

はじめてのLSI設計 ～HDLでデジタル回路編

秋田純一(金沢大)/MakeLSI:

Contents

- ☑ HDLからのLSI設計の流れ
- ☑ はじめてのLSI設計（HDLでデジタル設計）
 - ☑ HDLの入手
 - ☑ Qflowで半手動設計

HDL→LSIレイアウトへの道のり

HDLで書いた論理回路

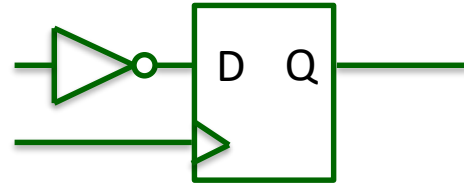
```
always @(posedge ck) begin
  if (rst == 1) cnt <= 0;
  else cnt <= cnt + 1;
end
```

※HDL=Hardware Description Language
(ハードウェア記述言語)

※最近ではC言語から論理合成
(高位合成)も実用化されつつある

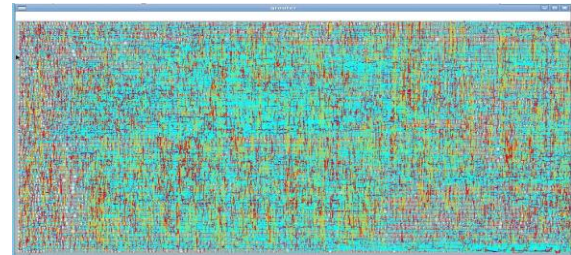
論理合成 ツール

ネットリスト(回路図)

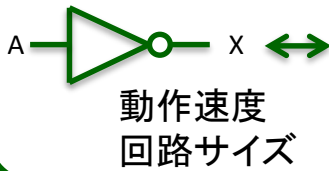


配置配線 ツール

レイアウト図



ライブラリ

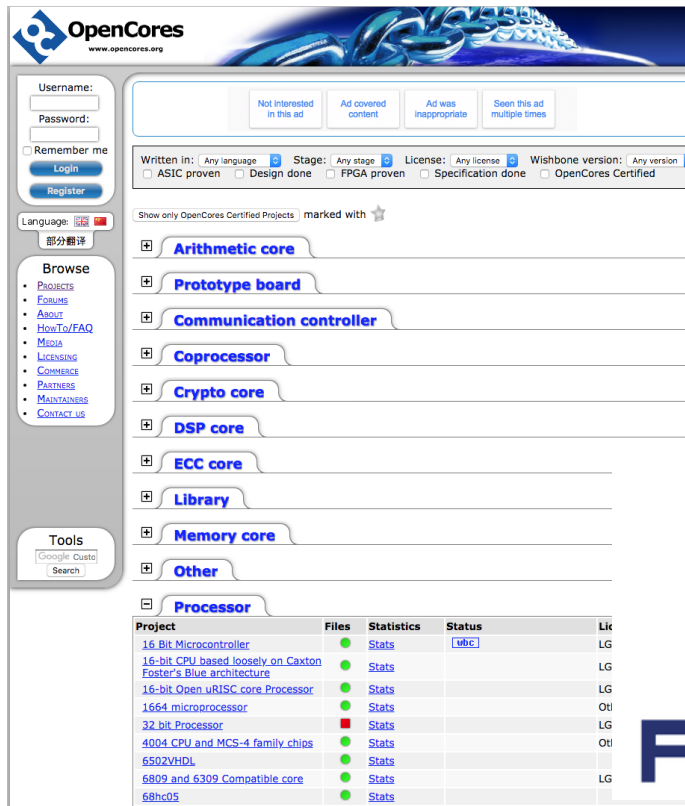


制約条件

- 動作速度
- 回路サイズ など

まずはHDLを入手

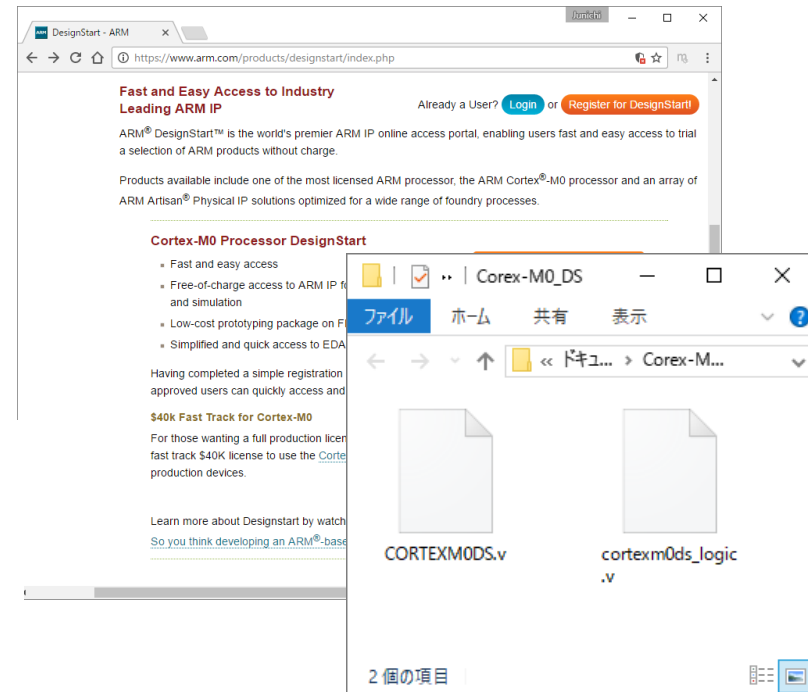
☑購入(IP)／OpenSourceHW／その他



OpenCores website showing various IP cores categorized by type:

- Arithmetic core
- Prototype board
- Communication controller
- Coprocessor
- Crypto core
- DSP core
- ECC core
- Library
- Memory core
- Other
- Processor

Project	Files	Statistics	Status	License
16 Bit Microcontroller	●	Stats	LG	UBC
16-bit CPU based loosely on Cxton Foster's Blue architecture	●	Stats	LG	
16-bit Open uRISC core Processor	●	Stats	LG	
1664 microprocessor	●	Stats	LG	
32 bit Processor	●	Stats	LG	
4004 CPU and MCS-4 family chips	●	Stats	LG	
6502VHDL	●	Stats	LG	
6809 and 6309 Compatible core	●	Stats	LG	
68hc05	●	Stats	LG	



ARM DesignStart website showing information about the Cortex-M0 processor and DesignStart access. A file explorer window shows the following files:

- CORTEXMODS.v
- cortexm0ds_logic.v



ARM Cortex-M0 DesignStart
(評価用は無償でHDL・量産時に課金)

RISC-V
オープンソース、
階層的なISA(命令セット)
各種実装あり

今回のお題: poyo-v

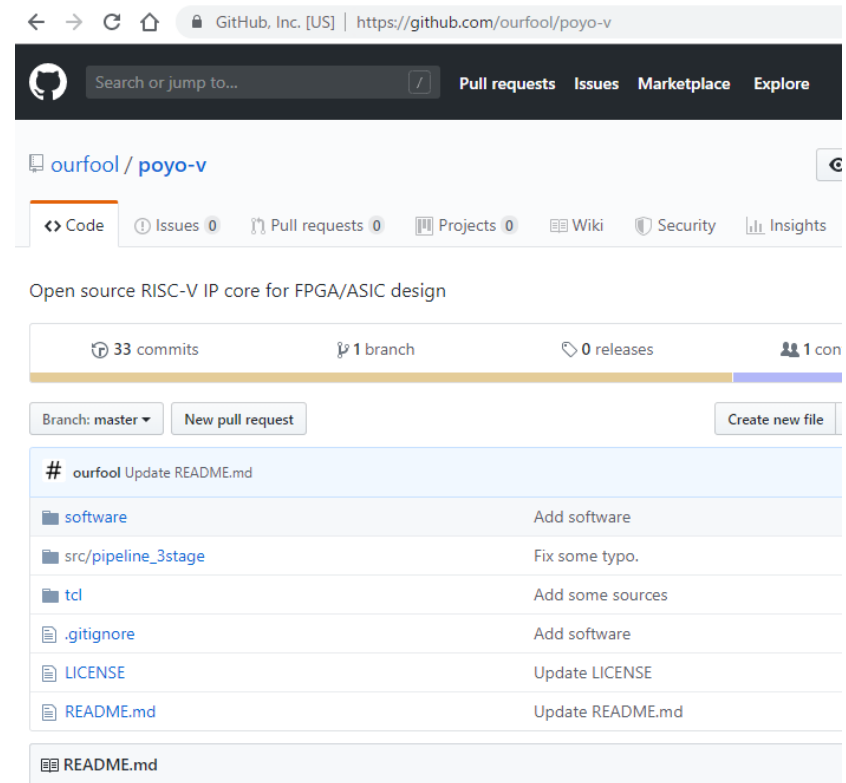
✓ ourfool氏によるRISC-Vの実装の一つ

✓ 命令セット: RV32I
(RISC-Vの32bit整数版)

✓ FPGAでの動作例あり

✓ 記述がとてもシンプル

✓ ここからforkした
右のリポジトリの中の
src/pipeline_3stage/
design_blink 内を
使います
(source.blink内にDL済み)



<https://github.com/akita11/poyo-v>

今回使うツール: Qflow

☑ Tim Edward氏作の上位設計ツール群

<http://opencircuitdesign.com/qflow/>

☑ 論理合成＋配置配線＋レイアウト(確認編集)
(配置配線以外は既存のものを流用)

☑ Open Source Software

☑ オープンソースなライブラリつき
(Oklahoma State Univ.のosu035/osu05)

HDLからのLSI設計の一般論

✓半自動(全自動ではない)

✓論理合成、配置配線の各ステップで、
手動設定・試行錯誤が入ることが多い

✓設定パラメータがけっこうだいじ

✓レイアウトの縦横比

✓論理ゲートセルの配置密度(高すぎると配線できない)

✓電源配線をおく密度

initial density	rowsep	未配線	実行時間(place,route)[分]
0.6	0.6	0	8,12
0.6	0.7	0	8,12
0.8	0.2	0	10,9
0.6	0.2	26(Stage2)	9,7
0.8	0.1	156(Stage2)	9,5
1.0	0.1	649(Stage2)	9,7
0.8	0.05	263(Stage2)	11,11
0.6	0.1	43(Stage2)	12,10
0.9	0.2	268(Stage2)	8,8

パラメータによっては配線が終わらないなど

Qflowでの設計: Qflowの準備

☑ Qflowをインストール

☑ <https://scrapbox.io/makelsi/>の
「Qflowのインストール」を参考に

☑ Ubuntu Linux 16.04 (仮想マシンやAWSでもOK)

☑ (インストール済みの共用サーバもあり)

☑ GUI版も入れておくとベター? (今回は使わない)

☑ <https://scrapbox.io/makelsi/>の
「Qflowのインストール」の最後のところを参考に

☑ ただしQflowのバージョンが上がって、
設計パラメータも変えないといけなさそう...

Qflowでの設計：準備

- ✓ Ubuntuにログイン

- ✓ プロジェクトのディレクトリをつくる

 - ✓ 例：~/poyo-v/

- ✓ その中に作業用ディレクトリ(3つ)をつくる

 - ✓ “source”, “layout”, “synthesis”

- ✓ “source”の中に、ファイルー式をコピーする

 - ✓ souce.poyov_blinkにあるファイルー式すべて

 - ✓ `$ cp (上記path) /* ~/poyo-v/source/`

Qflowでの設計：論理合成

✓プロジェクトディレクトリで以下を実行

```
$ qflow synthesize poyov_blink
```

※少し時間がかかる

Qflowでの設計：設定（その1）

☑ プロジェクトディレクトリのproject_var.shで
“initial_density” を変更



```
project_var.sh - emacs@mineda.anagix.com
File Edit Options Buffers Tools Sh-Script Help

# set yosys_debug =
# set abc_script =
# set nobuffers =
# set nofanout =
# set fanout_options = "-l 200 -c 30"

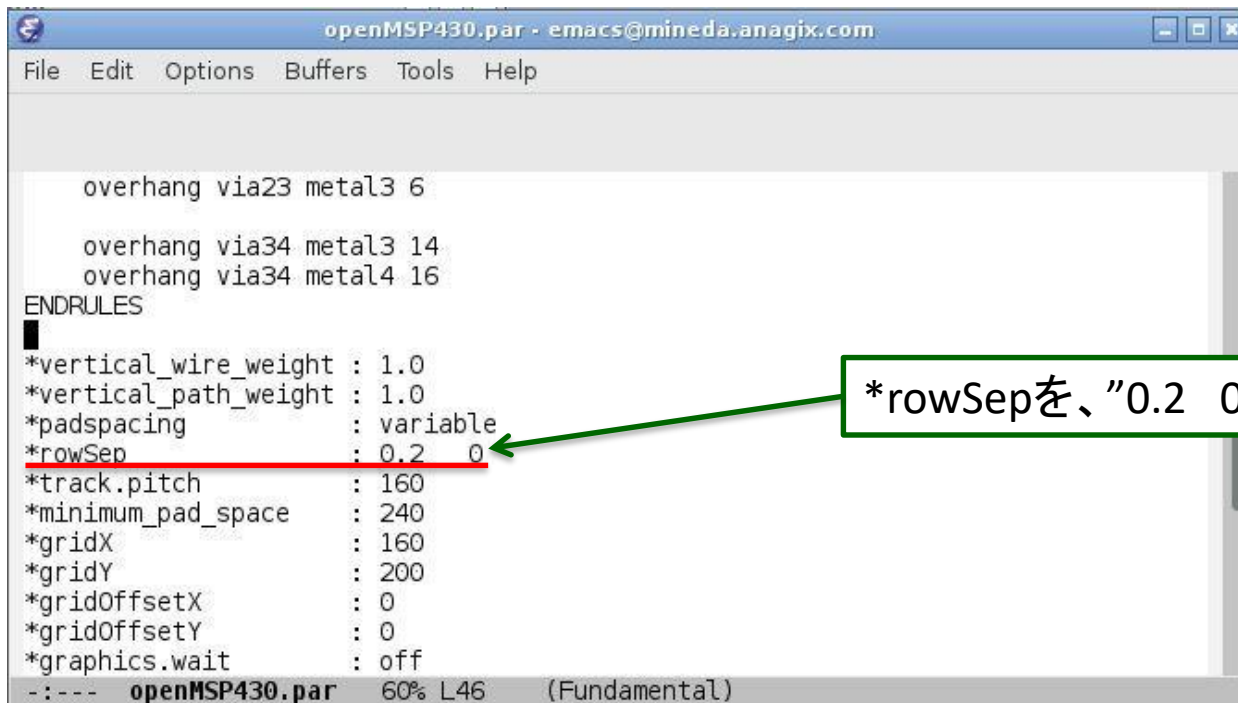
# Placement command options:
# -----
# set initial_density =
# set graywolf_options =
# set addspacers_options = "-stripe 5 150 PG"

# Router command options:
# -----
# set route_show =
```

先頭の#をはずし、0.8に設定
set initial_density = 0.8

Qflowでの設計：設定（その2）

✓ layout/poyov_blink.parで
“*rowSep”を変更



```
openMSP430.par - emacs@mineda.anagix.com
File Edit Options Buffers Tools Help

overhang via23 metal3 6
overhang via34 metal3 14
overhang via34 metal4 16
ENDRULES
*vertical_wire_weight : 1.0
*vertical_path_weight : 1.0
*padspacing : variable
*rowSep : 0.2 0
*track.pitch : 160
*minimum_pad_space : 240
*gridX : 160
*gridY : 200
*gridOffsetX : 0
*gridOffsetY : 0
*graphics.wait : off
-:--- openMSP430.par 60% L46 (Fundamental)
```

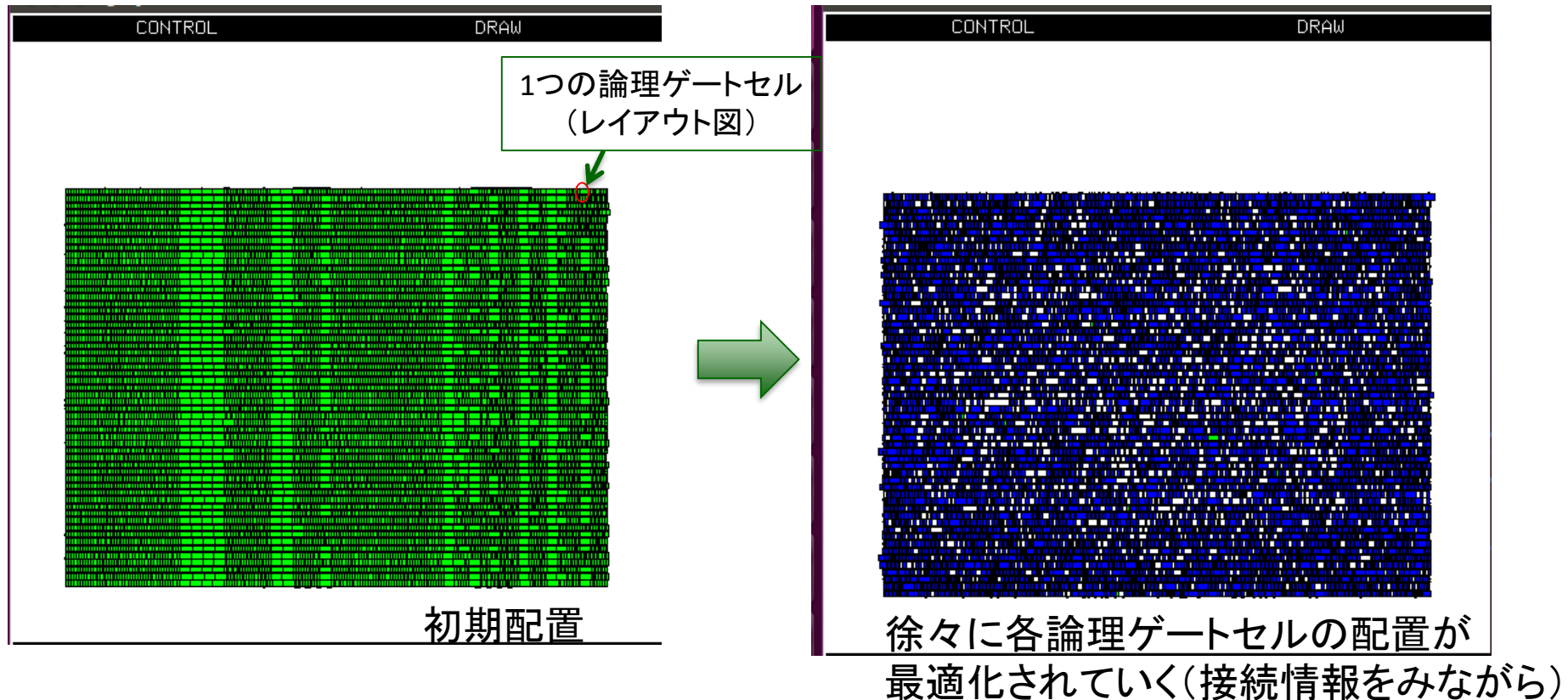
*rowSepを、“0.2 0”に変更

Qflowでの設計: 配置

☑ プロジェクトディレクトリで以下を実行

```
$ qflow place poyov_blink
```

※けっこう時間がかかる

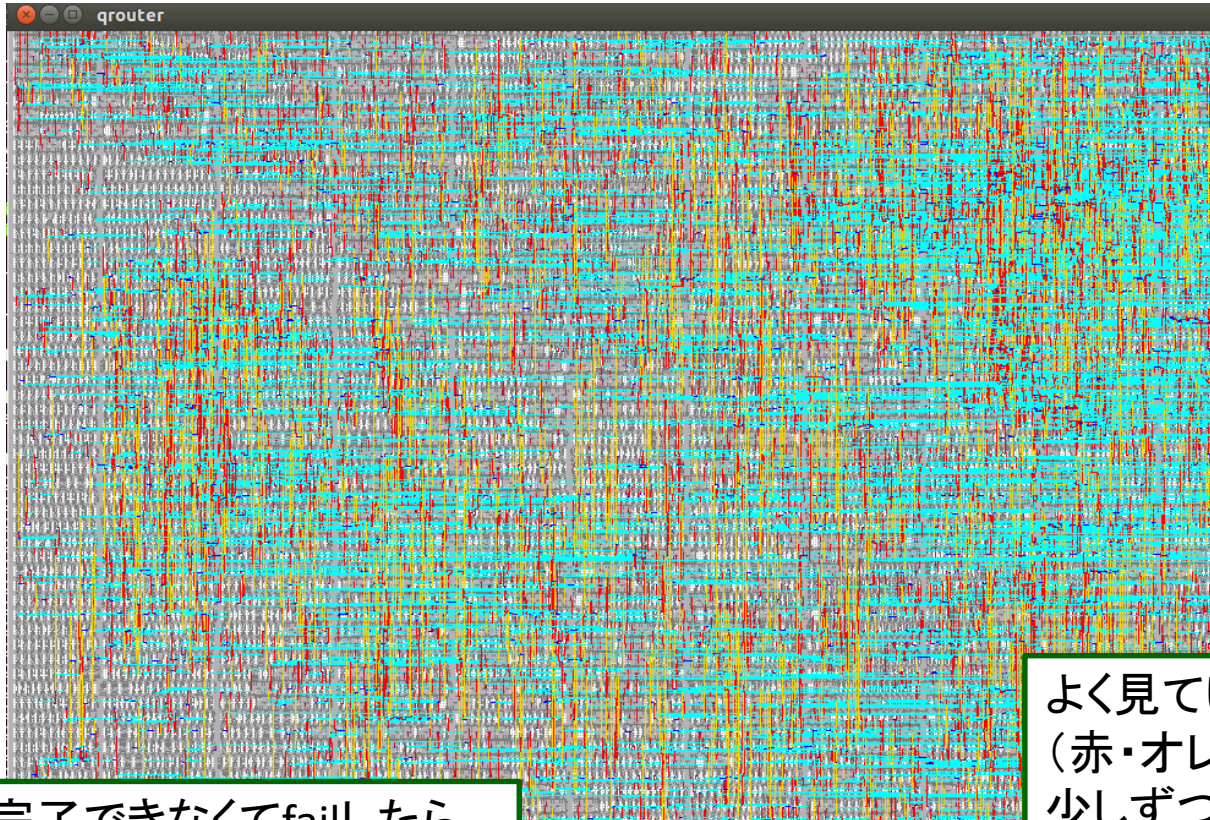


Qflowでの設計：配線

✓プロジェクトディレクトリで以下を実行

```
$ qflow route poyov_blink
```

※けっこう時間がかかる



配線が完了できなくてfailしたら、
パラメータを変えて再チャレンジ

よく見ていると、セル間の配線
(赤・オレンジ等の線)が
少しずつ進んでいる

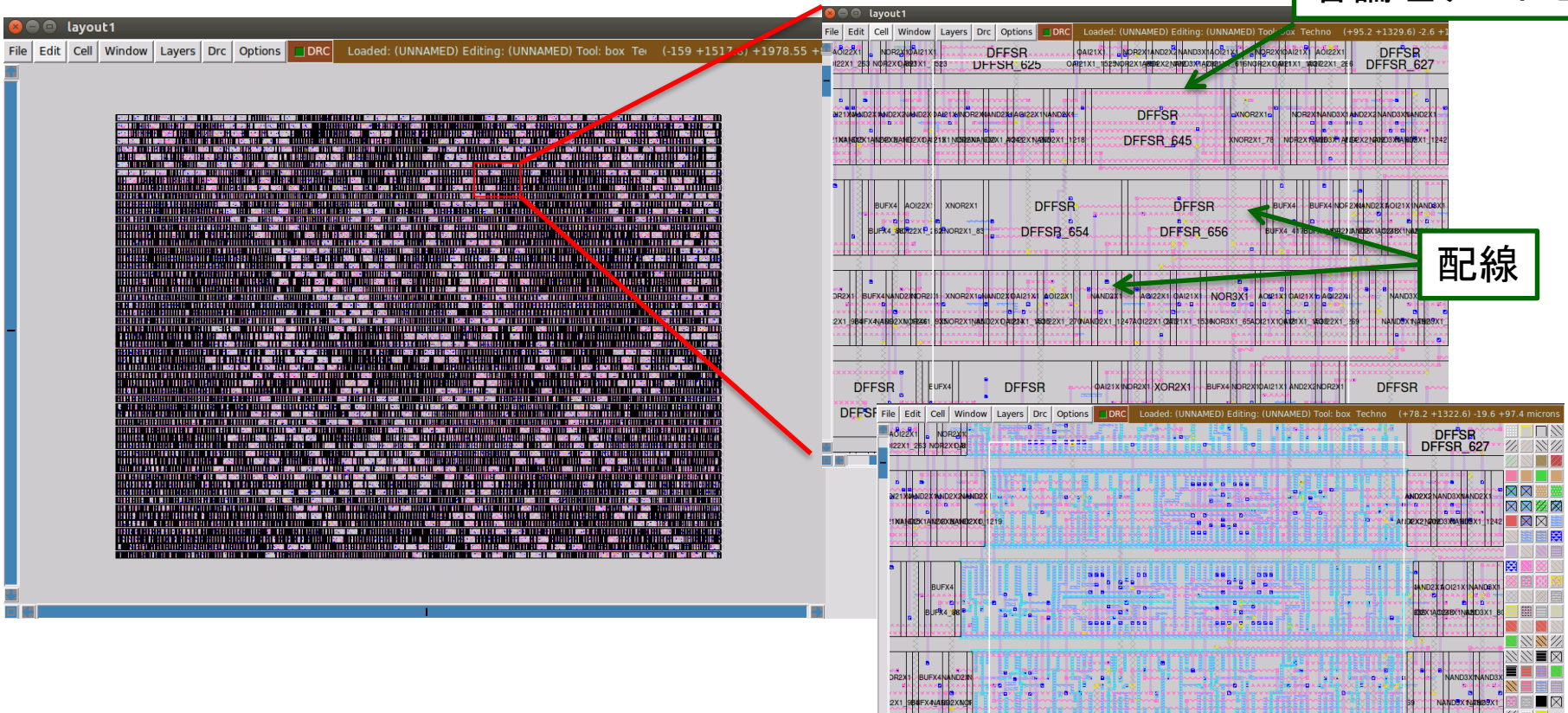
Qflowでの設計:レイアウトの確認

☑プロジェクトディレクトリで以下を実行

```
$ qflow display poyov_blink
```

各論理ゲートセル

配線



Qflowでの設計：最終レイアウト(1)

- ✓ まず論理ゲート(スタセル)の、トランジスタを含む実体レイアウトのデータ(*.mag)を作成
- ✓ プロジェクトdir→layoutに移動し、以下を実行(これでスタセルの*.magがdigital/以下に作成される)

コマンドライン

```
% mkdir digital  
% cp .magicrc digital  
% cd digital  
% magic
```



magicのコマンドウィンドウ

```
> gds read /usr/local/share/qflow/tech  
    /osu035/osu035_stdcells.gds2 (1行で)  
> writeall force  
> quit
```

- ✓ .magicrcには“addpath digital”が加わっているはずなので確認(スタセルmagの検索パス)

Qflowでの設計：最終レイアウト(2)

✓ 次にトップ回路のレイアウト (*.mag) の作成

✓ cd .. でプロジェクトdir/layoutに戻り、magic を起動

magicのコマンドウインドウ

```
> def read <トップ回路名>.def    ※トップ回路名=poyov_blink  
> writeall force <トップ回路名>
```

※スタセルLEFは明示的に読み込まず (digital/以下が自動で読まれる)、
writeallでトップ回路.magのみ書き込むのがポイント

✓ これで<トップ回路名.mag>が作られる。

✓ 再度magicを、トップ回路magを指定して起動

✓ % magic <トップ回路名.mag>

✓ (スタセルLEFはdigital/以下 (=実体) が自動で読み込まれる)

✓ (write gdsすればGDS形式でも保存できる)

poyo-vの設計結果

- ☑使用ライブラリ: OSU 0.35um
- ☑ゲート数: 1万弱
(log/synth.logの"Number of Cells"より)
- ☑レイアウト: 約2000[um] x 1300[um]
(log/place.logの"cell width/height"より)
- ☑動作速度: (要調査)

シミュレーションしてみる

✓ sourceへ移動

✓ シミュレーション実行ファイルの生成

```
$ iverilog -f poyov-sim.v
```

✓ シミュレーションの実行

```
$ vvp a.out
```

✓ ずっとシミュレーションが続くので、適当なところで
Ctrl+Cで止めて、finishで抜ける

✓ シミュレーション結果の確認(ck,rst,LEDを選択)

```
$ gtkwave dump.v
```

✓ 出力LEDが、「チカチカ」しているはず