 7 * $strideY]
set p8y [expr $baseAY + 8 * $strideY]
set p9y [expr $baseAY + 9 * $strideY]
placeInstance COL_0__PE_7_ $p7x $p0y -fixed
placeInstance COL_0__PE_6_ $p6x $p0y -fixed
placeInstance COL_0__PE_5_ $p5x $p0y -fixed
placeInstance COL_0__PE_4_ $p4x $p0y -fixed
placeInstance COL_0__PE_3_ $p3x $p0y -fixed
placeInstance COL_0__PE_2_ $p2x $p0y -fixed
placeInstance COL_0__PE_1_ $p1x $p0y -fixed
placeInstance COL_0__PE_0_ $p0x $p0y -fixed
placeInstance COL_1__PE_7_ $p7x $p1y -fixed
placeInstance COL_1__PE_6_ $p6x $p1y -fixed
...
```

で場所を指定している。これは、多分 tcl が達者な人ならば tcl で書けたのかもしれないが、この辺の技術が未熟なので、プログラム placegen.c で生成した。

ここまで実行すると、以下のようになる。

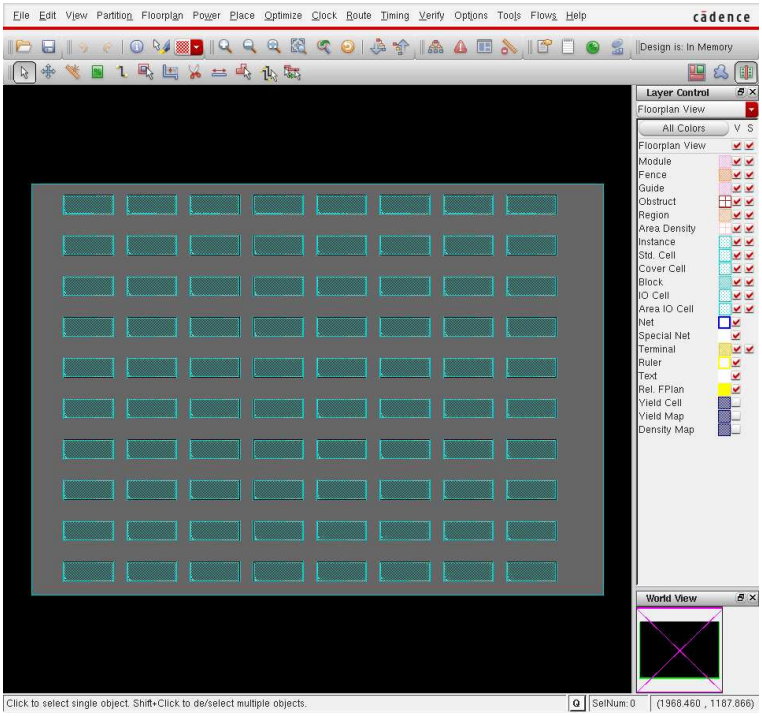

図 5: PE ARRAY フロアプラン

ここで一度終了する。

3.4 電源配線

再び SoCE を立ち上げる。

```
set DESIGN PE_ARRAY
source "./script/env.tcl"
source "./conf/VDEC_soce.conf"
set inputLEF { ./lib/lef_proute ./lib/PE.LEF}
set inputVerilog "./${DESIGN}_floorplan.v"
set inputSDC     "./sdc/${DESIGN}.sdc"
set inputDef     "./${DESIGN}_floorplan.def"
source "./script/load_def.tcl"
source "./script/proute.tcl"
```

LEF を電源配線用に入れ替えて電源配線を行う。スクリプト自体は、PE と同じである。

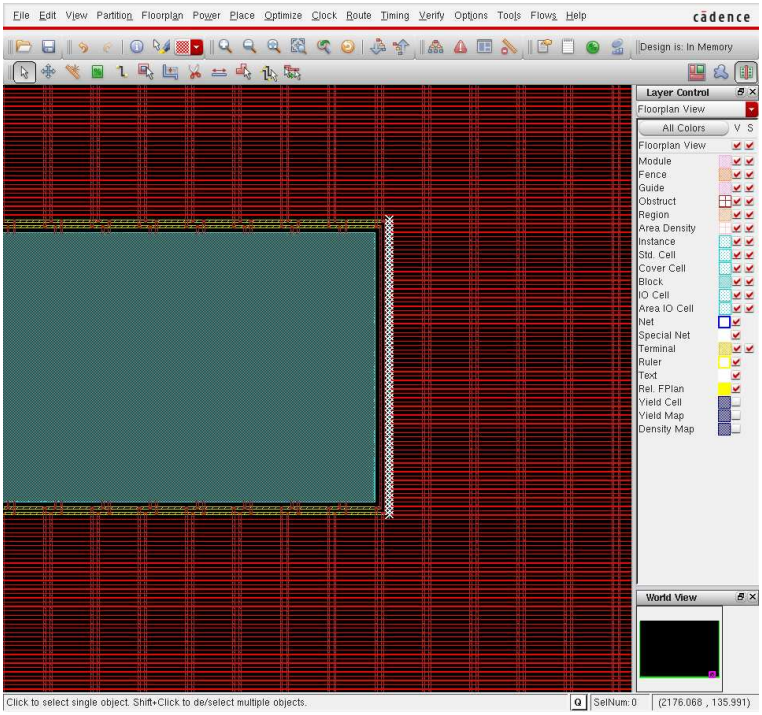

図 6: PE ARRAY 電源配線

3.5 配置

再び SoCE9.1 で立ち上げる。

```
set DESIGN PE_ARRAY
source "./script/env.tcl"
source "./conf/VDEC_soce.conf"
set inputVerilog "./${DESIGN}_proute.v"
set inputSDC     "./sdc/${DESIGN}.sdc"
set inputDef     "./${DESIGN}_proute.def"
set inputLEF {./lib/lef ./lib/PE.LEF}
source "./script/load_def.tcl"
source "./script/${DESIGN}_blockage.tcl"
source "./script/${DESIGN}_pin.tcl"
```

```
source "./script/place.tcl"
```

今までとの違いは、ブロッケージを入れる点である。
まず、電源配線の下に入れる。

```
source "./script/floorvar.tcl"



createObstruct $PE_ARRAY_X1 $PE_ARRAY_Y1 [expr $PE_ARRAY_X1 + 1.919] [expr $PE_ARRAY_Y2 + 2* $cell_height]
createObstruct $PE_ARRAY_X2 $PE_ARRAY_Y1 [expr $PE_ARRAY_X2 - 1.919] [expr $PE_ARRAY_Y2 + 2* $cell_height]

 for {set x [expr $PE_ARRAY_X1 + 6.567]} \
    {$x < $PE_ARRAY_X2} \
    {set x [expr $x + [expr 0.264 * 49]] } {
        createObstruct [expr $x - 0.594 / 2] $PE_ARRAY_Y1 \
            [expr $x + 0.594 / 2] [expr $PE_ARRAY_Y2 + 2* $cell_height]
        createObstruct [expr $x - 0.594 / 2 + 1.716] $PE_ARRAY_Y1 \
            [expr $x + 0.594 / 2 + 1.716 ] [expr $PE_ARRAY_Y2 + 2* $cell_height]
    }
```

次に、PE に配置のブロッケージと配線のブロッケージを両方入れる。

```
set eY 2.3
set eX 0.1

 for {set x $baseAX} \
    {$x < [expr $PE_ARRAY_X2 - $baseAX] } \
    {set x [expr $x + $strideX] } {
        for {set y $baseAY} \
        {$y < [expr $PE_ARRAY_Y2 - $baseAY]} \
        {set y [expr $y + $strideY]} {
           createObstruct $x $y \
                        [expr $x + $PE_X2 -$eX] [expr $y + $PE_Y2 +$eY]
           createObstruct [expr $x - $powerGapH - 1.919] [expr $y -$powerGapV] \
                [expr $x - $powerGapH] [expr $y + $PE_Y2 + 2 * $powerGapV]
           createObstruct [expr $x +$PE_X2 + $powerGapH] [expr $y -$powerGapV] \
                [expr $x +$PE_X2 + $powerGapH + 1.919] [expr $y + $PE_Y2 + 2 * $powerGapV]
           createRouteBlk -box $x $y \
                        [expr $x + $PE_X2 -$eX] [expr $y + $PE_Y2 +$eY]
        }
    }
```

Trial route の結果は、以下の感じで Overflow も少ないので、行けると思う。

```
Usage: (2.4%H 5.7%V) = (1.177e+06um 2.278e+06um) = (1782973 1917120)
Overflow: 274 = 16 (0.00% H) + 258 (0.01% V)
```

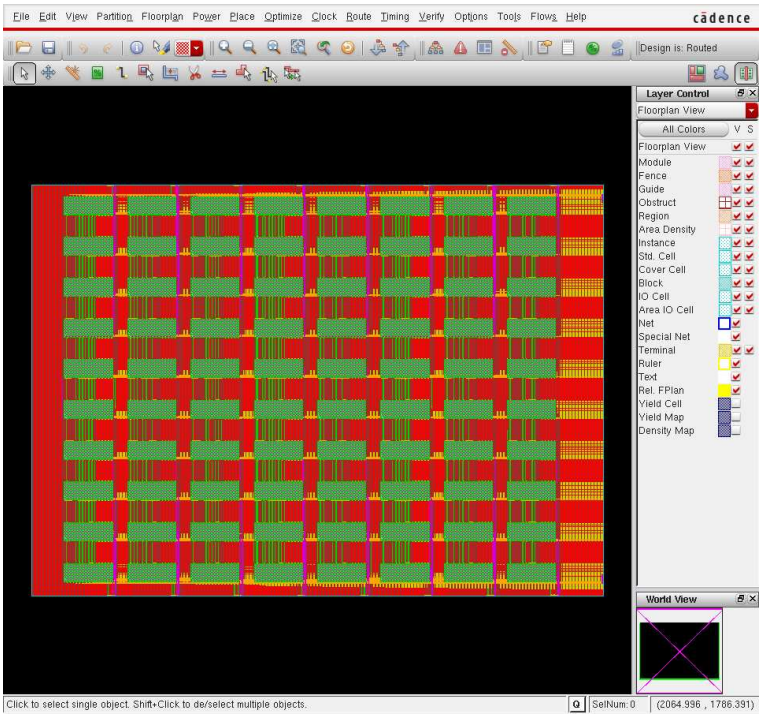

図 7: PE ARRAY 配置後

3.6 配線

そうは言っても、今度は対象が大きいので、何回か ecoroute を回している。多分、後半はなくても良い。このステップは非常に時間がかかるので終わったら save しておく。

```
deleteAllRouteBlks
source "./script/nanoroute.tcl"
source script/repeater.tcl
source "./script/ecoroute.tcl"
source script/repeater.tcl
source "./script/ecoroute.tcl"
source "./script/ecoroute.tcl"
source "./script/ecoroute.tcl"
source "./script/ecoroute.tcl"
set outputDef    "./${DESIGN}_outroute.def"
set outputVerilog "./${DESIGN}_outroute.v"
source script/save.tcl
```

リピータの入れ方は PE と同様なので省略する。

3.7 フィラー、メタル埋め

PE と同様のステップでフィラー、メタル埋めを行う。

```
deleteAllRouteBlks
source "./script/cdfill.tcl"
deleteObstruction -all
```

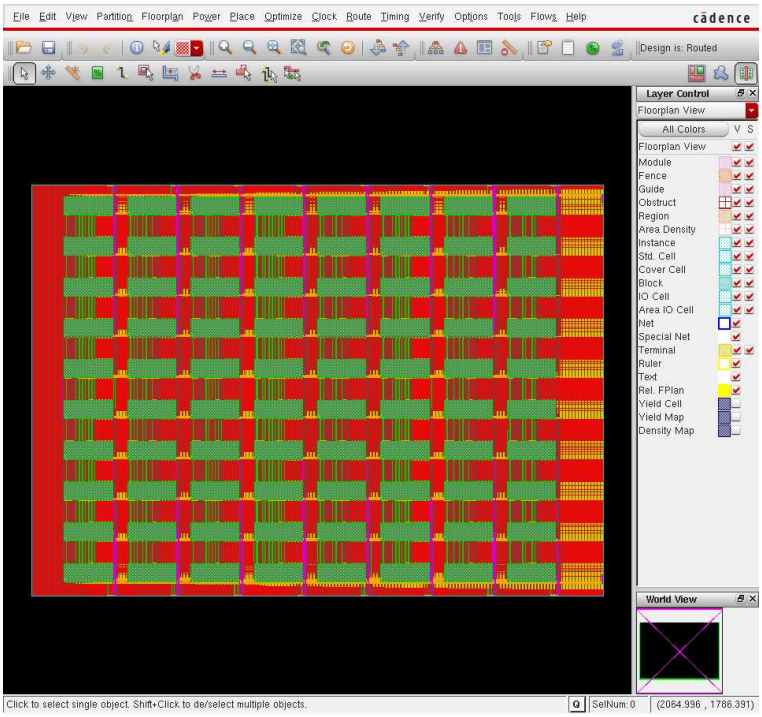

図 8: PE ARRAY 配線後

```
source "./script/filler_no8GC1.tcl"
source "./script/${DESIGN}_blockage.tcl"
source "./script/metal_fill.tcl"
deleteAllRouteBlks
deleteObstruction -all
```

メタルが PE へ侵入せず、配線もまっすぐ入っていればエラーは出ない。

3.8 検証、吐き出し

これも PE と同じである。

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

このレイアウトは、基本的なエラーは出ないのだが、Calibre で孤立 Via が生じる。これは SoCE の自動レイアウトでは解決できないので、人手に頼ることになる。

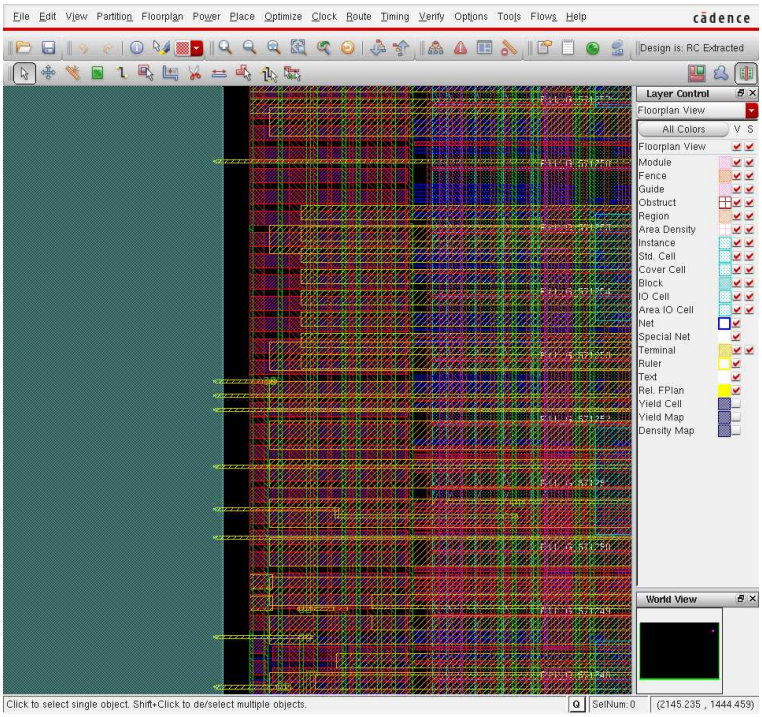

図 9: PE ARRAY フィル後