オープンソースEDAを活用したLSI開発環境の開拓

日置雅和、堀洋平、片下敏宏、更田裕司、秋田一平

先端半導体研究センター
集積回路設計研究チーム

自己紹介

- 2003年3月　東北大学大学院工学研究科電子工学専攻修了　舛岡富士雄先生（フラッシュメモリの発明者@東芝）
  SGT（GAA）型フラッシュメモリセルの設計と形状効果の研究、博士（工学）
- 2003年4月　産総研入所
  電力再構成可能FPGAであるFlex Power FPGAのアーキテクチャ設計、回路設計、チップ評価
- 2012-2013年　経産省 情報通信機器課（現情報産業課）出向
- 2013-2018年　産総研に戻る
  SOTB版Flex Power FPGA設計・評価、超伝導量子ビット設計環境整備・簡易PDK作成・設計
- 2018-2023年　AIチップデザインオープンイノベーションラボラトリ（AIDL）　チーム長
  デバイス技術研究部門先端集積回路研究グループ　グループ長
  AIチップ設計拠点（AIDC）の窓口・庶務担当
- 2023年～現在　先端半導体研究センター（SFRC）
  集積回路設計研究チーム　チーム長

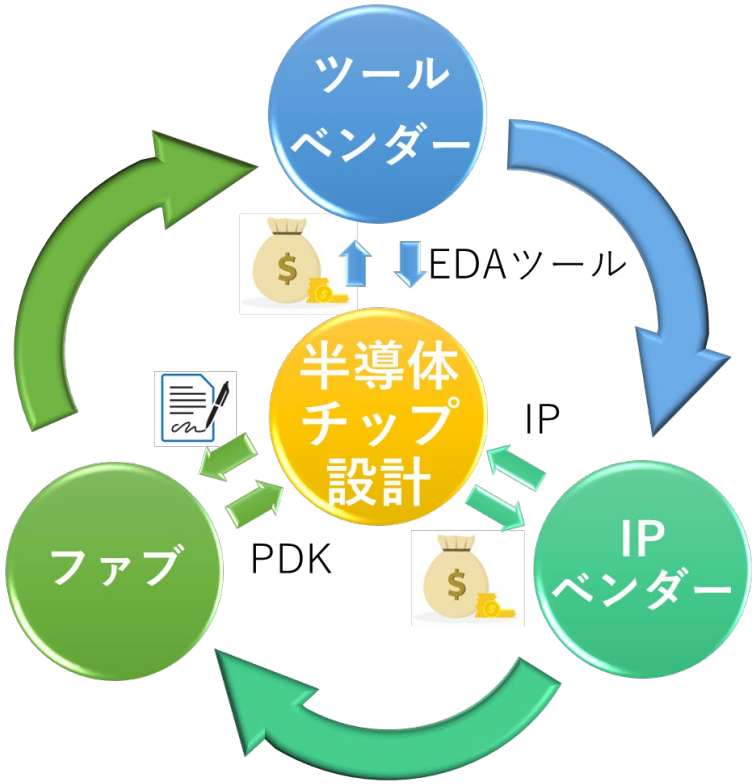
SGT: Surrounding Gate Transistor
GAA: Gate All Around
SOTB: Silicon On Thin Buried oxide

LSI開発のエコシステム

✓現在のLSI開発：既存エコシステムの束縛⇒**新規参入者にとって高い参入障壁**

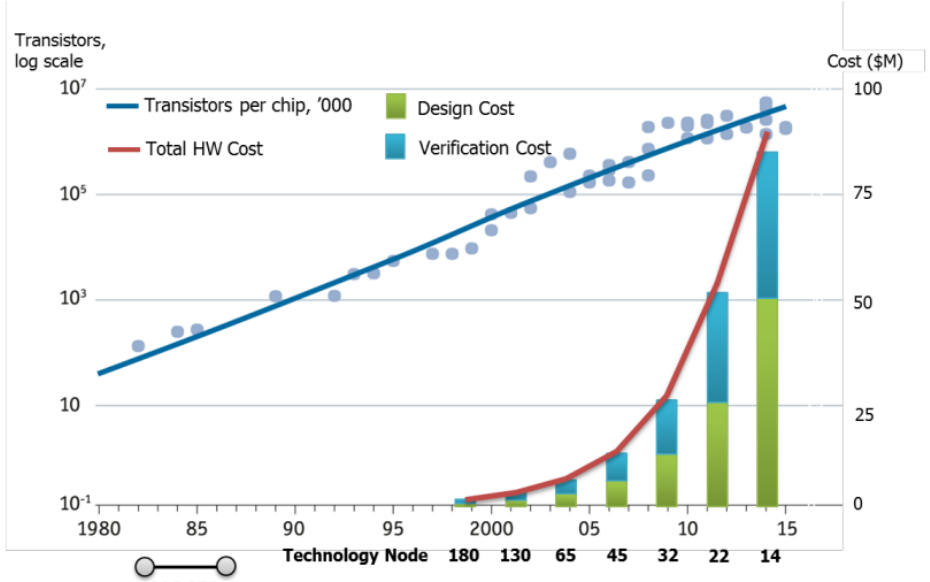
- **EDAツール**（設計ソフトウェア）
- 高価な**IP**（設計資産：インターフェース、メモリ等）
- 入手に手間のかかる**PDK**（Process Design Kit：デザイン情報）

⇒**ハードウェアイノベーションのバリアを下げるための施策@USA**



ERI: Electronics Resurgence Initiative, 2017

DARPA Lowering barriers to hardware innovation



New procedures for physical design and verification will lower the design barrier, enabling rapid specialization

Design

Intelligent Design of Electronic Assets (IDEA)

- No human in the loop" 24-hour layout generation for mixed signal integrated circuits, systems-in-package, and printed circuit boards.Machine generated layout of electrical circuits and systems

Verification

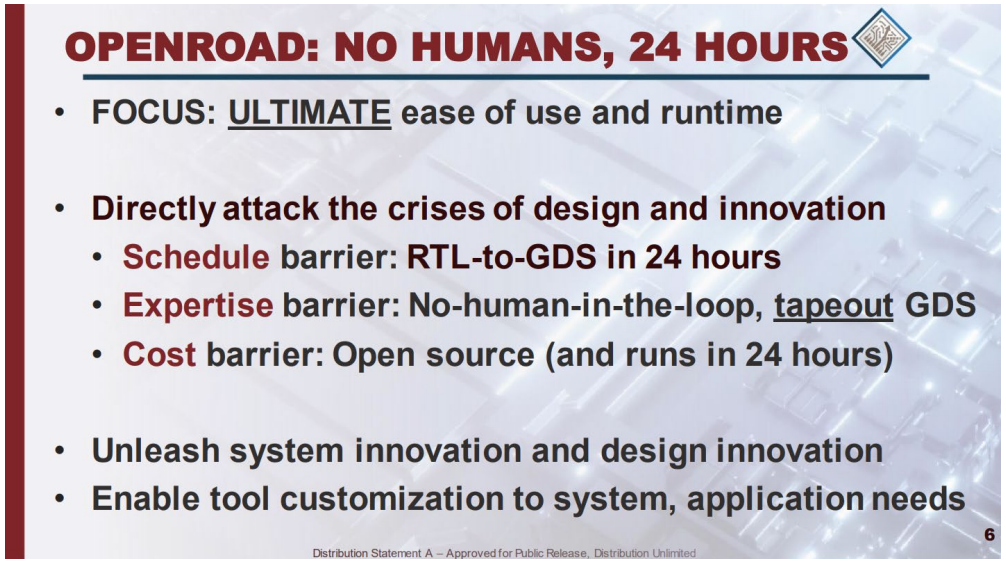
Posh Open Source Hardware (POSH)

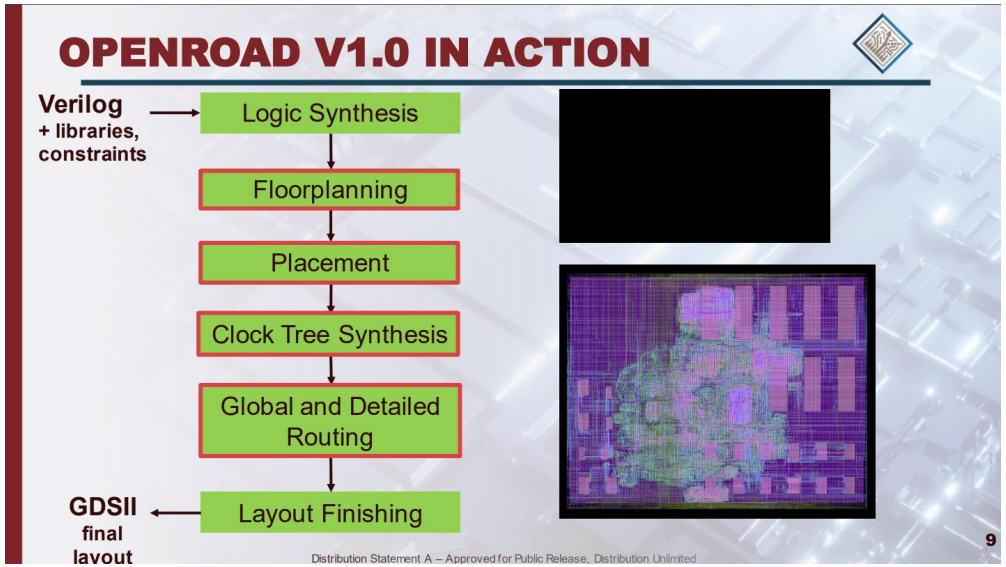
- An open source System on Chip (SoC) design and verification eco-system that enables cost effective design of ultra-complex SoCs.

The 1980's DARPA MOSIS effort removed fab cost and fab access barriers and launched the fabless industry. The ERI Design effort will address today's design complexity and cost barriers, creating the environment needed for the next wave of US semiconductor innovation.

https://www.darpa.mil/attachments/eri_design_proposers_day.pdf

IDEAプログラム：OpenROADプロジェクト（2018～2023）

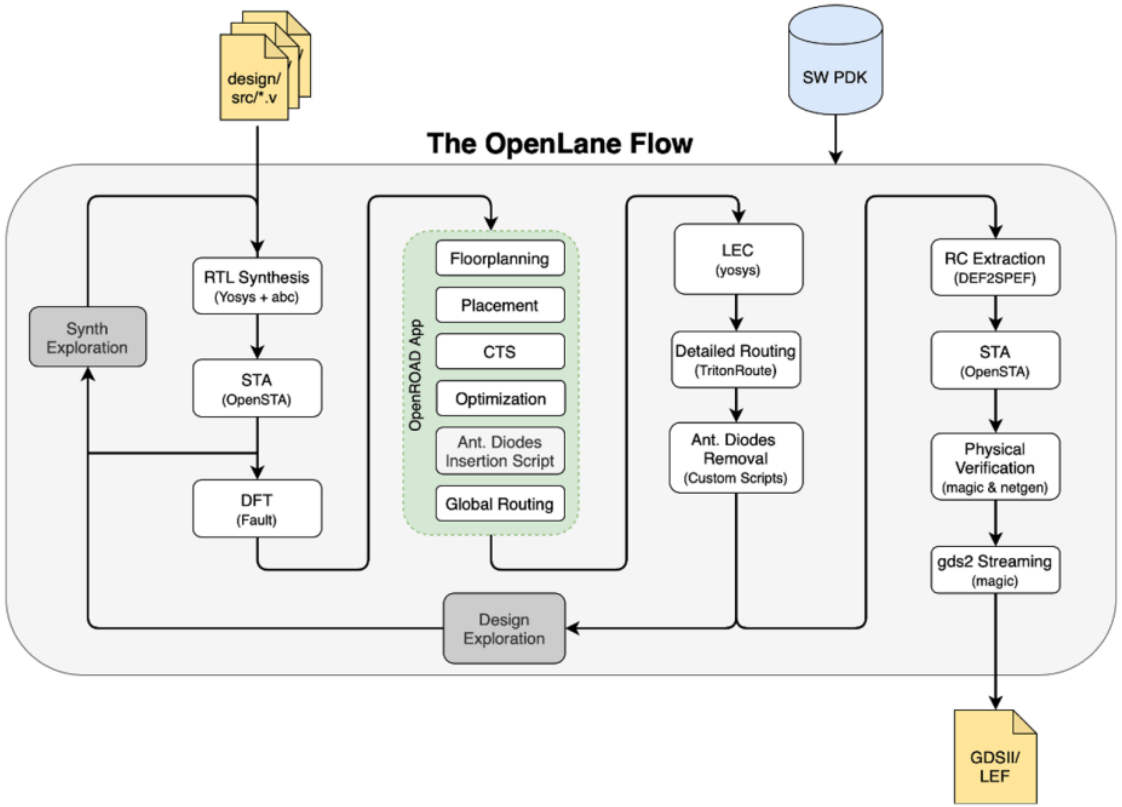




- PL: Prof. ANDREW B. KAHNG@UCSD
- DARPA ERI(Electronic Resurgence Initiative) Summit 2018
  - 24-hours, Non-Human-In-Loop layout design for SOC, Package and PCB with no Power-Performance-Area (PPA) loss, Tapeout-capable tools in source code form -> Open Source
- Funded by DARPA 2019-2023, Now supported by Precision Innovations
- Used in research and commercial apps.
  - OpenROAD-flow-scripts(OpenROAD)
  - OpenLane(Efabless)
  - Silicon Compiler (ZeroASIC)
  - Hammer (UCB)
  - OpenFASoC (IEDA-FASoC)
- PDK Support
  - Open Source: GF180nm, SKY130nm, FreePDK45nm, ASAP7nm
  - Proprietary: GF55nm, GF12nm, Intel22nm, Intel16nm, TSMC65nm

Ref. Kahng@ERI Summit 2020

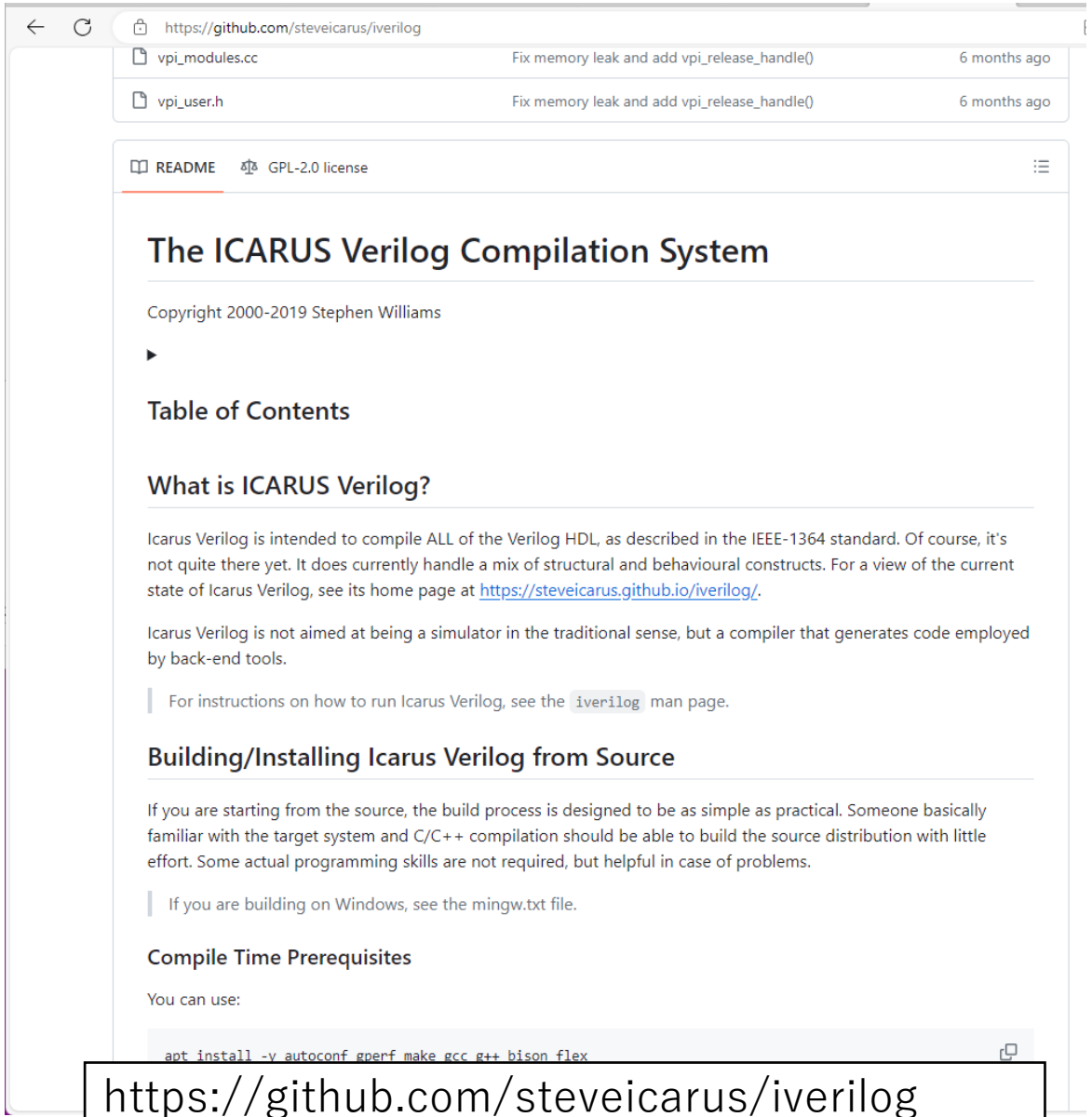
OpenLaneプロジェクト（2020～）



- 学会発表：A 130nm OpenROAD-based Tapeout-Proven Flow (ICCAD 2020)@efabless
- OpenLane ＝OpenROAD + Yosys, Magic, Netgen, CVC, Klayout and custom scripts
- PDK Support
  - Open Source: SKY130nm, GF180nm

https://openlane.readthedocs.io/en/latest/flow_overview.html

個々のツールの開発の歴史は古い
ICARUS Verilogシミュレータ（1998年～）



https://github.com/steveicarus/iverilog

GTKWave　波形表示（10年くらい前～）

産総研
ともに挑む。つぎを創る。

https://gtkwave.sourceforge.net

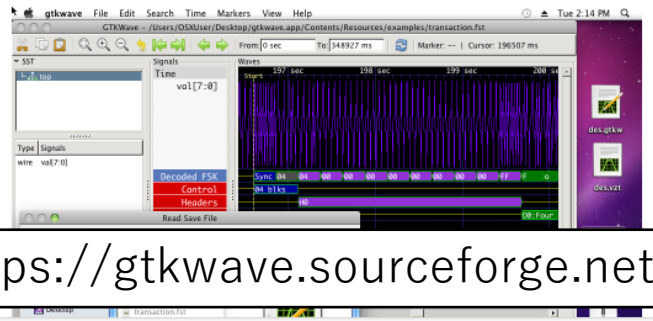
Welcome to GTKWave

GTKWave is a fully featured GTK+ based wave viewer for Unix, Win32, and Mac OSX which reads LXT, LXT2, VZT, FST, and GHW files as well as standard Verilog VCD/EVCD files and allows their viewing. You can grab version 3.3.117 here. Documentation in pdf format can be found here.

For svn access to the experimental, pre-release sourcetree on Sourceforge:
svn checkout svn://svn.code.sf.net/p/gtkwave/code/ gtkwave-code

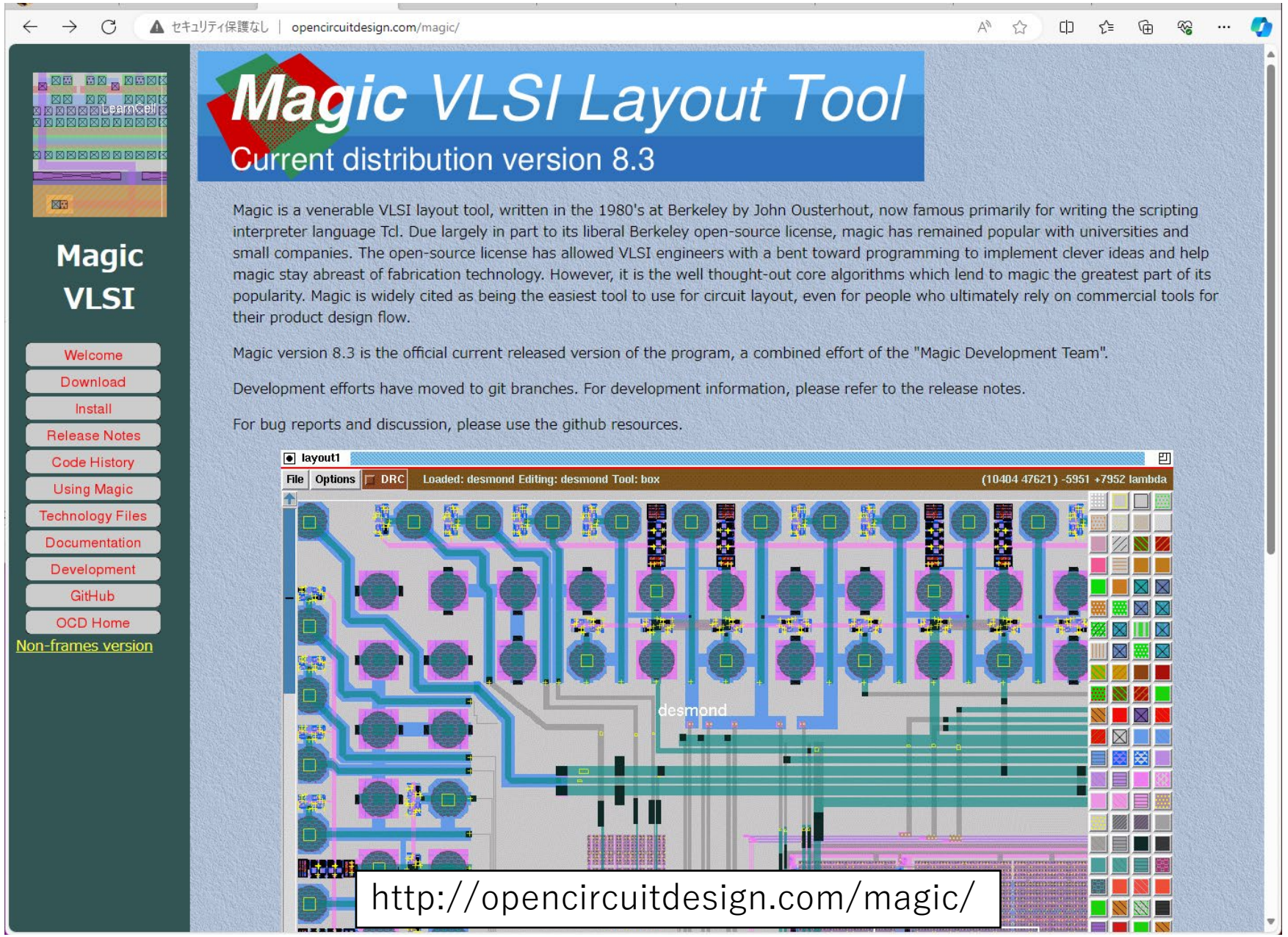
Native Win32 and OSX binaries are available here, however if you are a Windows user running Cygwin, running under that is recommended instead.
A Mac port can be found both here and here.
Ports to other platforms which GTK supports should be trivial.

An experienced professional shown violating most known rules of electrical safety with GTKWave. Please do not attempt this at home.

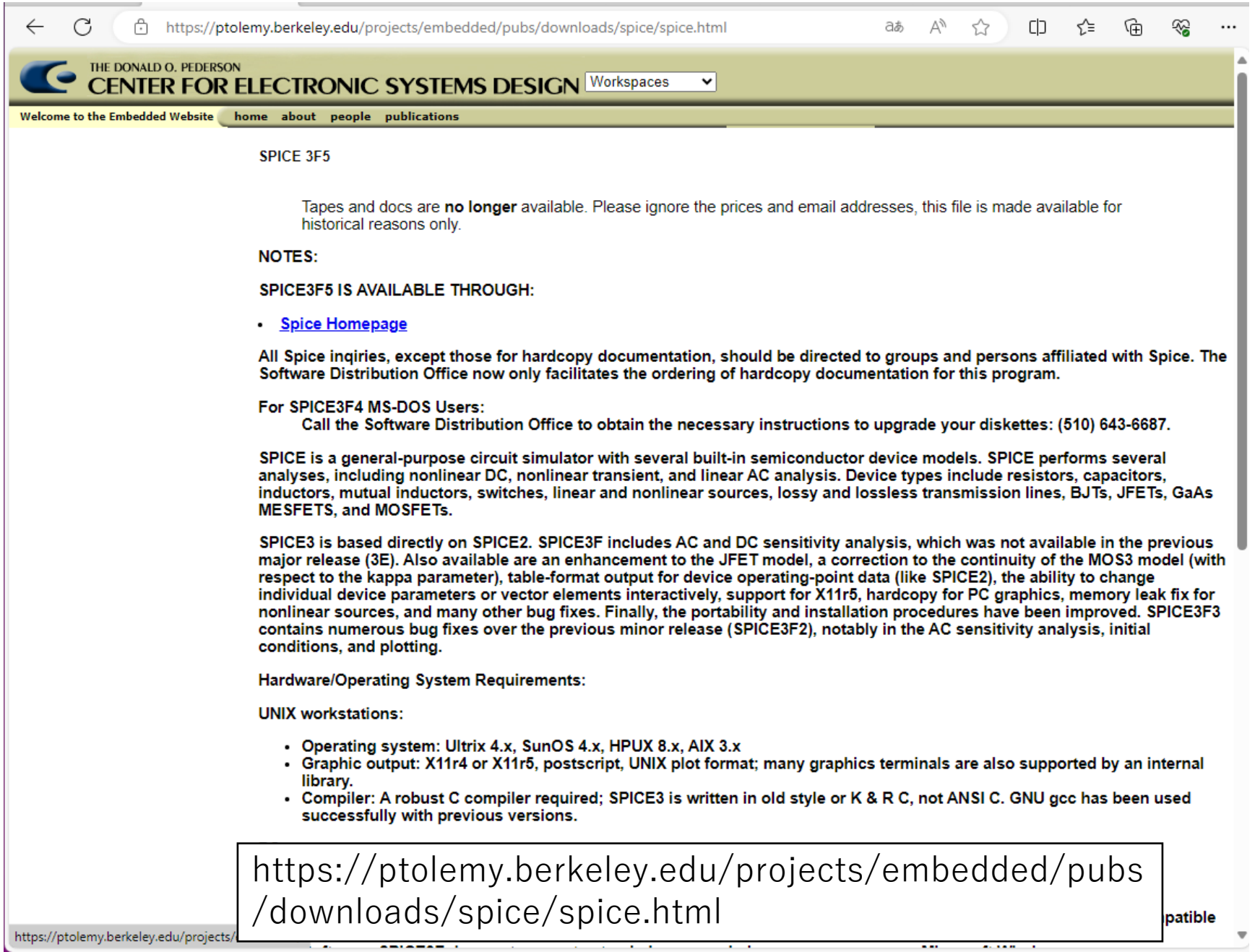


https://gtkwave.sourceforge.net/

Magic レイアウトエディタ（1980's～）



SPICE3　アナログ回路シミュレータ（1996～）



https://ptolemy.berkeley.edu/projects/embedded/pubs/downloads/spice/spice.html

THE DONALD O. PEDERSON
CENTER FOR ELECTRONIC SYSTEMS DESIGN　Workspaces

Welcome to the Embedded Website　home　about　people　publications

SPICE 3F5

Tapes and docs are **no longer** available. Please ignore the prices and email addresses, this file is made available for historical reasons only.

NOTES:

SPICE3F5 IS AVAILABLE THROUGH:

- Spice Homepage

All Spice inqiries, except those for hardcopy documentation, should be directed to groups and persons affiliated with Spice. The Software Distribution Office now only facilitates the ordering of hardcopy documentation for this program.

For SPICE3F4 MS-DOS Users:
Call the Software Distribution Office to obtain the necessary instructions to upgrade your diskettes: (510) 643-6687.

SPICE is a general-purpose circuit simulator with several built-in semiconductor device models. SPICE performs several analyses, including nonlinear DC, nonlinear transient, and linear AC analysis. Device types include resistors, capacitors, inductors, mutual inductors, switches, linear and nonlinear sources, lossy and lossless transmission lines, BJTs, JFETs, GaAs MESFETS, and MOSFETs.

SPICE3 is based directly on SPICE2. SPICE3F includes AC and DC sensitivity analysis, which was not available in the previous major release (3E). Also available are an enhancement to the JFET model, a correction to the continuity of the MOS3 model (with respect to the kappa parameter), table-format output for device operating-point data (like SPICE2), the ability to change individual device parameters or vector elements interactively, support for X11r5, hardcopy for PC graphics, memory leak fix for nonlinear sources, and many other bug fixes. Finally, the portability and installation procedures have been improved. SPICE3F3 contains numerous bug fixes over the previous minor release (SPICE3F2), notably in the AC sensitivity analysis, initial conditions, and plotting.

Hardware/Operating System Requirements:

UNIX workstations:

- **Operating system: Ultrix 4.x, SunOS 4.x, HPUX 8.x, AIX 3.x**
- **Graphic output: X11r4 or X11r5, postscript, UNIX plot format; many graphics terminals are also supported by an internal library.**
- **Compiler: A robust C compiler required; SPICE3 is written in old style or K & R C, not ANSI C. GNU gcc has been used successfully with previous versions.**

patible

https://ptolemy.berkeley.edu/projects/

https://ptolemy.berkeley.edu/projects/embedded/pubs/downloads/spice/spice.html

Xschem　回路図エディタ＋ネットリスティング（1998～）

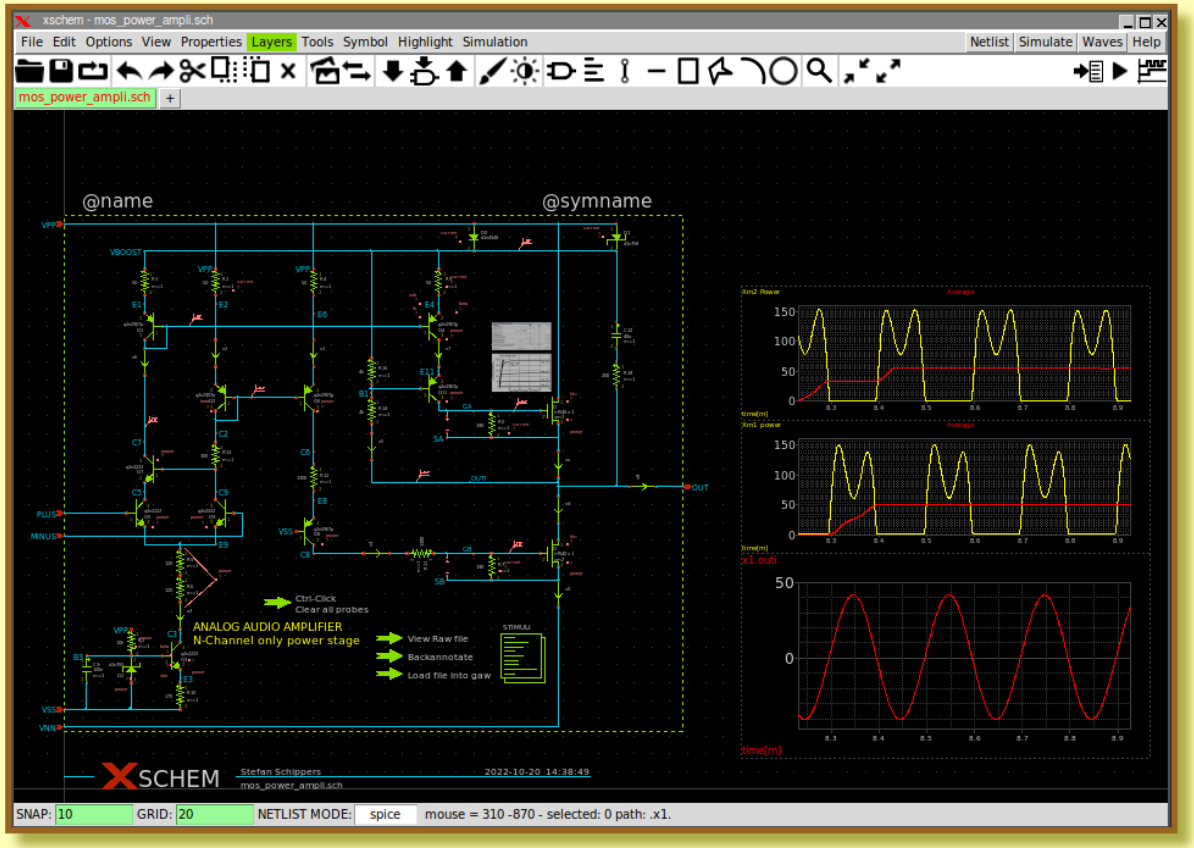
https://xschem.sourceforge.io/stefan/index.html

HOME　XSCHEM MANUAL　GOOGLE

- Solaris sparc
- Windows (with the cygwin layer and cygwin/Xorg X11 server, plus the tcl/tk toolkit and the -dev libraries)

Screenshots

- analog circuit example

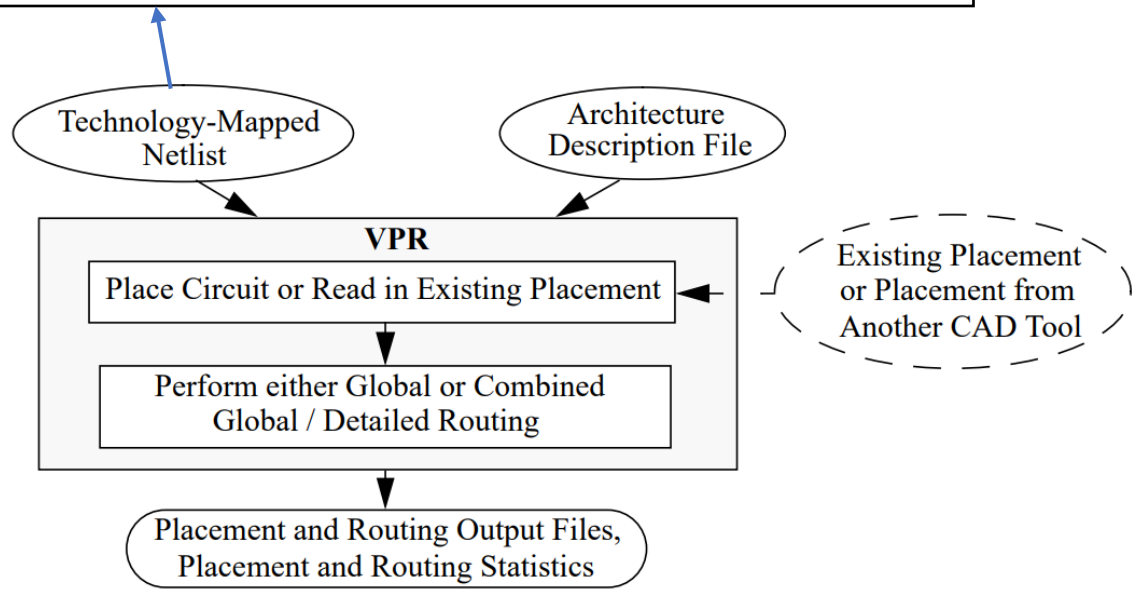


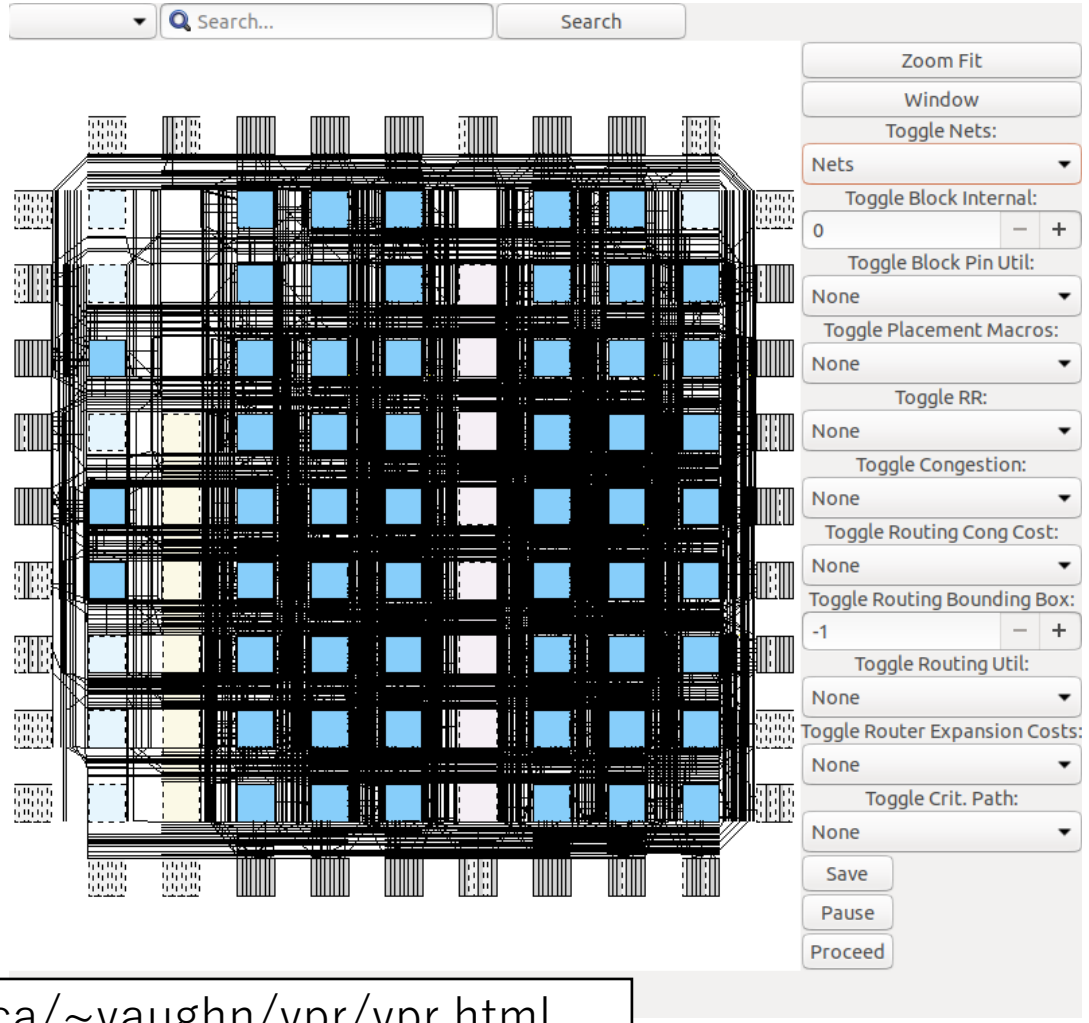
- digital system for VH

https://xschem.sourceforge.io/stefan/index.html

FPGAアーキテクチャ探索ツール
VPR:Versatile Place and Route @ Toronto University（1997～2011？）

BLIF: Berkeley Logic Interchange Format





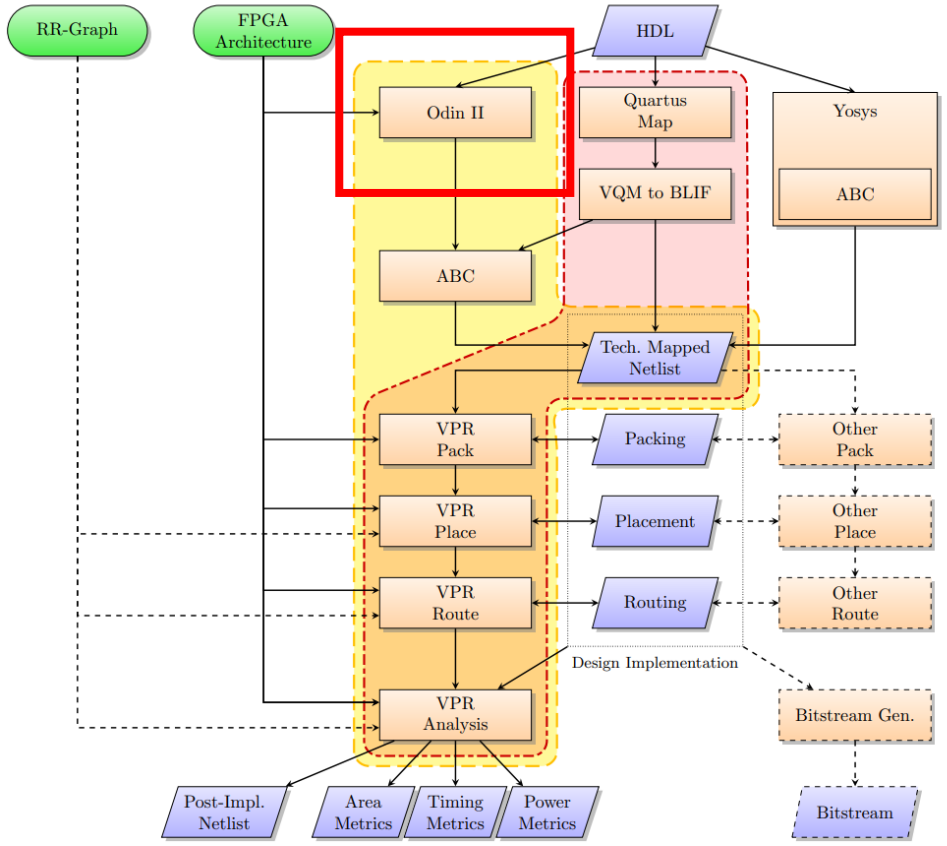
- オープンソースEDAツールでアカデミアにおけるFPGA研究を加速
- 入力がBLIFで使いにくい

https://www.eecg.utoronto.ca/~vaughn/vpr/vpr.html

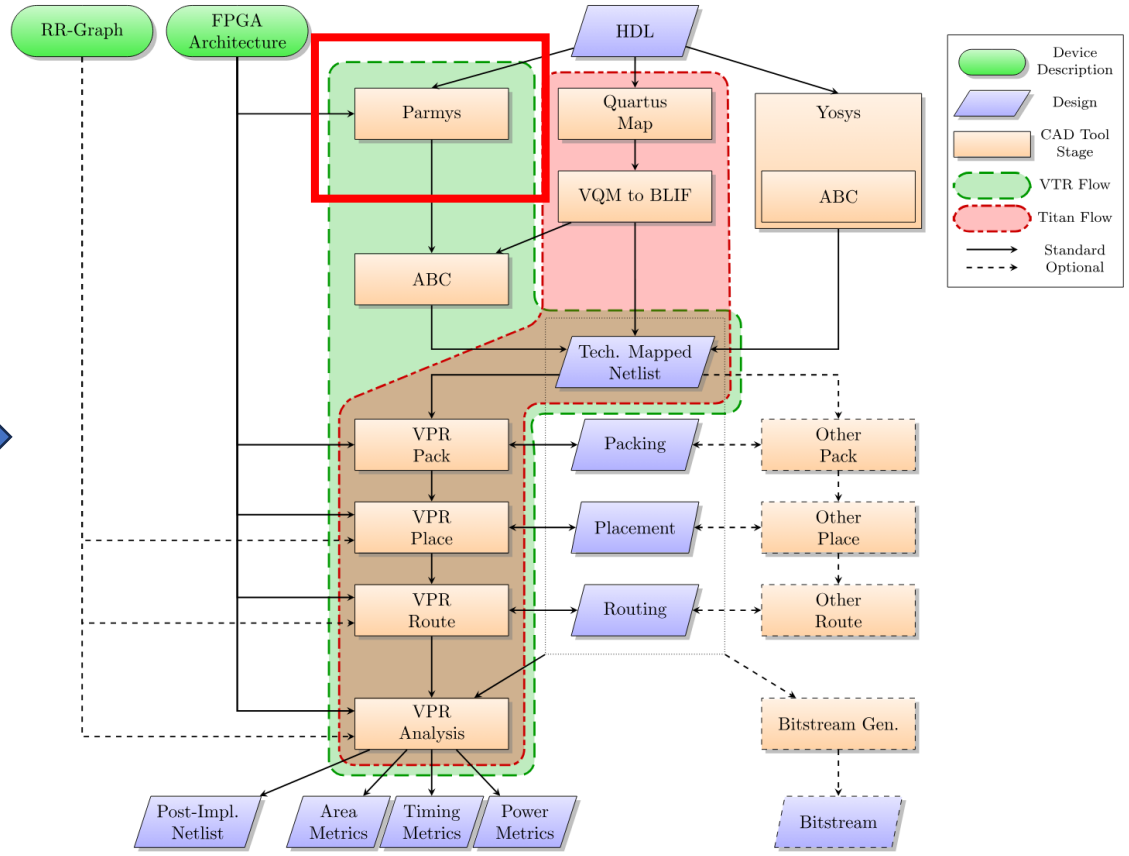
VTR: Verilog to Routing @ Toronto University（2012～）

Input: BLIF -> Verilog

Older version



Latest version (VTR8)

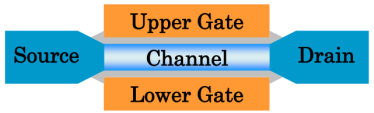


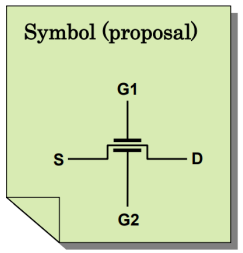
https://docs.verilogtorouting.org/en/latest/

オープンソースEDAを用いた研究の歴史１
XMOS(double gate MOS)モデルをspice3f5に組み込み@2004

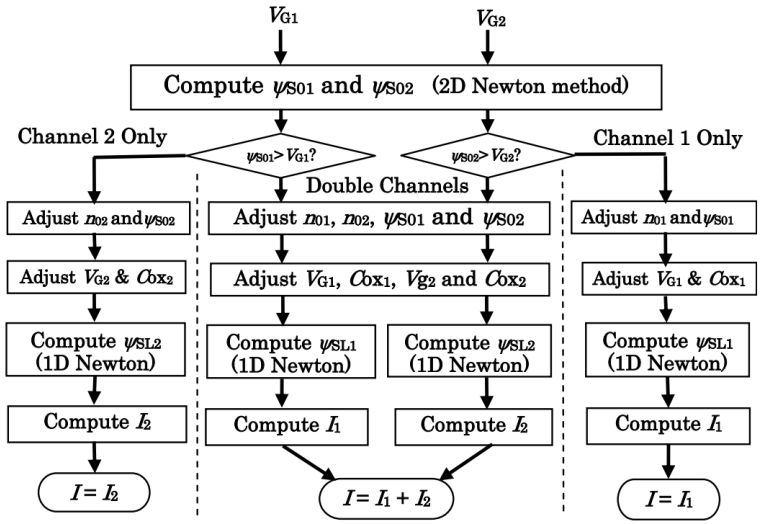
DG MOSFET

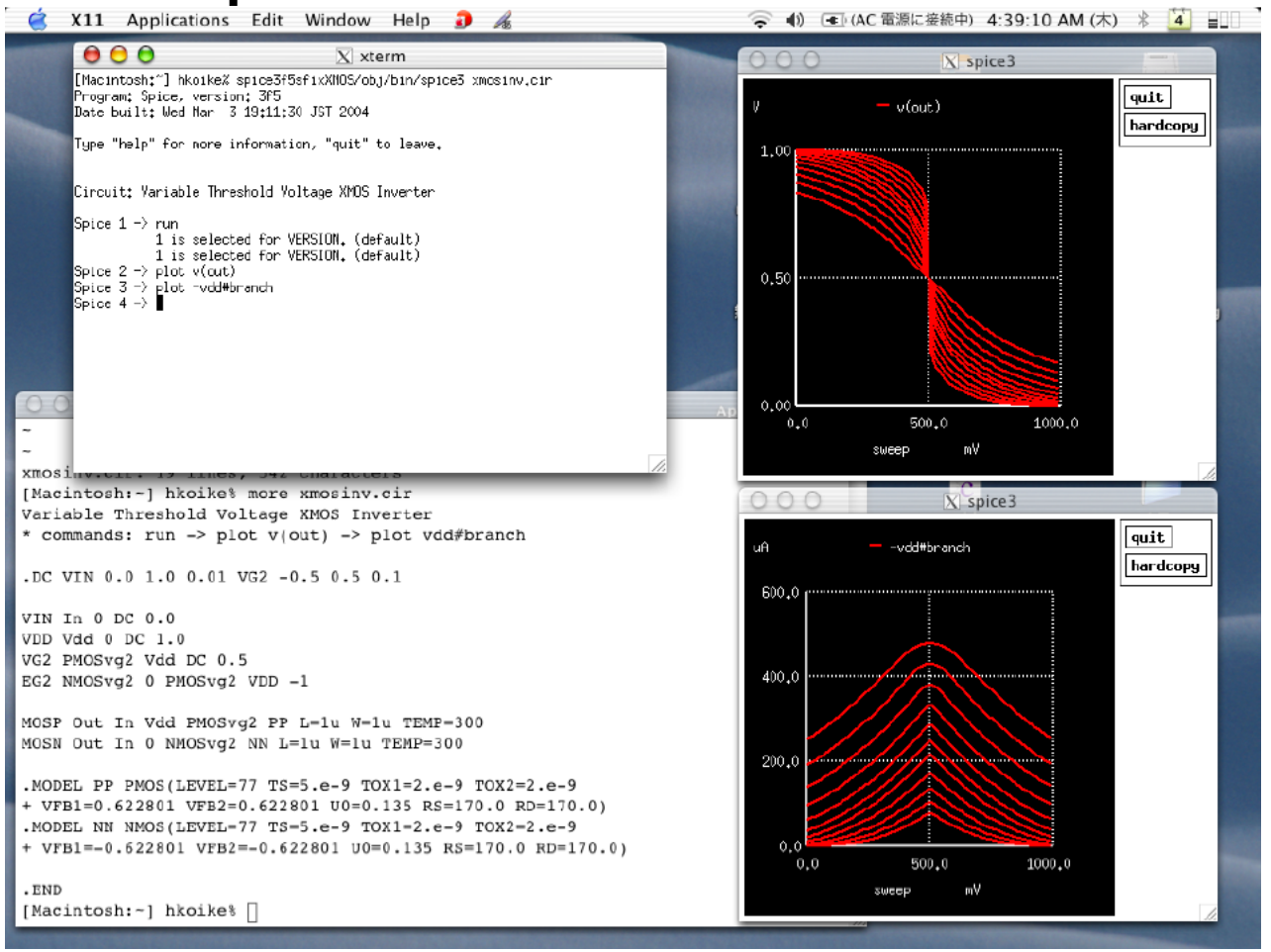
Two gates sandwich the Si channel.
The additional gate can be used as
-the gate electrically tied with the other, or
-another signal input, or
-an alternative for the body bias electrode.





In all usage, minimum short-channel effect is distinct.
When used as electrically tied gates (as FinFETs),
high current drivability and ideal S factor are prominent.
When used as an alternative for the body bias electrode,
high frequency body bias, aggressive forward body bias
and other sophisticated body bias techniques can be easily achieved.
Predominant disadvantage is difficulty of manufacturing,
now overcome in research level, but
long way to go in LSI or production level.

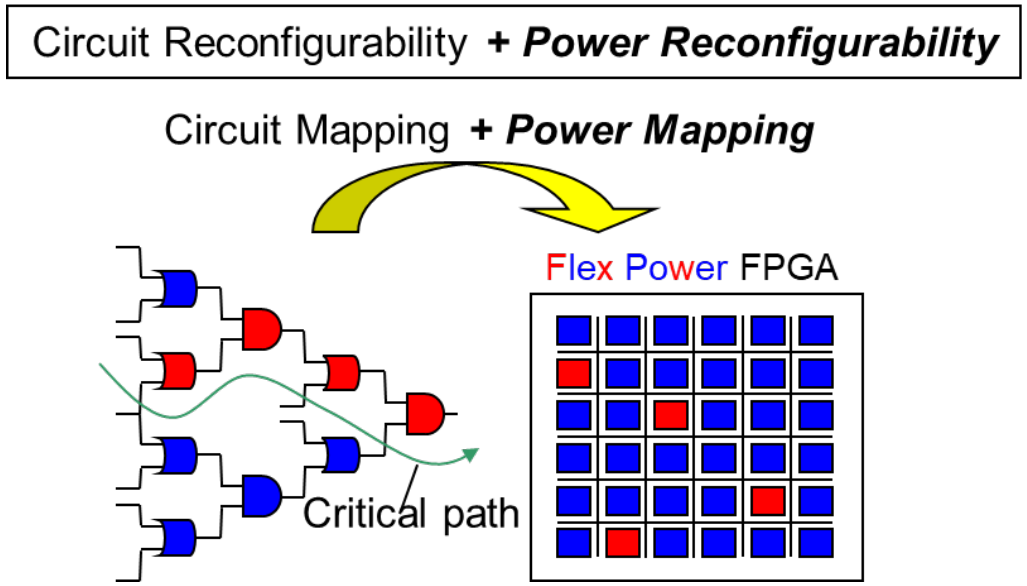


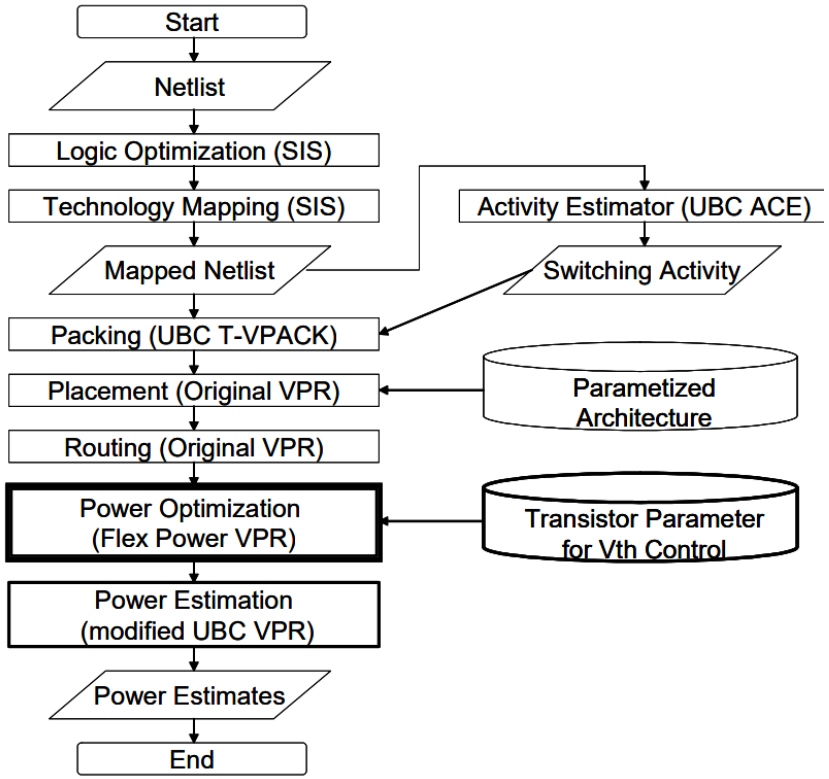


Ref. 中川他@AIST、nanotech2004

オープンソースEDAを用いた研究の歴史２
Flex Power FPGA用のCADフローを開発
（Ref. 河並他@AIST、IEICE2004）

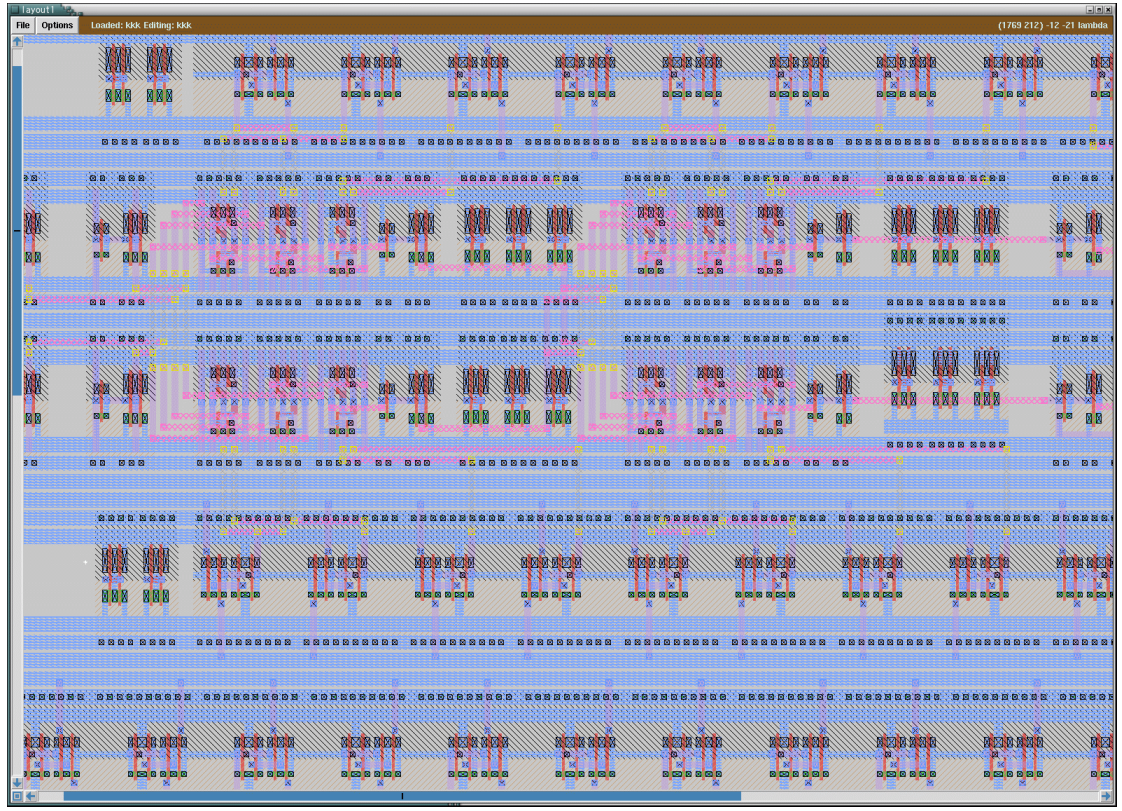
- 電力を再構成可能なFPGA：Flex Power FPGAの研究開発
- TorontoのVPRにVT Mapperを追加
- タイミングスラックをパワーに変換



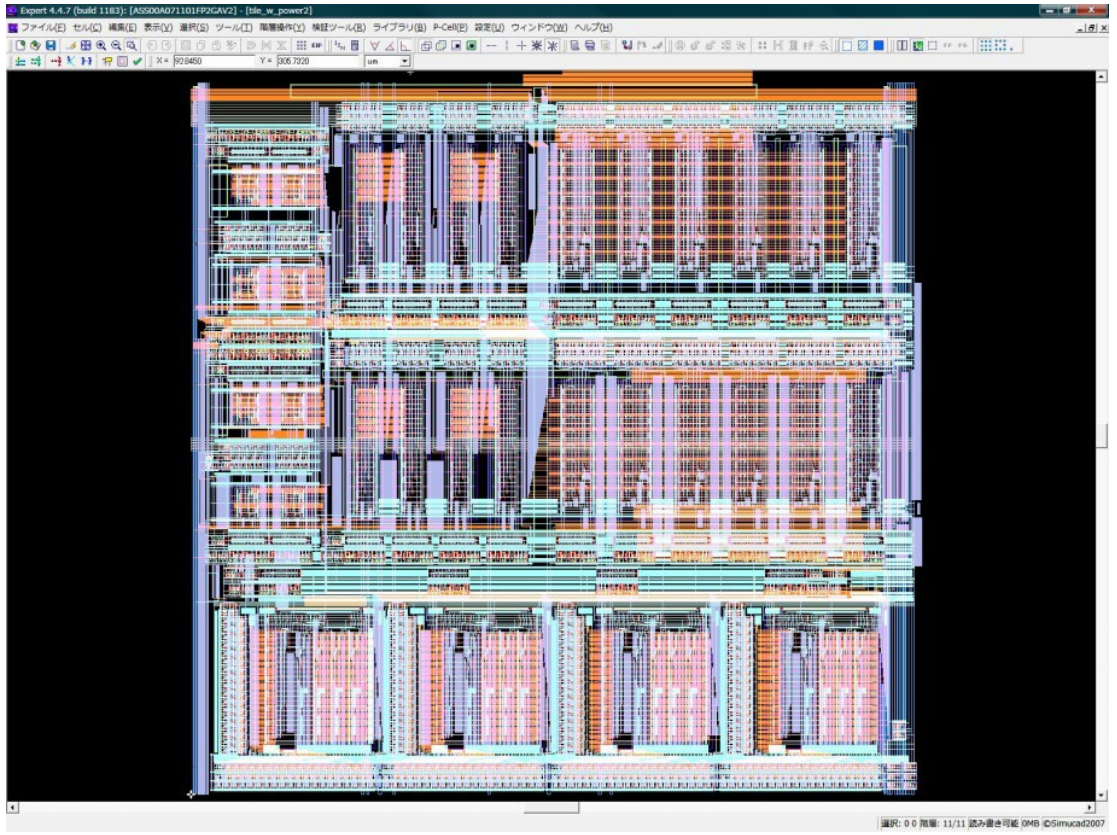


オープンソースEDAを用いた研究の歴史３
オープンソースレイアウトエディタMagicを使って
Flex Power FPGAレイアウトを予備評価@2007

Magicを使った面積評価



商用レイアウトツールで設計



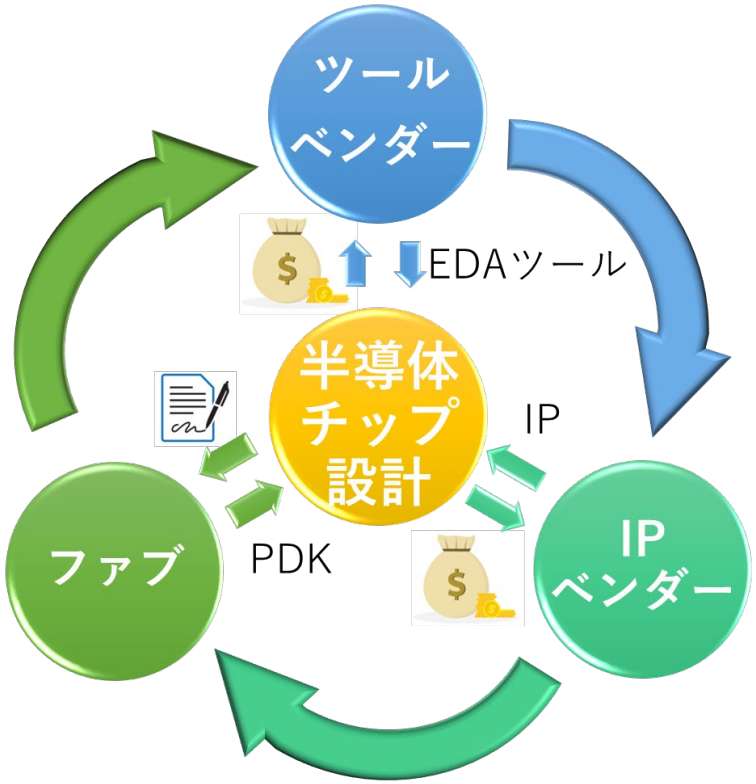
商用ツールへ移行
⇒ツール向けPDKがあったから（TSMC90nm）

（再掲）LSI開発のエコシステム

✓現在のLSI開発：既存エコシステムの束縛⇒**新規参入者にとって高い参入障壁**

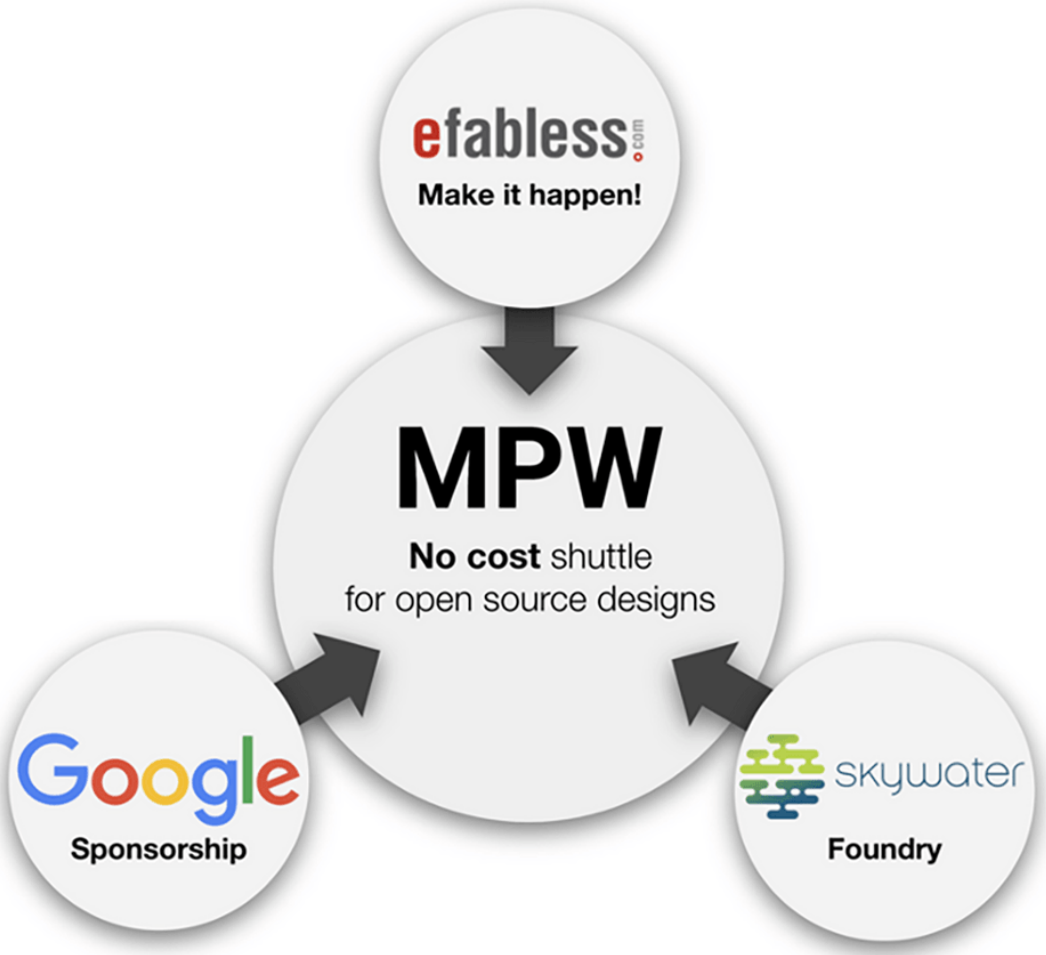
- 高価な**EDAツール**（設計ソフトウェア）
- 高価な**IP**（設計資産：インターフェース、メモリ等）
- 入手に手間のかかる**PDK**（Process Design Kit：デザイン情報）

⇒**EDAツールへのテコ入れだけでいいの？**



Open MPW Shuttle Program （2020～）

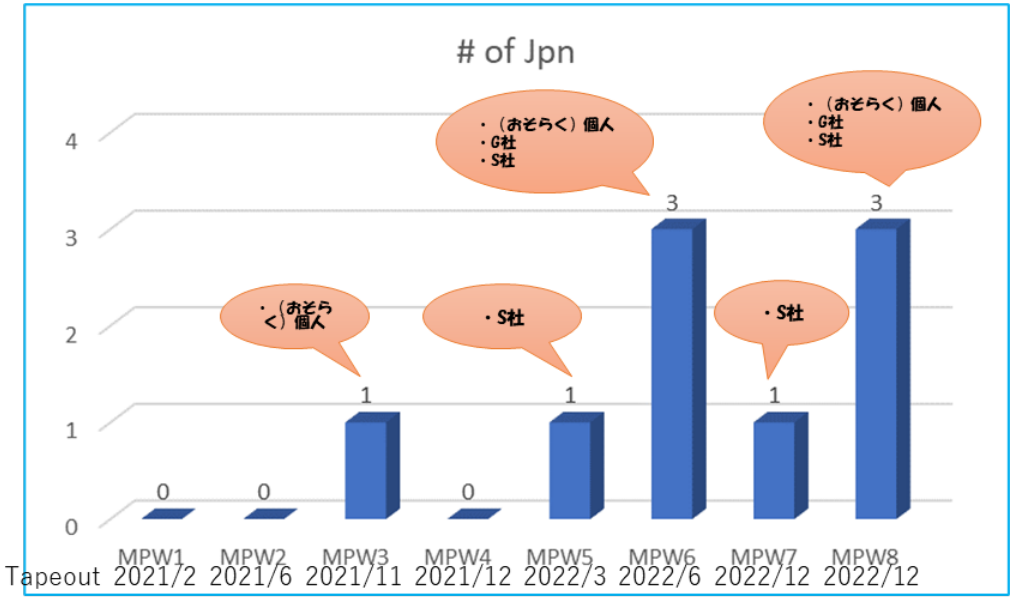
- Google-efabless-SkyWater Open MPW(:Multi Project Wafer) Shuttle Program from 2020
- オープンソースPDK
- 無償のシャトルプログラム
- 設計IPは公開
- 各機関の役割
  - Google: スポンサー、オープンソースPDK管理、高位合成ツールチェーン（XLS）開発など
  - SkyWater: 130nmプロセスファンドリサービス、PDK
  - efabless: Open Source EDA 環境(OpenLane)
- Globalfoundriesが2022年に参画



https://efabless.com/open_shuttle_program

国内の利用状況

- ✓2020年よりGoogle社, skywater社, efabless社がオープンソースEDA環境・PDK/IPを開発し、無償シャトルサービス（情報公開が原則）を開始。全世界で**600件以上**の利用実績。
- ✓2022年よりGlobalFoundriesが参画
- ✓日本国内からの利用例は少なかった（**10件未満**）
- ✓**設計に係る情報や知見が足りない？**



オープンソースEDA環境・試作サービスの国内の利用件数

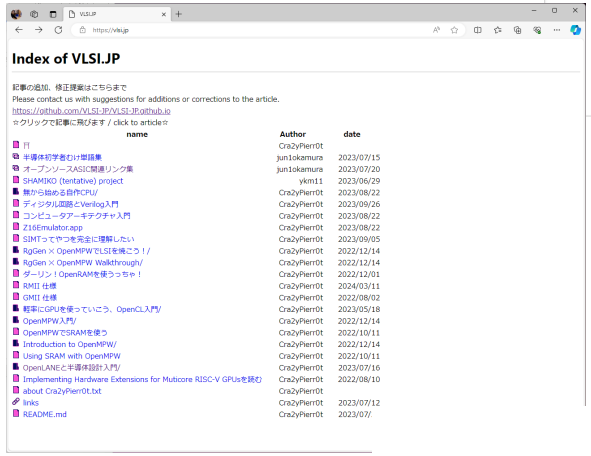
2024/3EのGFMPW-1の日本エントリ：10件⇒TOフェーズ：6件！！

GFMPW-1 Open MPW Shuttle Service

prova	N/A	Gian Domenico Licciardo	Italy	No	N/A	N/A	0
rvcore_chip2	Digital	Kenji Kise	Japan	No	Pass	Pass	1
test	Digital	Miho Yamada	Japan	No	Fail	N/A	2
omotya	Digital	Riku Anan	Japan	Yes	Pass	Pass	3
xls-group-tapeout	Digital	Johan Euphrosine	Japan	No	Fail	N/A	1
micro_irritating_maze	Digital	Noritsuna Imamura	Japan	Yes	Pass	Pass	29
ISHI-KAI_Multiple_Projects_OpenGFMPW-1	Digital	KAI ISHI	Japan	No	Pass	Pass	9
my_gf180	Digital	Riku Anan	Japan	Yes	Pass	Pass	3
JSpi	Digital	Herbert T	Japan	No	Pass	N/A	5
ishikai-gds-test-homelith	Digital	Toru Homemoto	Japan	No	Pass	N/A	4
gitefu_check	Digital	Gitefu	Japan	No	Pass	Pass	1
Ophelia_eFPGA_rerun	Digital	Egor Lukyanchenko	Kazakhstan	Yes	Fail	Pass	2

国内コミュニティ

- ISHI会(website,discord)
- Open Source EDA supporters(discord)
- VLSI.jp(website)
- 個人ブログの記事
- AIST Solutions　AI・半導体プロデュース事業部(slack)



https://ishi-kai.org/
https://www.anagix.com/nyusu/オープンソースeda-supportersdiscordへの招待
https://vlsi.jp/
https://www.aist-solutions.co.jp/

AIST所内プロジェクトの立ち上げ（2023/6～）

産総研
ともに挑む。つぎを創る。

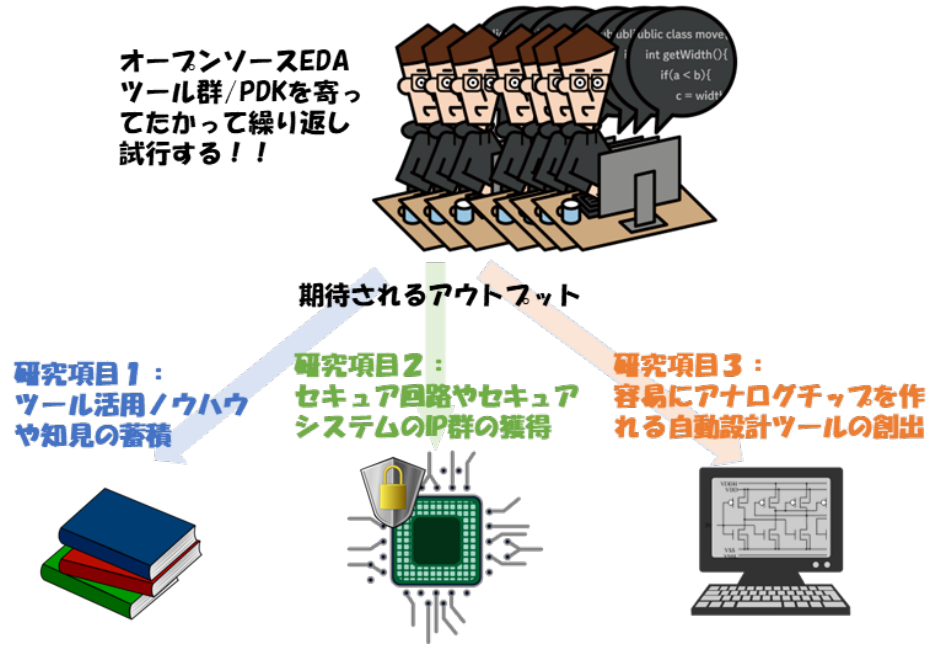
目的：国内のオープンソースEDA/PDKの利用普及に向けた開拓を産総研 がも 担う

研究項目１．利用環境構築と利用方法の知見の蓄積
- オープンソースEDAツールであるOpenLaneの利用環境構築と利用技術の確立
- efabless社の商用シャトル等を用いたLSIチップ試作・機能検証・評価

研究項目２．HWセキュリティ向け回路IPの開発
- 開発実績のある暗号・符号回路IPと新規開発の簡易RISC-Vコア・PUF回路を組み合わせたチップの作製・評価
- オープンソースEDA を用いたオリジナルセキュリティSoCの開発

研究項目３．オープンEDA/PDKを活用したアナログ設計環境の拡張
- 知識のない人でも簡単にアナログ回路を作成できる自動設計ツール開発

✓ 所内LSI研究者の英知を結集
✓ 産総研発回路IP※をフル活用



※代表的な産総研の回路IP
- PUF回路（NEDOプロ@2017-2021）
- 暗号回路（科研費等）
- モーションセンサー（Sensors 2021, Q1）
- 量子ビット読み出し回路（VLSI2022、プレス発表）
- 磁気センサー回路（ISSCC2022、プレス発表）

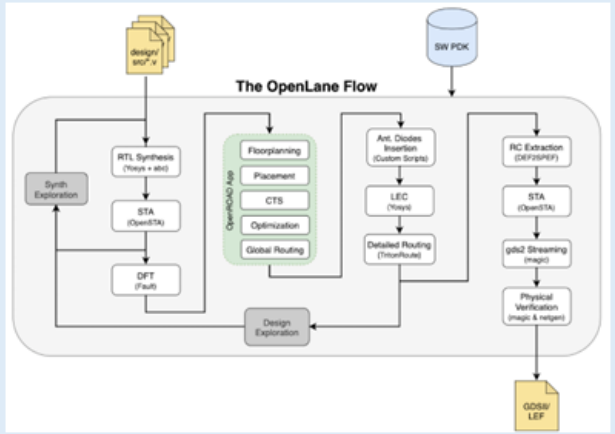
2023年度の進捗：利用環境構築と利用方法の知見の蓄積

産総研
ともに挑む。つぎを創る。

Summary

- オープンソースEDAであるOpenLane Flowの環境とシャトルテープアウト用のCaravel統合環境を試行。両環境を構築とフローの実行を完了
- 上記で得られた知見をドキュメント化

OpenLane flow



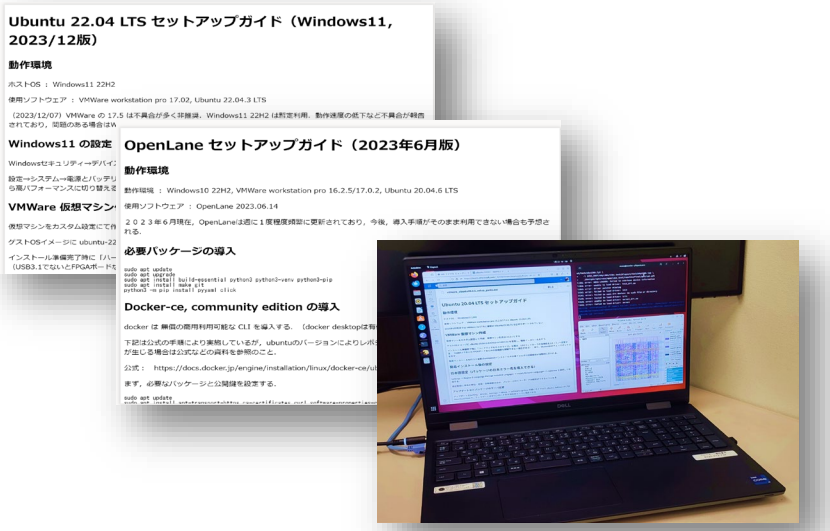
概要

- 複数のプログラムでできたツールチェーン
- 回路記述（RTL）からレイアウト（GDS）までを一気通貫で生成するフロー

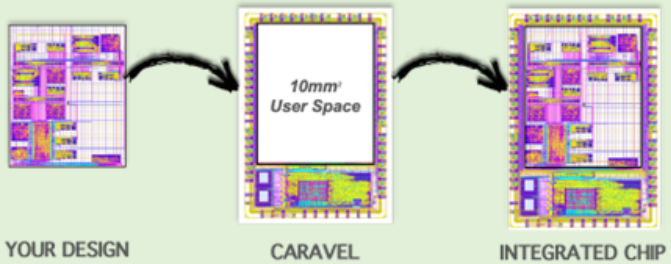
本家マニュアルに未記載の事項

- フローのアップデート方法
- 階層設計パラメータの設定
- 書式チェックの制限

など



Caravel統合環境



概要

- SkywaterとGlobalfoundriesのシャトルにテープアウトできるプログラムとチップフレームマクロのセット

本家マニュアルに未記載の事項

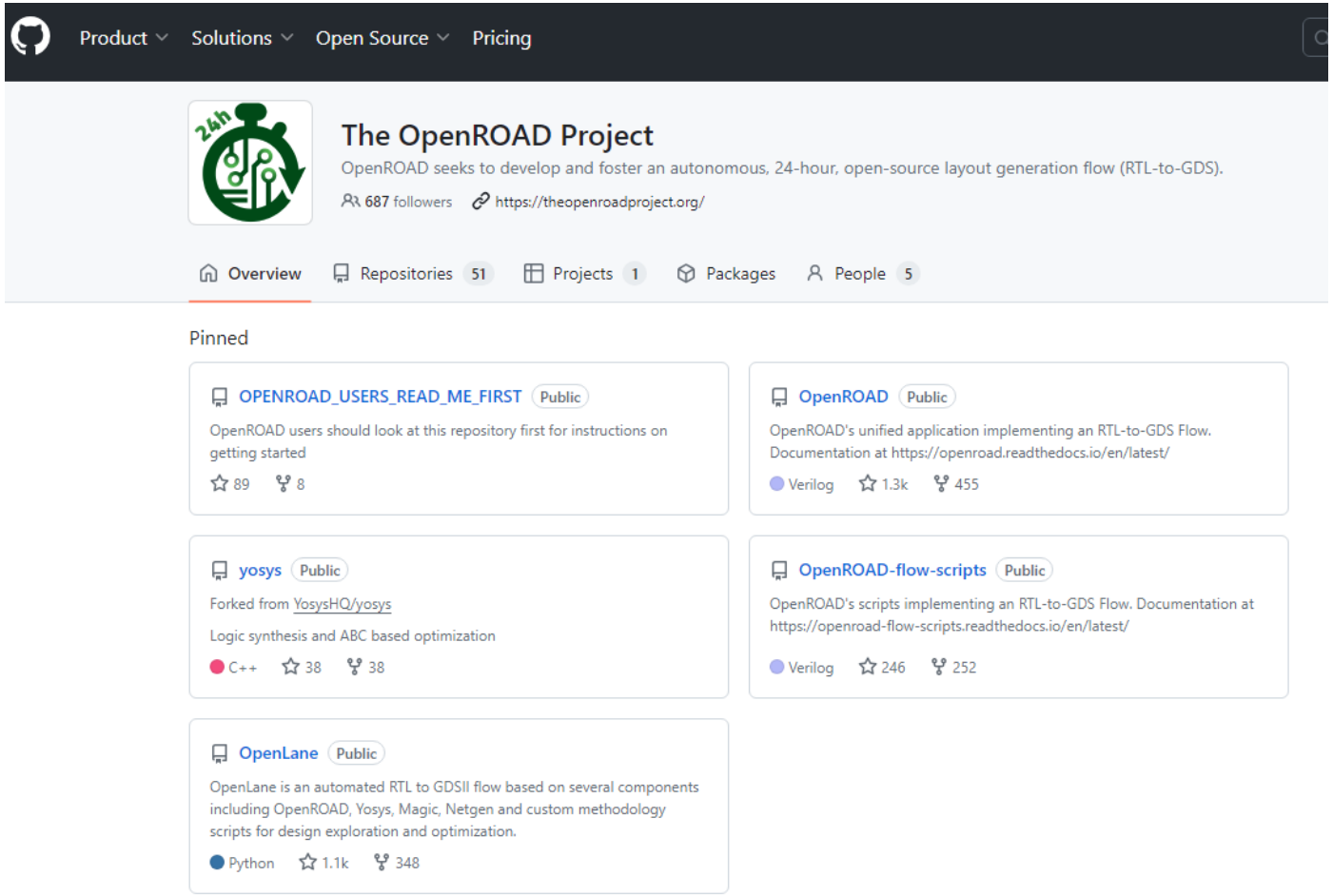
- READMEファイルの変更
- チップIO初期値の再設定
- 周辺回路IPやメモリIPの使い方

など

<u>成果物</u>

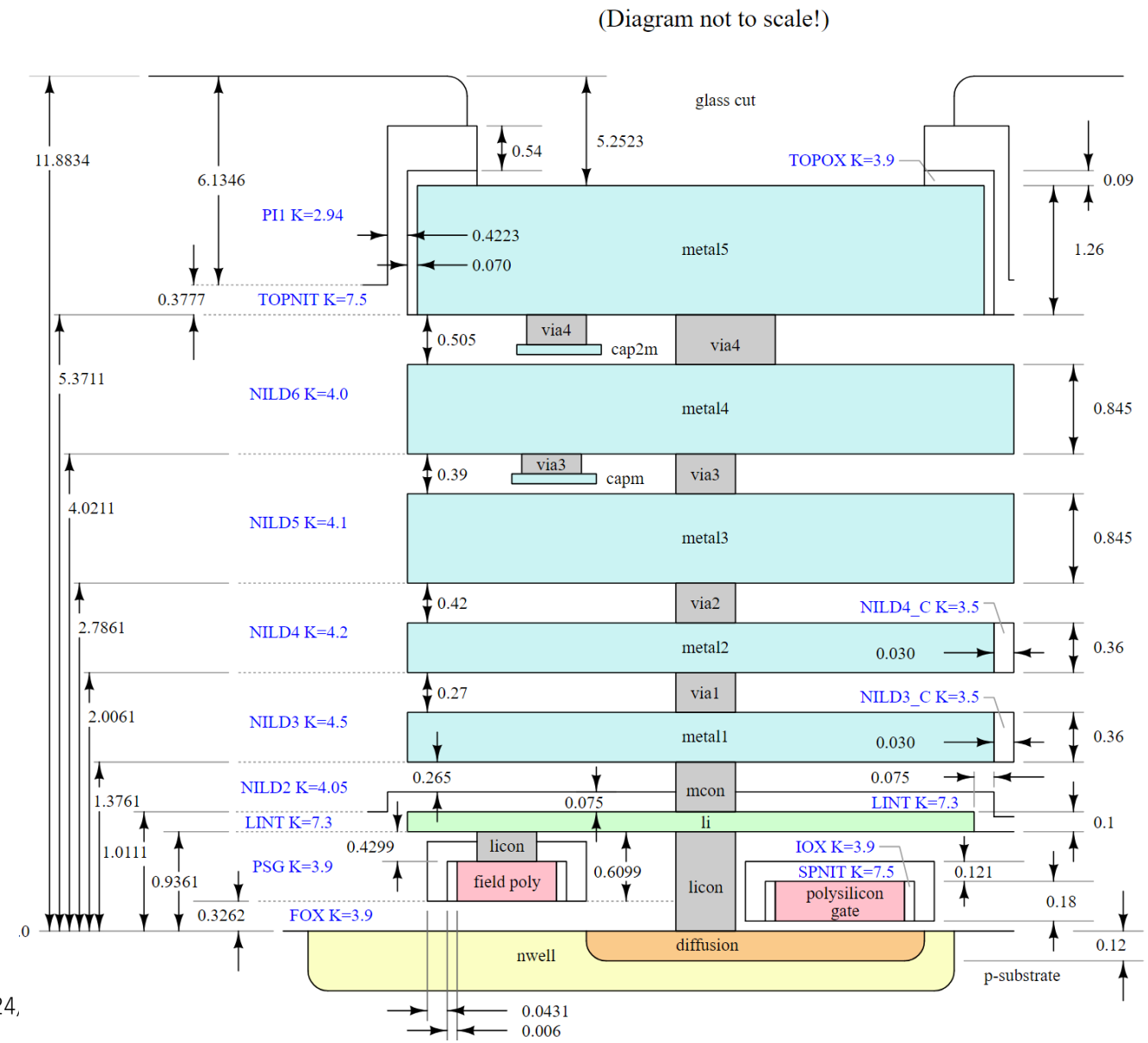
- OpenLane環境とCaravel環境の動作確認済み仮想イメージ
- 利用ドキュメント
  - Linux仮想環境構築
  - OpenLane導入と利用
  - Caravel導入と利用
  - Caravelと自作回路の結合とシミュレーション実行

OpenLaneの導入



- OS
  - Ubuntu 20.04+
  - macOS 11+
  - Windows10, 11+
  - Other Linux
- Download and Installation
  - Get python3.6
  - Get docker
  - git clone openlane
  - cd Openlane
  - Make （PDKも入る）
  - make test（インストールテスト）
  - make mount　（OpenLane環境の立ち上げ）
- バーチャルマシン上でも動く（ex. Ubuntu on Windows）：可搬性

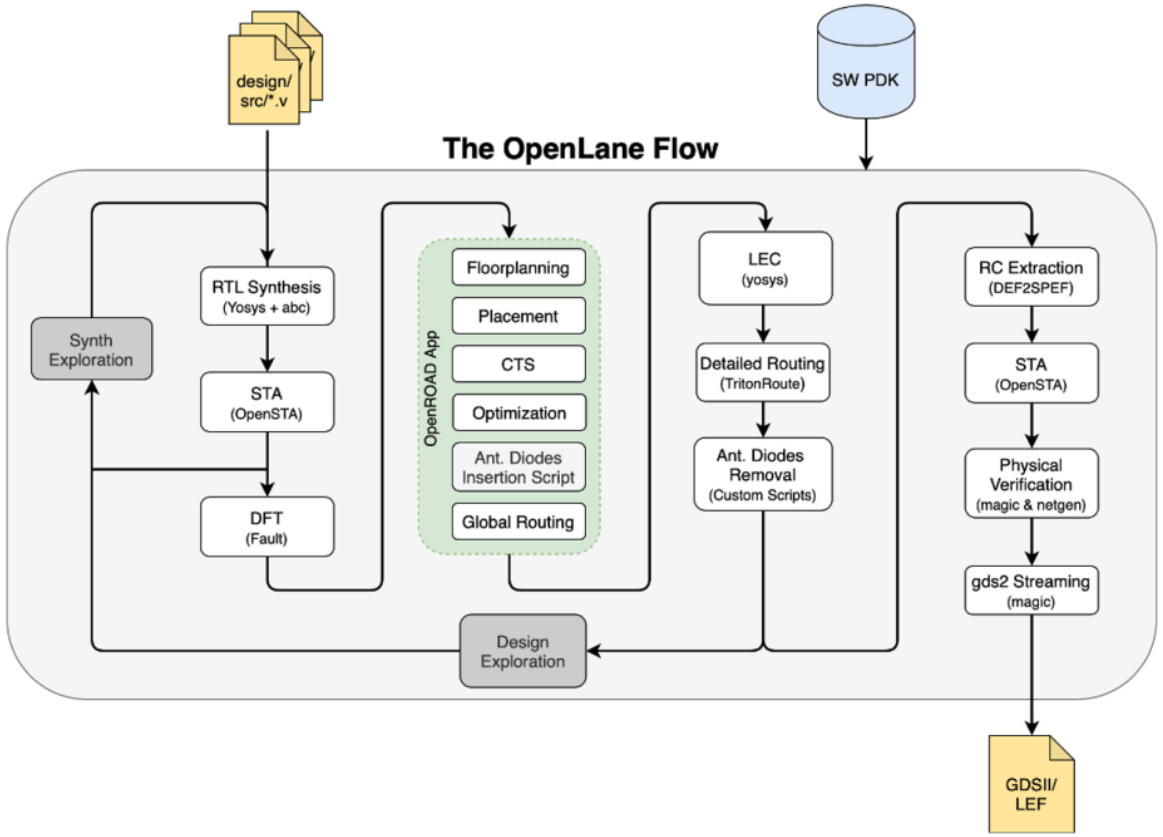
SkyWater 130nmプロセス



- 5 layer Al
- 1 local interconnect
- Poly gate
- Internal: 1.8V, I/O: 5V
- Inductor capable
- Poly resistor
- MIM Cap.

[GitHub - google/skywater-pdk: Open source process design kit for usage with SkyWater Technology Foundry's 130nm node.]

OpenLane Design Stage



LEC: Logic Equivalence Check
SPEF: Standard Parasitic Exchange Format

1. (Lint)
   - Verilator
2. Synthesis
   - Yosys/abc
   - OpenSTA
3. Floorplanning
   - Init_fp
   - ioplacer
   - Pdngen
   - tapcell
4. Placement
   - RePLace
   - Resizer
   - OpenDP
5. CTS
   - TritonCTS
6. Routing
   - FastRoute
   - TritonRoute
   - OpenRCX
7. Tapeout
   - Magic
   - KLayout
8. Signoff
   - Magic
   - Klayout
   - Netgen
   - CVC

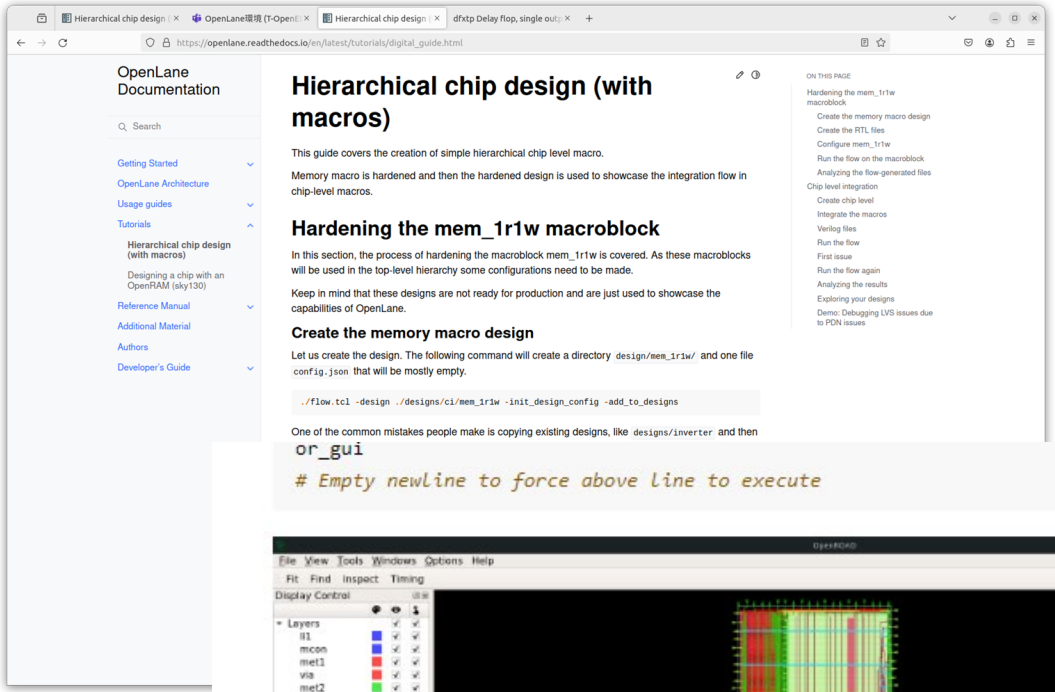
実際に実行すると48ステップある（Lint含む）

masa@oseda: ~/OpenLane

```
OpenLane Container (9dbd8b5):/openlane/designs/regfile_2r1w/runs/full_guide/logs$ ls *
cts:
18-cts.errors  18-cts.log  18-cts.warnings  19-cts_sta.errors  19-cts_sta.log  19-cts_sta.warnings  20-resizer.errors  20-resizer.log  20-resizer.warnings

floorplan:
3-initial_fp.errors  3-initial_fp.log  3-initial_fp.warnings  4-io.errors  4-io.log  4-io.warnings  8-tap.errors  8-tap.log  8-tap.warnings  9-pdn.errors  9-pdn.log  9-pdn.warnings

placement:
11-gpl_sta.errors    12-global.warnings  14-gpl_sta.log       16-detailed.errors    17-dpl_sta.warnings  6-gpl_sta.log        9-global_skip_io.errors
11-gpl_sta.log       12-io.errors        14-gpl_sta.warnings  16-detailed.log       4-global.errors      6-gpl_sta.warnings   9-global_skip_io.log
11-gpl_sta.warnings  12-io.log           15-resizer.errors    16-detailed.warnings  4-global.log         7-basic_mp.errors    9-global_skip_io.warnings
12-global.errors     12-io.warnings      15-resizer.log       17-dpl_sta.errors     4-global.warnings    7-basic_mp.log
12-global.log        14-gpl_sta.errors   15-resizer.warnings  17-dpl_sta.log        6-gpl_sta.errors     7-basic_mp.warnings

routing:
21-resizer_design.errors    23-resizer_timing.errors    25-antenna_diodes_1.log      25-global.warnings               27-grt_sta.warnings  29-detailed.warnings
21-resizer_design.log       23-resizer_timing.log       25-antenna_route_1.errors    25-global_write_netlist.errors   28-fill.errors       30-wire_lengths.log
21-resizer_design.warnings  23-resizer_timing.warnings  25-antenna_route_1.log       25-global_write_netlist.log      28-fill.log
22-rsz_design_sta.errors    24-rsz_timing_sta.errors    25-antenna_route_1.warnings  25-global_write_netlist.warnings  28-fill.warnings
22-rsz_design_sta.log       24-rsz_timing_sta.log       25-global.errors             27-grt_sta.errors                29-detailed.errors
22-rsz_design_sta.warnings  24-rsz_timing_sta.warnings  25-global.log                27-grt_sta.log                   29-detailed.log

signoff:
31-parasitics_extraction.min.errors    34-rcx_mcsta.max.errors              37-irdrop.errors     38-lef.errors        41-spice.log                      45-drc.errors
31-parasitics_extraction.min.log       34-rcx_mcsta.max.log                 37-irdrop.log        38-lef.log           41-spice.warnings                 45-drc.log
31-parasitics_extraction.min.warnings  34-rcx_mcsta.max.warnings            37-irdrop.warnings   38-lef.warnings      42-write_powered_def.log          45-drc.warnings
32-rcx_mcsta.min.errors                35-parasitics_extraction.nom.errors  38-gdsii.errors      38-maglef.errors     42-write_powered_verilog.errors   46-drc-klayout.log
32-rcx_mcsta.min.log                   35-parasitics_extraction.nom.log     38-gdsii.log         38-maglef.log        42-write_powered_verilog.log      47-arc.errors
32-rcx_mcsta.min.warnings              35-parasitics_extraction.nom.warnings  38-gdsii.warnings  38-maglef.warnings   42-write_powered_verilog.warnings  47-arc.log
33-parasitics_extraction.max.errors    36-rcx_mcsta.nom.errors              38-gds_ptrs.errors   39-gdsii-klayout.log  44-lvs.lef.log                    47-arc.warnings
33-parasitics_extraction.max.log       36-rcx_mcsta.nom.log                 38-gds_ptrs.log      40-xor.log           44-regfile_2r1w.lef.lvs.json
33-parasitics_extraction.max.warnings  36-rcx_mcsta.nom.warnings            38-gds_ptrs.warnings  41-spice.errors      44-regfile_2r1w.lef.lvs.log

synthesis:
1-synthesis.errors  1-synthesis.log  1-synthesis.warnings  2-sta.errors  2-sta.log  2-sta.warnings  linter.log
OpenLane Container (9dbd8b5):/openlane/designs/regfile_2r1w/runs/full_guide/logs$
```

実際に実行すると48ステップある

事例紹介：Hierarchcal Chip Designチュートリアルの試行




```
or_gui
# Empty newline to force above line to execute
```

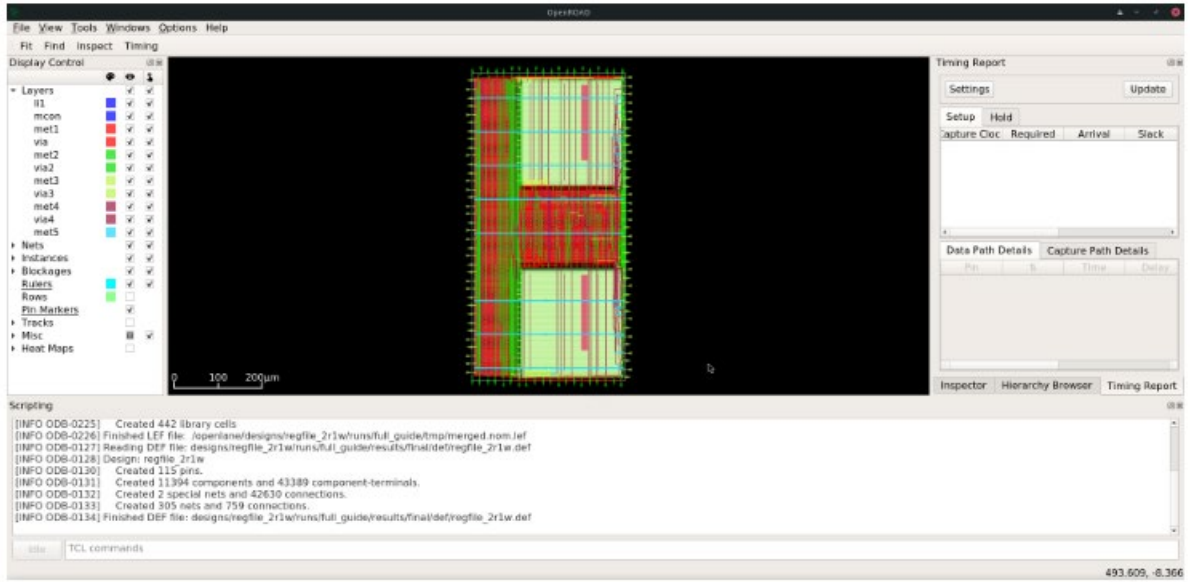


Fig. 2 OpenROAD GUI with loaded final DEF file

手順

1．1Read 1Write Memoryマクロブロックを作成

2．上記マクロブロックを2個並べて2Read 1Write Regfile作成がゴール

（Verilogコード、設定ファイル記述例、実行コマンドあり）

⇒ 1．は特に問題なく実行が完了する。
2．でエラーが起こる

Lintチェック（Verilog文法チェック）で止まっている

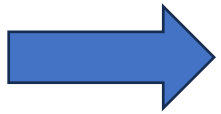
産総研
ともに挑む。つぎを創る。

エラー：マクロブロックのパラメータ継承あたり

```
OpenLane Container (9dbd8b5):/openlane$ more designs/regfile_2r1w/runs/full_guide/logs/synthesis/linter.log
%Error-PINNOTFOUND: /openlane/designs/regfile_2r1w/src/regfile_2r1w.v:39:13: Parameter not found: 'DEPTH_LOG2'
   39 | mem_1r1w #(.DEPTH_LOG2(DEPTH_LOG2), .WIDTH(WIDTH)) lane0(
      |             ^~~~~~~~~~
                    ... For error description see https://verilator.org/warn/PINNOTFOUND?v=5.018
%Error-PINNOTFOUND: /openlane/designs/regfile_2r1w/src/regfile_2r1w.v:39:38: Parameter not found: 'WIDTH'
   39 | mem_1r1w #(.DEPTH_LOG2(DEPTH_LOG2), .WIDTH(WIDTH)) lane0(
      |                                      ^~~~~
%Error-PINNOTFOUND: /openlane/designs/regfile_2r1w/src/regfile_2r1w.v:53:13: Parameter not found: 'DEPTH_LOG2'
   53 | mem_1r1w #(.DEPTH_LOG2(DEPTH_LOG2), .WIDTH(WIDTH)) lane1(
      |             ^~~~~~~~~~
%Error-PINNOTFOUND: /openlane/designs/regfile_2r1w/src/regfile_2r1w.v:53:38: Parameter not found: 'WIDTH'
   53 | mem_1r1w #(.DEPTH_LOG2(DEPTH_LOG2), .WIDTH(WIDTH)) lane1(
      |                                      ^~~~~
%Error: Exiting due to 4 error(s)
       ... See the manual at https://verilator.org/verilator_doc.html for more assistance.
OpenLane Container (9dbd8b5):/openlane$
```

対策：モジュール呼び出しからパラメータを削除

```
mem_1r1w #(.DEPTH_LOG2(DEPTH_LOG2), .WIDTH(WIDTH)) lane0(
    .clk(clk),

    .read_addr(rs1_addr),
    .read(rs1_read),
    .read_data(rs1_rdata),

    .write(write),
    .write_addr(write_addr),
    .write_data(write_data)
);



mem_1r1w #(.DEPTH_LOG2(DEPTH_LOG2), .WIDTH(WIDTH)) lane1(
    .clk(clk),

    .read_addr(rs2_addr),
    .read(rs2_read),
    .read_data(rs2_rdata),

    .write(write),
    .write_addr(write_addr),
    .write_data(write_data)
);


endmodule


U:---  regfile_2r1w.v   All L45    (Verilog)
<C-M-print> is undefined
```



```
mem_1r1w lane0(
    .clk(clk),

    .read_addr(rs1_addr),
    .read(rs1_read),
    .read_data(rs1_rdata),

    .write(write),
    .write_addr(write_addr),
    .write_data(write_data)
);



mem_1r1w lane1(
    .clk(clk),

    .read_addr(rs2_addr),
    .read(rs2_read),
    .read_data(rs2_rdata),

    .write(write),
    .write_addr(write_addr),
    .write_data(write_data)
);


endmodule


U:---  regfile_2r1w.v   All L54    (Verilog)
Wrote /home/masa/OpenLane/designs/regfile_2r1w/src/regfile_2r1w.v
```

今度はSTAで止まっている

産総研
ともに挑む。つぎを創る。

エラー：“ブラックボックスファイルがRTLでなくゲートレベルネットリストであることを確認すべし”

```
1-synthesis.log        2-sta.errors           2-sta.warnings
OpenLane Container (9dbd8b5):/openlane$ more designs/regfile_2r1w/runs/full_guide/logs/synthesis/2-sta.log
OpenSTA 2.4.0 75f2f325b7 Copyright (c) 2023, Parallax Software, Inc.
License GPLv3: GNU GPL version 3 <http://gnu.org/licenses/gpl.html>

This is free software, and you are free to change and redistribute it
under certain conditions; type `show_copying' for details.
This program comes with ABSOLUTELY NO WARRANTY; for details type `show_warranty'.
define_corners Typical
read_liberty -corner Typical /home/masa/.volare/sky130A/libs.ref/sky130_fd_sc_hd/lib/sky130_fd_sc_hd__tt_025C_1v80.lib
Using 1e-12 for capacitance...
Using 1e+03 for resistance...
Using 1e-09 for time...
Using 1e+00 for voltage...
Using 1e-03 for current...
Using 1e-09 for power...
Using 1e-06 for distance...
Reading netlist '/openlane/designs/regfile_2r1w/runs/full_guide/results/synthesis/regfile_2r1w.v'…
Error while reading /openlane/designs/regfile_2r1w/bb/mem_1r1w.bb.v:
Make sure that this a gate-level netlist not an RTL file
You can add the following comment '/// sta-blackbox' in the file to skip it and blackbox the modules inside if needed.
Error: /openlane/designs/regfile_2r1w/bb/mem_1r1w.bb.v line 5, syntax error, unexpected '=', expecting '('
```

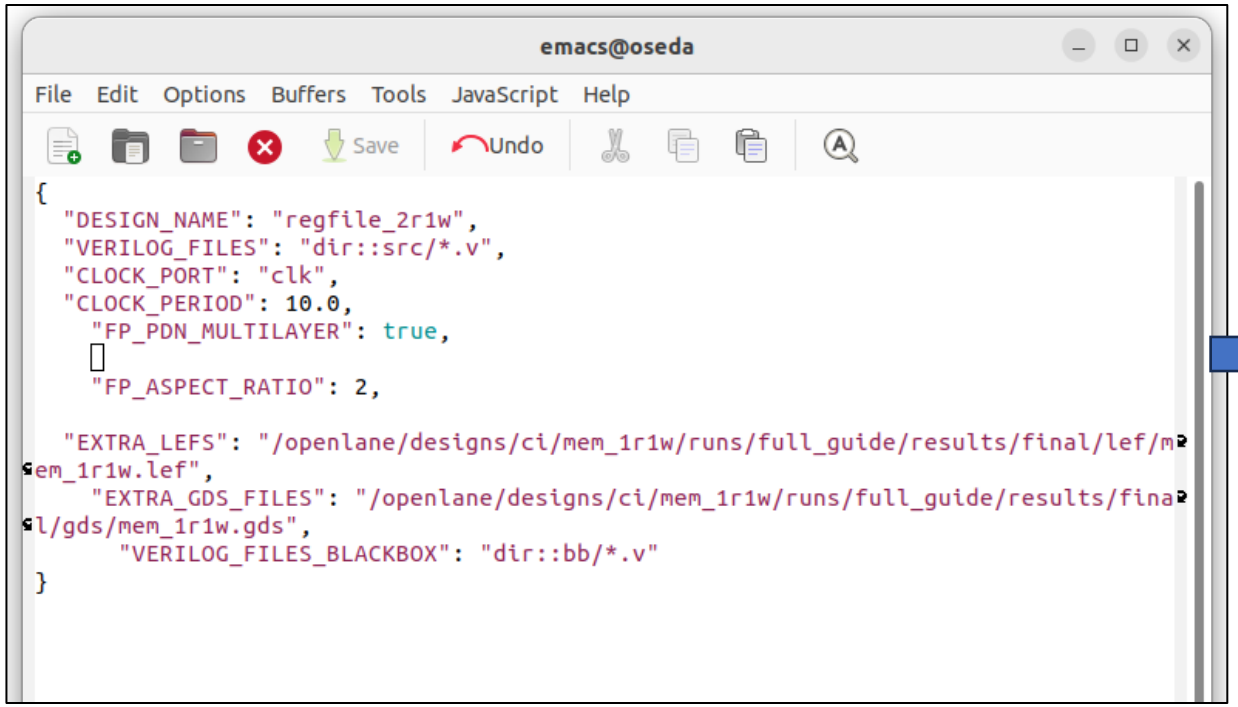
チュートリアルに記載のブラックボックスファイルの例題
⇒おもいっきりRTL、、、

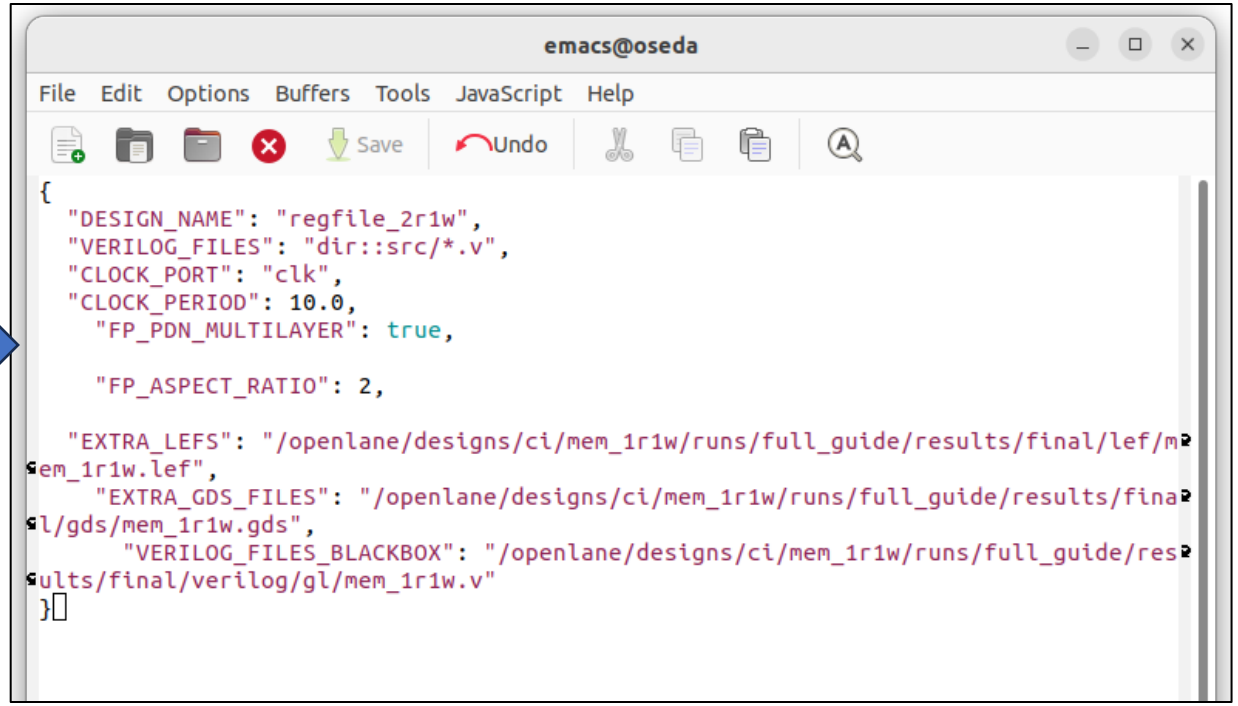
Create the verilog blackbox:

```
(*blackbox*)

module mem_1r1w (clk, read_addr, read, read_data, write_addr, write, write_data);
        parameter DEPTH_LOG2 = 4;
        localparam ELEMENTS = 2**DEPTH_LOG2;
        parameter WIDTH = 32;

        input wire clk;

    input wire [DEPTH_LOG2-1:0] read_addr;
    input wire read;
        output reg [WIDTH-1:0] read_data;


        input wire [DEPTH_LOG2-1:0] write_addr;
        input wire write;
        input wire  [WIDTH-1:0] write_data;

endmodule
```

対策：ブラックボックスファイルとしてゲートレベルネットリストを使用

1r1wメモリマクロブロック作成時にゲートレベルネットリストが既に作成されているので
ブラックボックスファイルとしてパスを通す





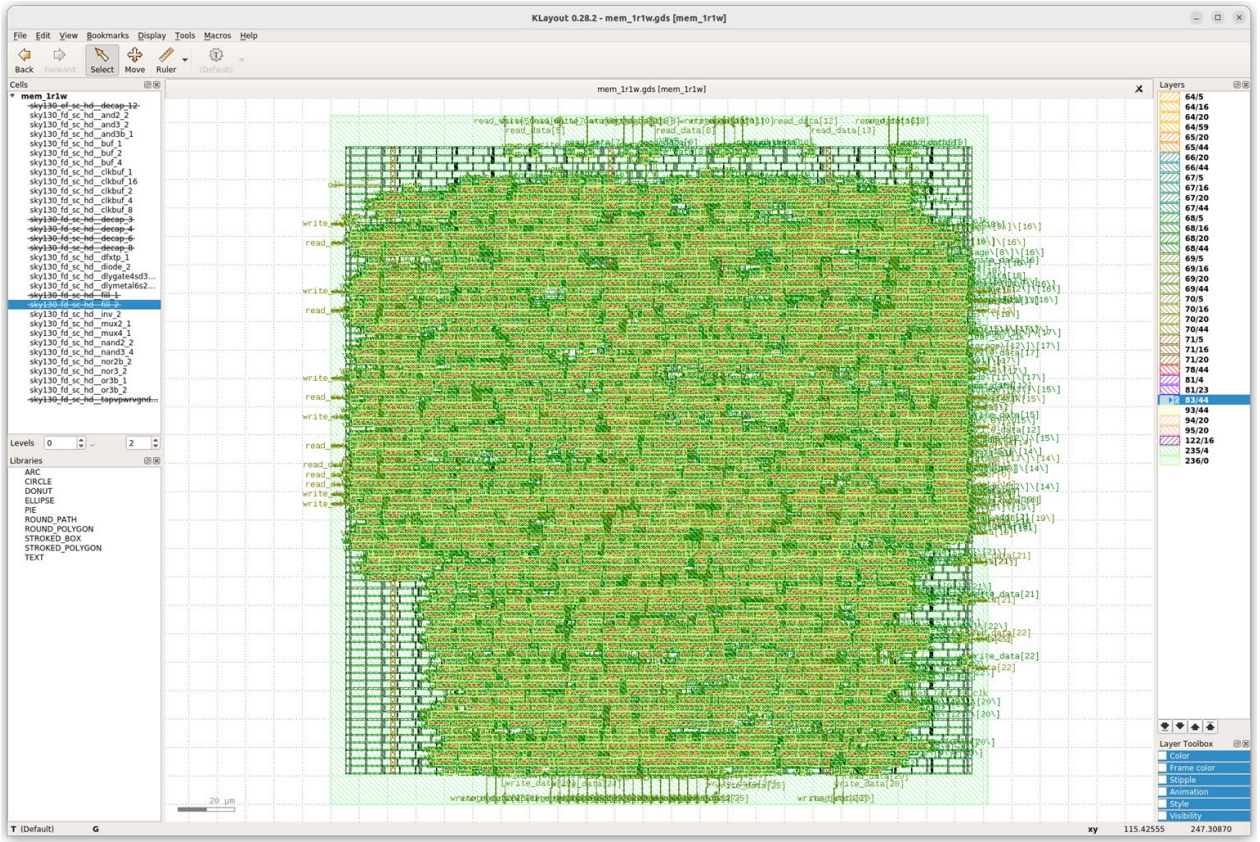
Flow Complete!

```
eports/signoff/36-sta-rcx_nom/multi_corner_sta.checks.rpt'.
[INFO]: There are no hold violations in the design at the Typical corner.
[INFO]: There are no setup violations in the design at the Typical corner.
[SUCCESS]: Flow complete.
[INFO]: Note that the following warnings have been generated:
[WARNING]: PNR_SDC_FILE is not set. It is recommended to write a custom SDC file for the design. Defaulting to BASE_SDC_FILE
[WARNING]: SIGNOFF_SDC_FILE is not set. It is recommended to write a custom SDC file for the design. Defaulting to BASE_SDC_FILE
[WARNING]: 4 warnings found by linter
[WARNING]: VSRC_LOC_FILES was not given a value, which may make the results of IR drop analysis inaccurate. If you are not integrating
a top-level chip for manufacture, you may ignore this warning, otherwise, see the documentation for VSRC_LOC_FILES.
[WARNING]: There are max fanout violations in the design at the Typical corner. Please refer to 'designs/regfile_2r1w/runs/full_guide/r
eports/signoff/36-sta-rcx_nom/multi_corner_sta.checks.rpt'.

OpenLane Container (9dbd8b5):/openlane$
```

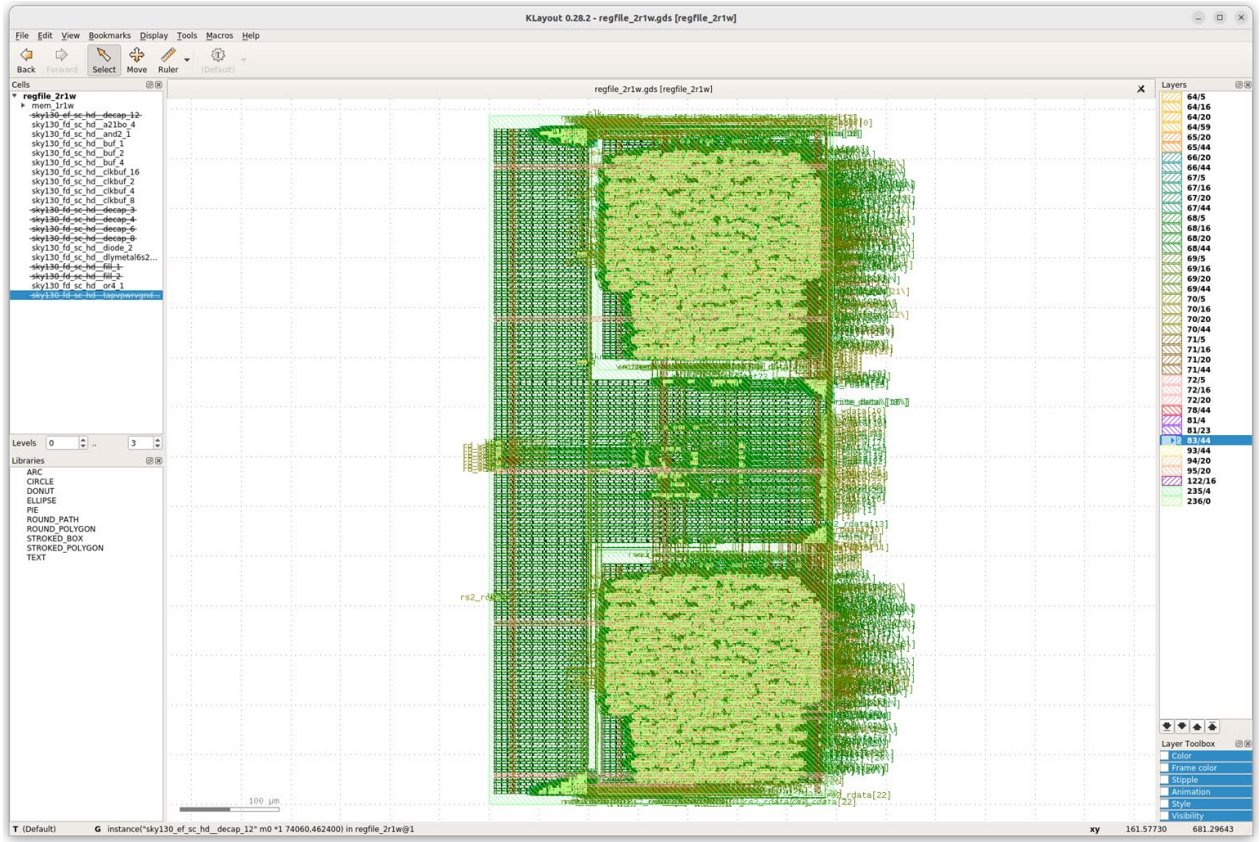
GDSファイルの生成結果

産総研
ともに挑む。つぎを創る。

Mem 1r1w



regfile 2r1w



- ツールは安定版に移行しているらしい
- ドキュメントの更新が追い付いていない？

2023年度の進捗： HWセキュリティ向け回路IPの開発

Summary

- HWセキュリティシステム用のRISC-V、疑似PUF回路、誤り訂正符号、楕円曲線暗号の回路IP開発を目標
- RISC-Vと疑似PUF回路は回路記述ファイル（RTL）・機能検証・レイアウト生成を完了。
- 誤り訂正符号と楕円曲線暗号はC言語で作成完了。
  ⇒Vivadoを使ってC言語からRTL変換したら不要なモジュールがたくさん生成されてP&Rが進まない。高位合成ツール模索中

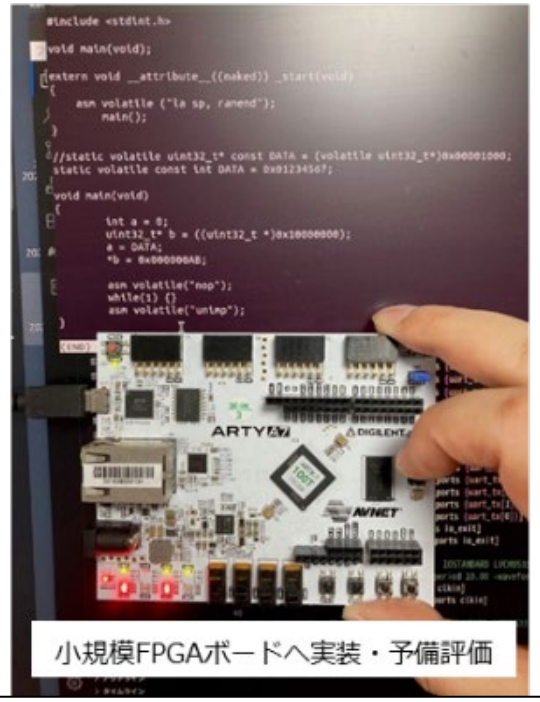
PUFシステム向けプロセッサの開発

公開されている
32-bit サブセット
RISC-Vコア

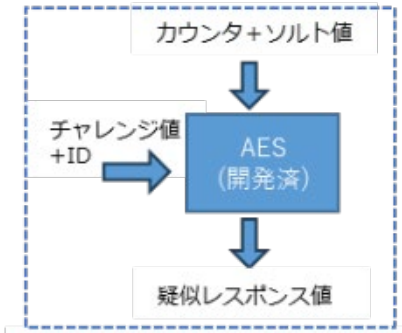
Chisel高位言語を利用
PUF制御に必要な機構や
インタフェースの追加、
不要な機能の削除など改修

改修版
32-bit サブセット
RISC-Vコア

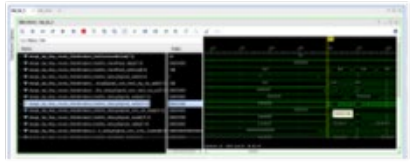
RISC-V用コード
生成の環境構築



疑似PUF回路の設計



AESカウンタモードを利用した
疑似ID生成を予定



AES単体のシミュレーション

PUFシステム向けモジュールの検証

OpenLANEによる機能モジュールの合成と検証

改修版
32-bit サブセット
RISC-Vコア
(Chisel高位合成)

AES暗号コア
(Verilog HDL)

→ 論理合成と
配置配線を確認

BCH誤り訂正符号
(C言語高位合成)

→ 論理合成のみ確認

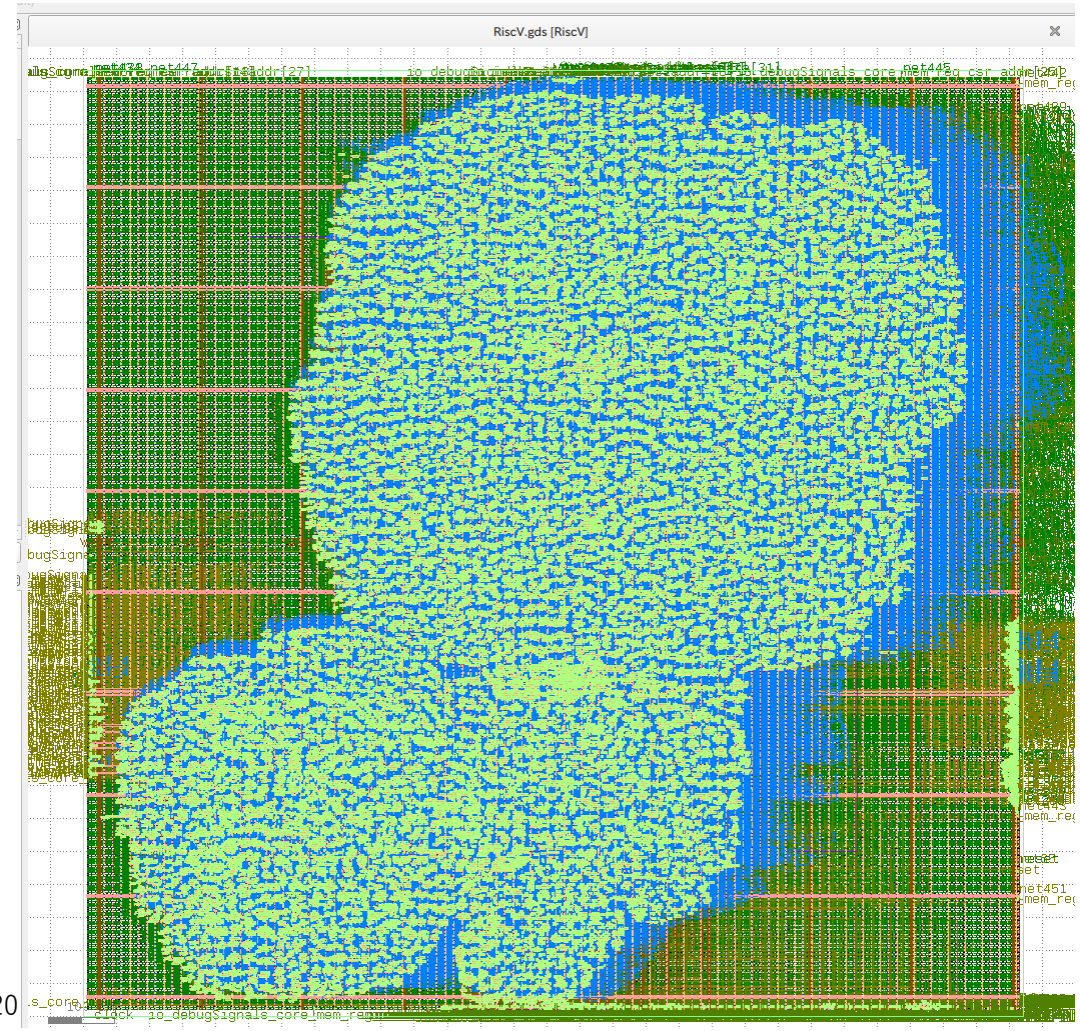
楕円曲線暗号
ECDH
(C言語高位合成)

→ 論理合成も難しい

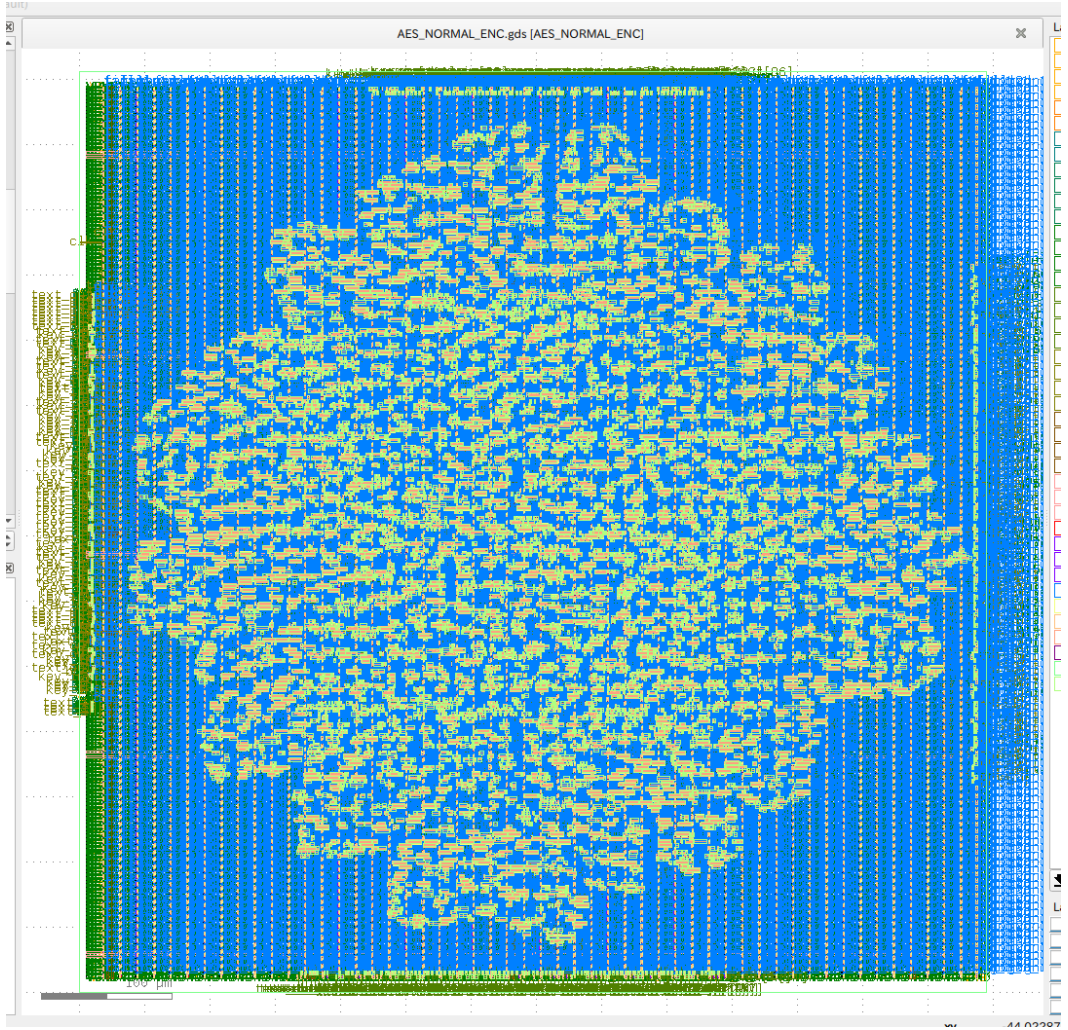
FPGA向け高位合成のコードであり
FPGA機能を利用しない改修などが課題

GDS生成結果

RISC-V
Area: 1400um□
# of cells: 28267



疑似PUF
Area: 700um□
# of cells: 6216

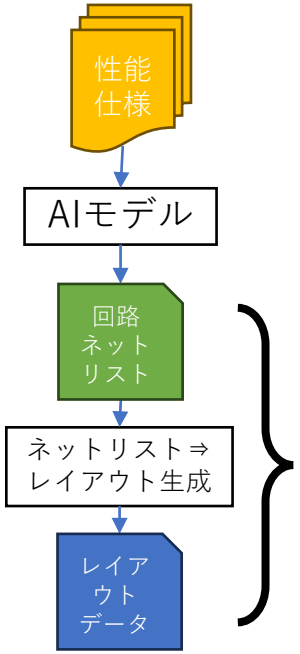


2023年度の進捗： アナログ設計環境の拡張

Summary

- オープンソースの回路ネットリスト⇒レイアウト生成ツールを用いてオペアンプを例題に回路ネットリストからskywater130nmのレイアウト生成環境を整備
- 加えてオープンソースEDAで完結する形でデザインルール検証環境（DRC）、ネットリストとレイアウトの整合性検証環境（LVS）も整備

拡張アナログ設計フロー

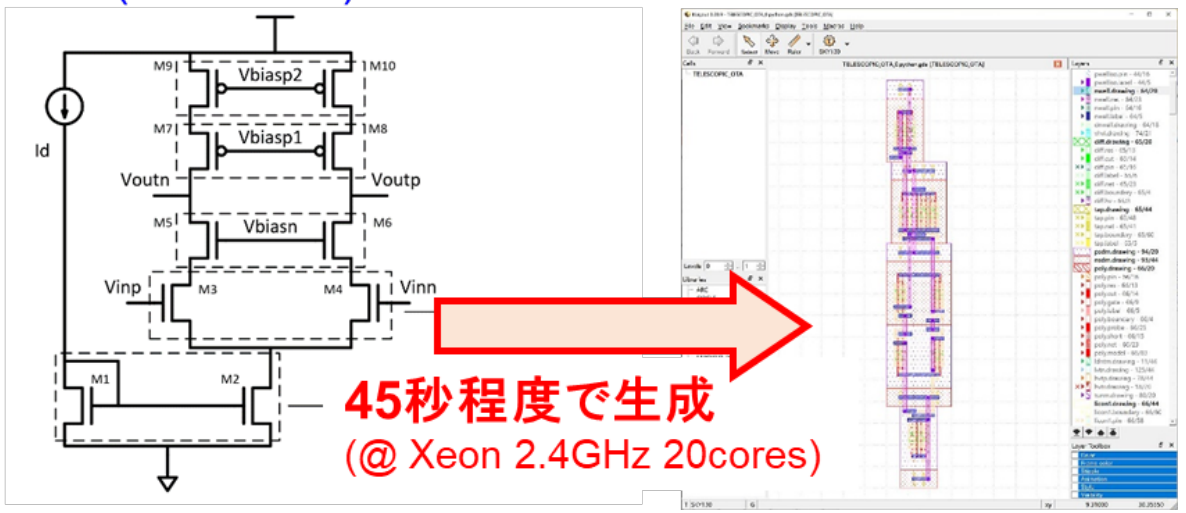


- 回路ネットリスト⇒レイアウト生成ツール“ALIGN”
- Sky130のパッシブ素子まわりの設定修正
- DRC、LVSまわりも整備

オペアンプを用いた変換例

回路図(ネットリスト)を入力
(例はアンプ)

GDSを出力



ALIGN: Analog Layout, Intelligently Generated from Netlists

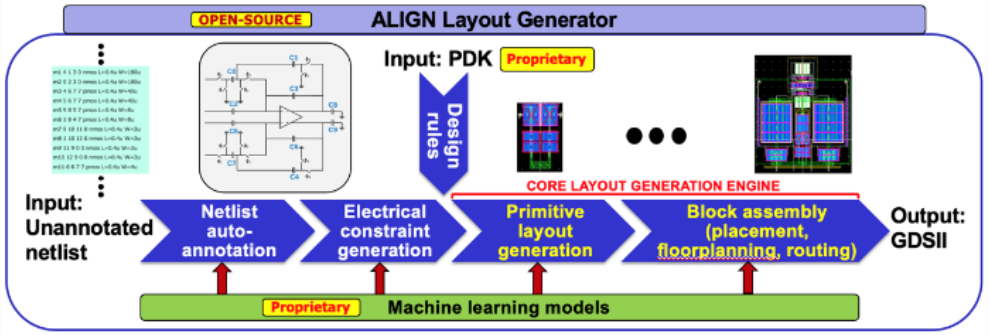
ALIGN: flow — ALIGN

https://align-analoglayout.github.io/ALIGN-public/notes/flow.html

ALIGN: flow

The ALIGN flow includes the following steps:

- *Circuit annotation* creates a multilevel hierarchical representation of the input netlist. This representation is used to implement the circuit layout in using a hierarchical manner.
- *Design rule abstraction* creates a compact JSON-format represetation of the design rules in a PDK. This repository provides a mock PDK based on a FinFET technology (where the parameters are based on published data). These design rules are used to guide the layout and ensure DRC-correctness.
- *Primitive cell generation* works with primitives, i.e., blocks the lowest level of design hierarchy, and generates their layouts. Primitives typically contain a small number of transistor structures (each of which may be implemented using multiple fins and/or fingers). A parameterized instance of a primitive is automatically translated to a GDSII layout in this step.
- *Placement and routing* performs block assembly of the hierarchical blocks in the netlist and routes connections between these blocks, while obeying a set of analog layout constraints. At the end of this step, the translation of the input SPICE netlist to a GDSII layout is complete.

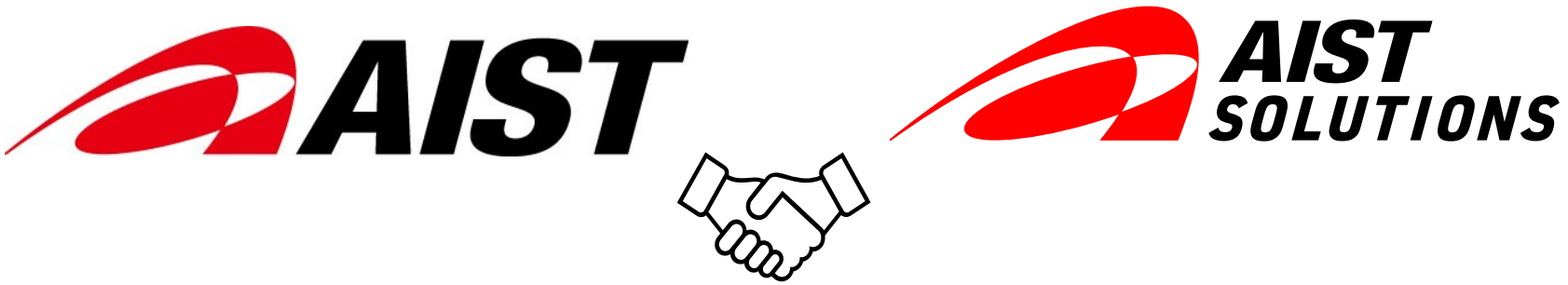


Inputs

AIST SOLUTIONSとのコラボレーション

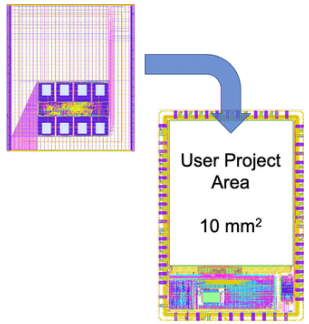
本研究で得られた知見や情報を産総研単独で広く公開するのは簡単ではない
- ヒューマンリソース
- コネクション
- メソッド　etc

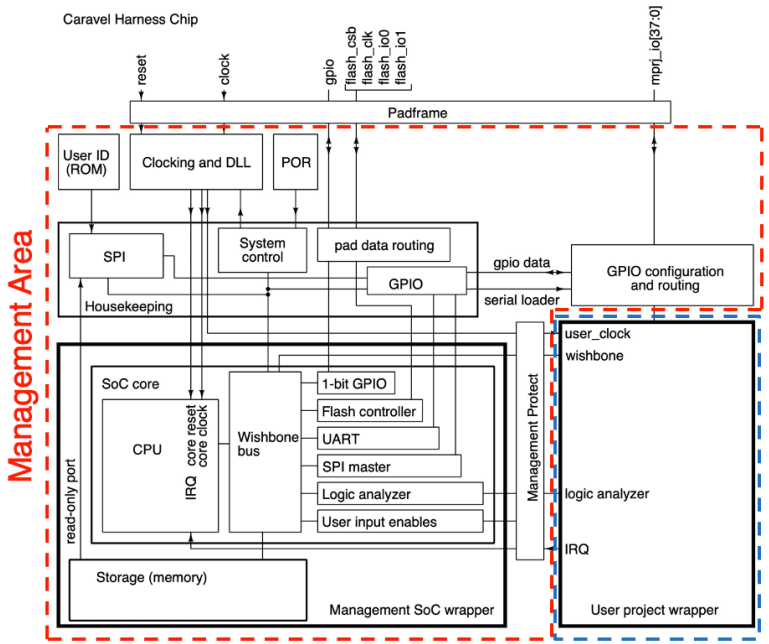
⇒AIST SOLUTIONSで加速
- Open Source Silicon活用検証委員会を立ち上げ
- 委員会活動の一環で本研究で得た知見を公開予定
- efablessのChipIgniteシャトルでチップ試作の機会

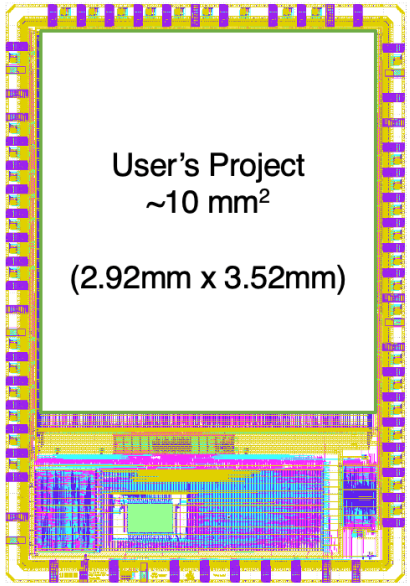


ChipIgniteシャトルサービスChipIgniteシャトルサービス

- ChipIgniteシャトル：efablessによる有償シャトル（$9,750）
- Caravelハーネス、オープンPDK、Openlaneからなる統合環境
- ユーザ回路のVerilog to GDS、ハーネスへの配置、テープアウトチェックまでほぼ自動
- シミュレーション環境もある
- QFNチップ100個、評価ボード付







- 38 programmable IO's
- 10 mm2 of User Project Area
- Diagnostic port including IO configuration and Flash pass-thru access for programming
- VexRiscv CPU with debug
- 3 kbytes of RAM
- Flash controller supporting execute-in-place
- SPI, UART and GPIO
- Counter/Timers
- 128 signal logic analyzer for project

ChipIgniteシャトルでのTapeout方針

- efablessから提供されているexampleファイル（カウンター回路）を活用し、回路IPを組み込む
- Exampleファイルは必要最小限の変更に留める
  - IPポート名、IPポート数
  - ピン配置
  - もともと存在するモジュールに手を付けない
- 回路IPは公開ベースのものを使用（人為的エラー回避、設計時間の短縮）

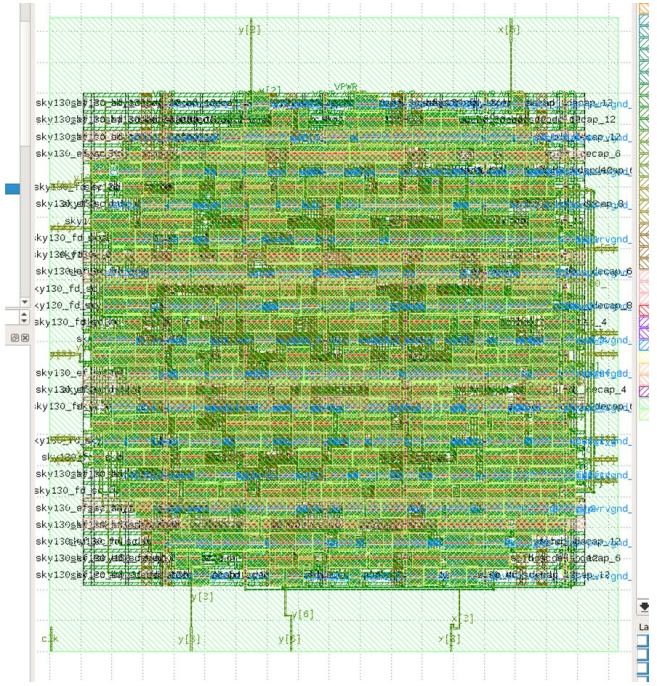
Our design: AES SBOX-like function

産総研
ともに挑む。つぎを創る。

```
ファイル(F)  編集(E)  書式(O)  表示(V)  ヘルプ(H)
//============================================== AES_SBOX_ENC
module AES_SBOX_ENC
  (input  wire [7:0] x,
   output wire [7:0] y);

   //----------------------------------------------
   assign y = s(x[7:0]);

   function [7:0] s (input [7:0] x);
      case (x)
        8'h00: s=8'h63;  8'h01: s=8'h7c;  8'h02: s=8'h77;  8'h03: s=8'h7b;
        8'h04: s=8'hf2;  8'h05: s=8'h6b;  8'h06: s=8'h6f;  8'h07: s=8'hc5;
        8'h08: s=8'h30;  8'h09: s=8'h01;  8'h0A: s=8'h67;  8'h0B: s=8'h2b;
        8'h0C: s=8'hfe;  8'h0D: s=8'hd7;  8'h0E: s=8'hab;  8'h0F: s=8'h76;

        8'h10: s=8'hca;  8'h11: s=8'h82;  8'h12: s=8'hc9;  8'h13: s=8'h7d;
        8'h14: s=8'hfa;  8'h15: s=8'h59;  8'h16: s=8'h47;  8'h17: s=8'hf0;
        8'h18: s=8'had;  8'h19: s=8'hd4;  8'h1A: s=8'ha2;  8'h1B: s=8'haf;
        8'h1C: s=8'h9c;  8'h1D: s=8'ha4;  8'h1E: s=8'h72;  8'h1F: s=8'hc0;

        8'h20: s=8'hb7;  8'h21: s=8'hfd;  8'h22: s=8'h93;  8'h23: s=8'h26;
        8'h24: s=8'h36;  8'h25: s=8'h3f;  8'h26: s=8'hf7;  8'h27: s=8'hcc;
        8'h28: s=8'h34;  8'h29: s=8'ha5;  8'h2A: s=8'he5;  8'h2B: s=8'hf1;
        8'h2C: s=8'h71;  8'h2D: s=8'hd8;  8'h2E: s=8'h31;  8'h2F: s=8'h15;

        8'h30: s=8'h04;  8'h31: s=8'hc7;  8'h32: s=8'h23;  8'h33: s=8'hc3;
        8'h34: s=8'h18;  8'h35: s=8'h96;  8'h36: s=8'h05;  8'h37: s=8'h9a;
        8'h38: s=8'h07;  8'h39: s=8'h12;  8'h3A: s=8'h80;  8'h3B: s=8'he2;
        8'h3C: s=8'heb;  8'h3D: s=8'h27;  8'h3E: s=8'hb2;  8'h3F: s=8'h75;

        8'h40: s=8'h09;  8'h41: s=8'h83;  8'h42: s=8'h2c;  8'h43: s=8'h1a;
        8'h44: s=8'h1b;  8'h45: s=8'h6e;  8'h46: s=8'h5a;  8'h47: s=8'ha0;
        8'h48: s=8'h52;  8'h49: s=8'h3b;  8'h4A: s=8'hd6;  8'h4B: s=8'hb3;
        8'h4C: s=8'h29;  8'h4D: s=8'he3;  8'h4E: s=8'h2f;  8'h4F: s=8'h84;

        8'h50: s=8'h53;  8'h51: s=8'hd1;  8'h52: s=8'h00;  8'h53: s=8'hed;
        8'h54: s=8'h20;  8'h55: s=8'hfc;  8'h56: s=8'hb1;  8'h57: s=8'h5b;
        8'h58: s=8'h6a;  8'h59: s=8'hcb;  8'h5A: s=8'hbe;  8'h5B: s=8'h39;
        8'h5C: s=8'h4a;  8'h5D: s=8'h4c;  8'h5E: s=8'h58;  8'h5F: s=8'hcf;

        8'h60: s=8'hd0;  8'h61: s=8'hef;  8'h62: s=8'haa;  8'h63: s=8'hfb;
        8'h64: s=8'h43;  8'h65: s=8'h4d;  8'h66: s=8'h33;  8'h67: s=8'h85;
        8'h68: s=8'h45;  8'h69: s=8'hf9;  8'h6A: s=8'h02;  8'h6B: s=8'h7f;
        8'h6C: s=8'h50;  8'h6D: s=8'h3c;  8'h6E: s=8'h9f;  8'h6F: s=8'ha8;

        8'h70: s=8'h51;  8'h71: s=8'ha3;  8'h72: s=8'h40;  8'h73: s=8'h8f;
        8'h74: s=8'h92;  8'h75: s=8'h9d;  8'h76: s=8'h38;  8'h77: s=8'hf5;
```

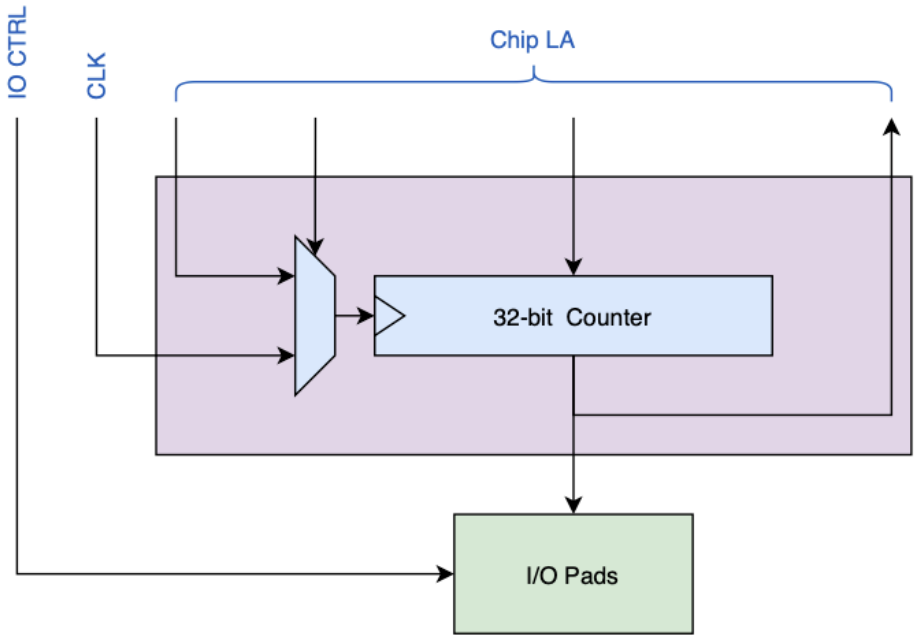
- 8bit inputs and outputs
- Defines 8bit functions
- Output a unique value corresponding to an input

OpenLaneによるGDS生成結果（90um□）



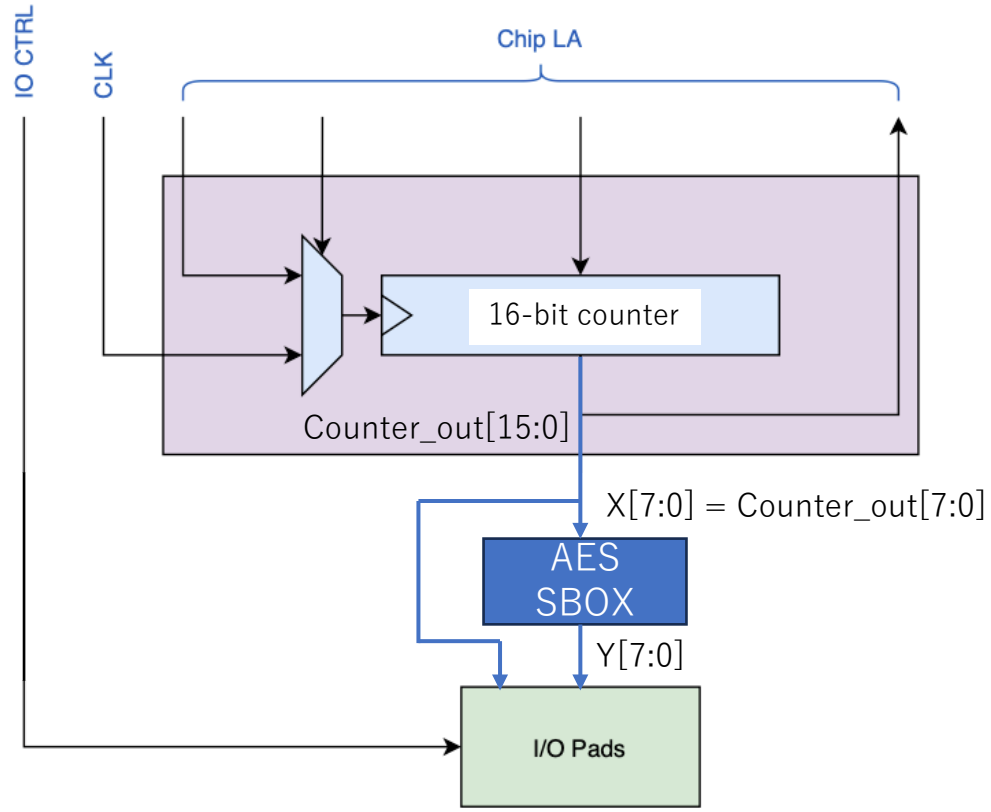
Embedding AES SBOX into the counter example

Original counter circuit example
in a caravel user project

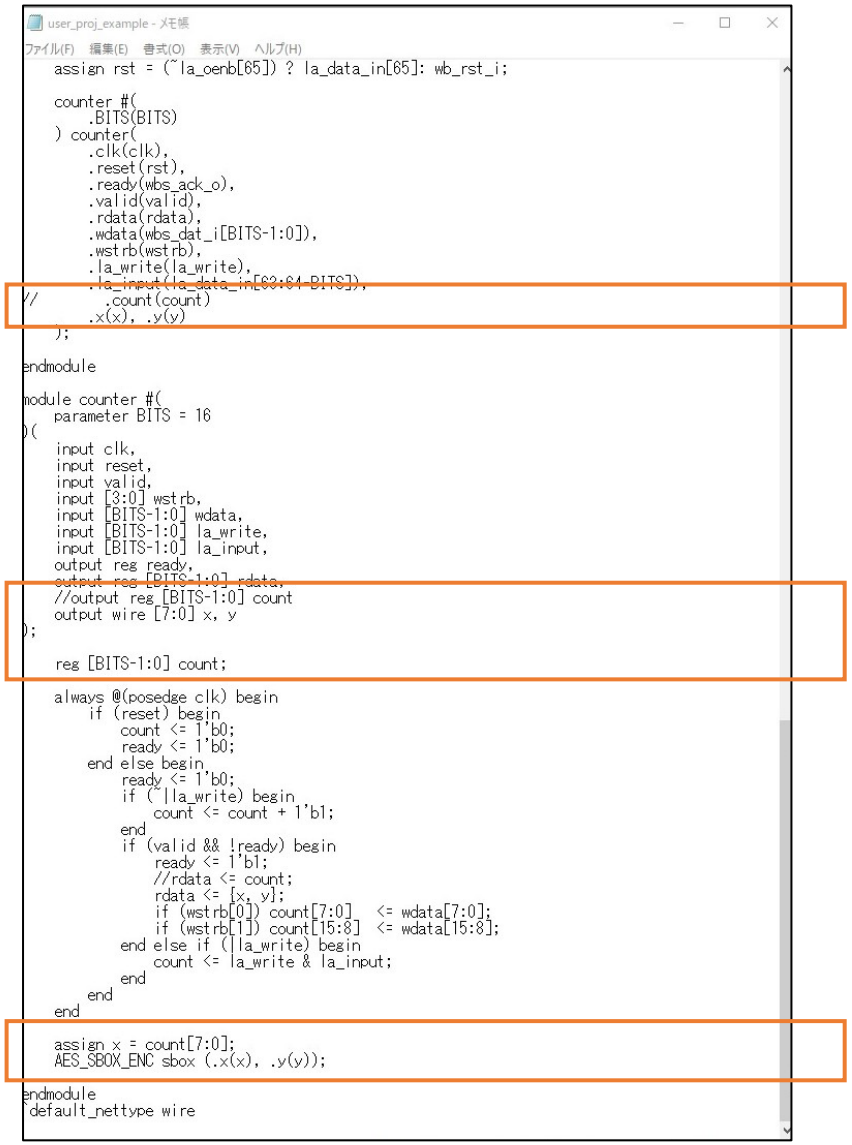


Change

Our design



Change points in User_proj_example.v



```
user_proj_example - メモ帳
ファイル(F)  編集(E)  書式(O)  表示(V)  ヘルプ(H)
    assign rst = (~la_oenb[65]) ? la_data_in[65]: wb_rst_i;

    counter #(
        .BITS(BITS)
    ) counter(
        .clk(clk),
        .reset(rst),
        .ready(wbs_ack_o),
        .valid(valid),
        .rdata(rdata),
        .wdata(wbs_dat_i[BITS-1:0]),
        .wstrb(wstrb),
        .la_write(la_write),
        .la_input(la_data_in[63:64-BITS]),
//        .count(count)
        .x(x), .y(y)
    );

endmodule

module counter #(
    parameter BITS = 16
)(
    input clk,
    input reset,
    input valid,
    input [3:0] wstrb,
    input [BITS-1:0] wdata,
    input [BITS-1:0] la_write,
    input [BITS-1:0] la_input,
    output reg ready,
    output reg [BITS-1:0] rdata,
    //output reg [BITS-1:0] count
    output wire [7:0] x, y
);

    reg [BITS-1:0] count;

    always @(posedge clk) begin
        if (reset) begin
            count <= 1'b0;
            ready <= 1'b0;
        end else begin
            ready <= 1'b0;
            if (~|la_write) begin
                count <= count + 1'b1;
            end
            if (valid && !ready) begin
                ready <= 1'b1;
                //rdata <= count;
                rdata <= {x, y};
                if (wstrb[0]) count[7:0]   <= wdata[7:0];
                if (wstrb[1]) count[15:8]  <= wdata[15:8];
            end else if (|la_write) begin
                count <= la_write & la_input;
            end
        end
    end

    assign x = count[7:0];
    AES_SBOX_ENC sbox (.x(x), .y(y));

endmodule
`default_nettype wire
```

- In counter subcircuit:
  - Comment out original “counter” port
  - Add “x/y” ports for sbox
- In counter module:
  - Comment out original “count” output port
  - Add wire type “x/y” output for sbox
  - Add register type “count”
  - Assign “x” = “count”
  - Call sbox subcircuit

Simulation results on a caravel environment

Function definitions in Verilog code of AES SBOX

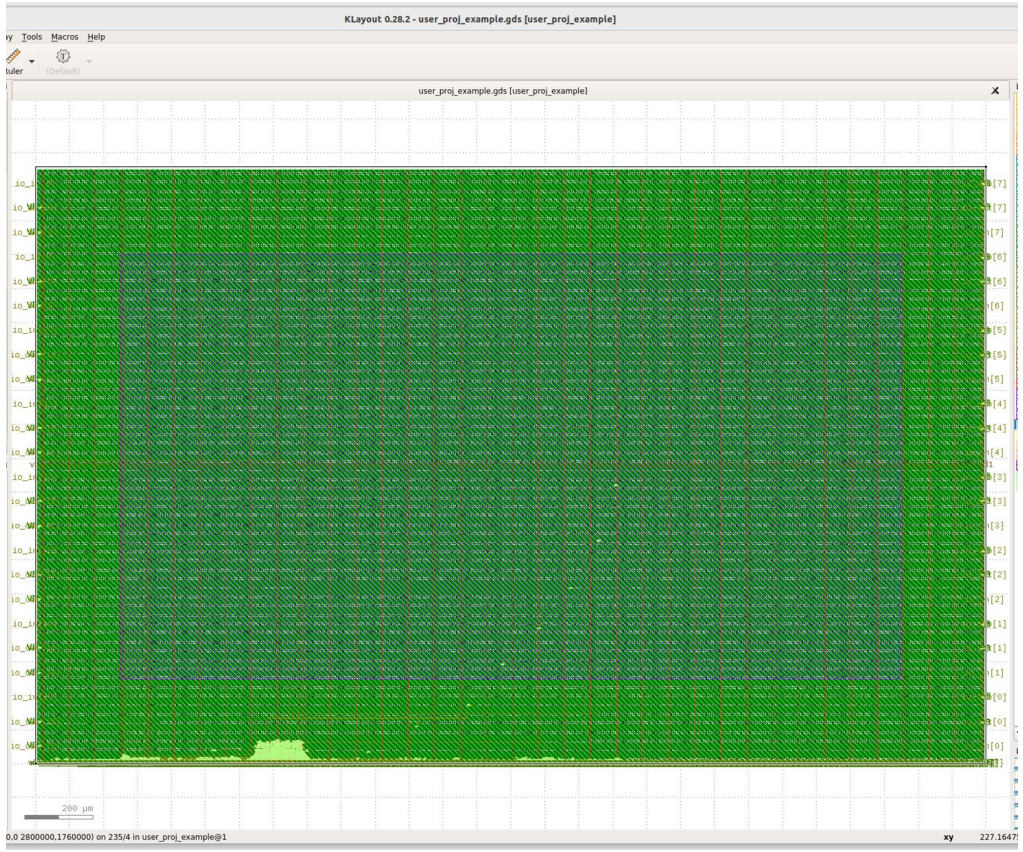
```
8'h90: s=8'h60;  8'h91: s=8'h81;  8'h92: s=8'h4f;  8'h93: s=8'hdc;
8'h94: s=8'h22;  8'h95: s=8'h2a;  8'h96: s=8'h90;  8'h97: s=8'h88;
8'h98: s=8'h46;  8'h99: s=8'hee;  8'h9A: s=8'hb8;  8'h9B: s=8'h14;
8'h9C: s=8'hde;  8'h9D: s=8'h5e;  8'h9E: s=8'h0b;  8'h9F: s=8'hdb;

8'hA0: s=8'he0;  8'hA1: s=8'h32;  8'hA2: s=8'h3a;  8'hA3: s=8'h0a;
8'hA4: s=8'h49;  8'hA5: s=8'h06;  8'hA6: s=8'h24;  8'hA7: s=8'h5c;
8'hA8: s=8'hc2;  8'hA9: s=8'hd3;  8'hAA: s=8'hac;  8'hAB: s=8'h62;
8'hAC: s=8'h91;  8'hAD: s=8'h95;  8'hAE: s=8'he4;  8'hAF: s=8'h79;

8'hB0: s=8'he7;  8'hB1: s=8'hc8;  8'hB2: s=8'h37;  8'hB3: s=8'h6d;
8'hB4: s=8'h8d;  8'hB5: s=8'hd5;  8'hB6: s=8'h4e;  8'hB7: s=8'ha9;
8'hB8: s=8'h6c;  8'hB9: s=8'h56;  8'hBA: s=8'hf4;  8'hBB: s=8'hea;
8'hBC: s=8'h65;  8'hBD: s=8'h7a;  8'hBE: s=8'hae;  8'hBF: s=8'h08;

8'hC0: s=8'hba;  8'hC1: s=8'h78;  8'hC2: s=8'h25;  8'hC3: s=8'h2e;
8'hC4: s=8'h1c;  8'hC5: s=8'ha6;  8'hC6: s=8'hb4;  8'hC7: s=8'hc6;
```

Simulation results

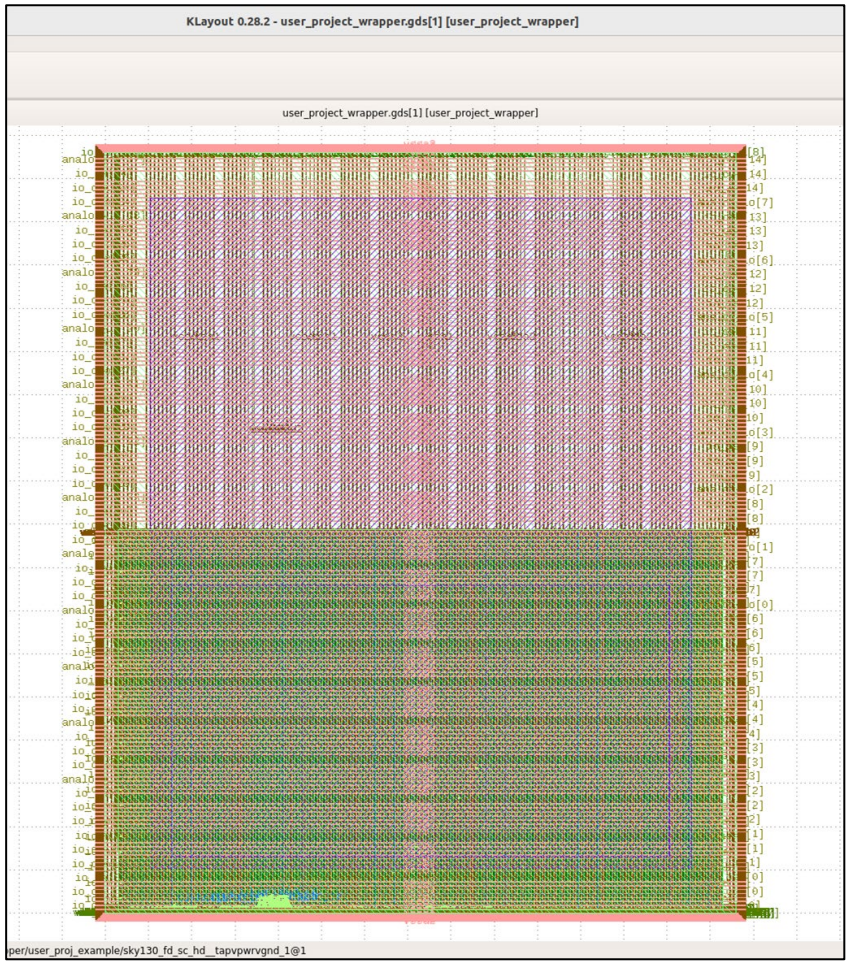
```
Reading io_ports.hex
io_ports.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
FST info: dumpfile io_ports.vcd opened for output.
MPRJ-IO state = zz zz : zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz1zzz
MPRJ-IO state = db 9f : zzzzzzzzzzzzz1001111111011011zzzz1zzz
MPRJ-IO state = e0 a0 : zzzzzzzzzzzzz1010000011100000zzzz1zzz
MPRJ-IO state = 32 a1 : zzzzzzzzzzzzz1010000100110010zzzz1zzz
...
MPRJ-IO state = b0 fc : zzzzzzzzzzzzz1111110010110000zzzz1zzz
MPRJ-IO state = 54 fd : zzzzzzzzzzzzz1111110101010100zzzz1zzz
MPRJ-IO state = bb fe : zzzzzzzzzzzzz1111111010111011zzzz1zzz
MPRJ-IO state = 16 ff : zzzzzzzzzzzzz1111111100010110zzzz1zzz
Monitor: Test 1 Mega-Project IO (RTL) Passed
io_ports_tb.v:208: $finish called at 494862500 (1ps)
mv io_ports.vcd RTL-io_ports.vcd
rm io_ports.vvp
```

User領域のGDSファイル生成

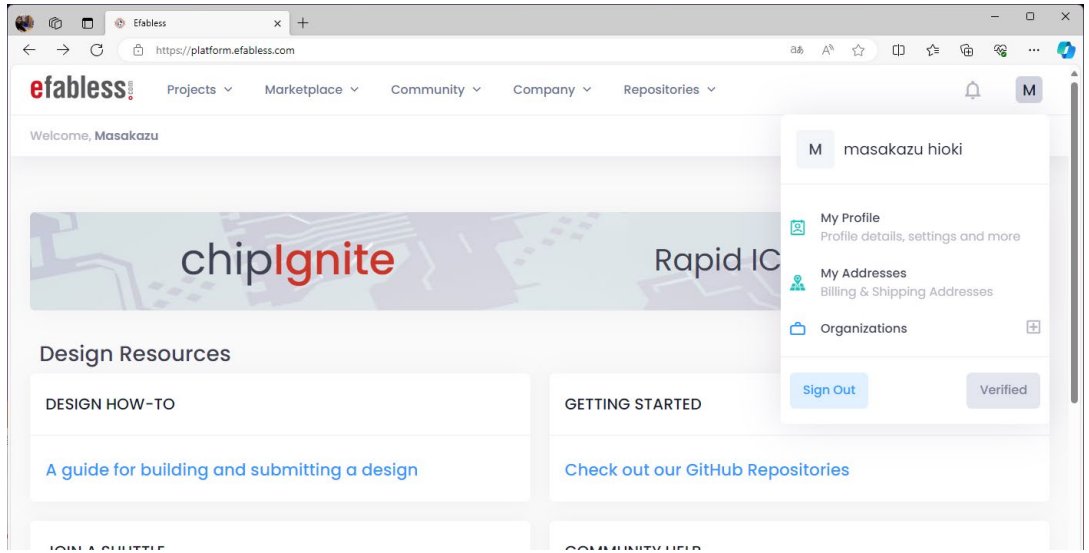
Example counter circuit + our sbox

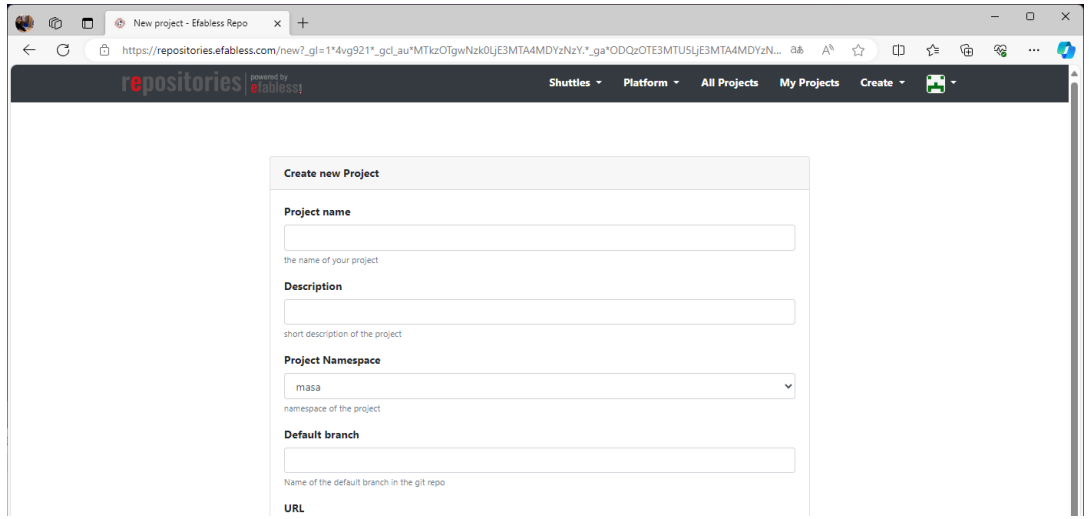


User area gds including our IP



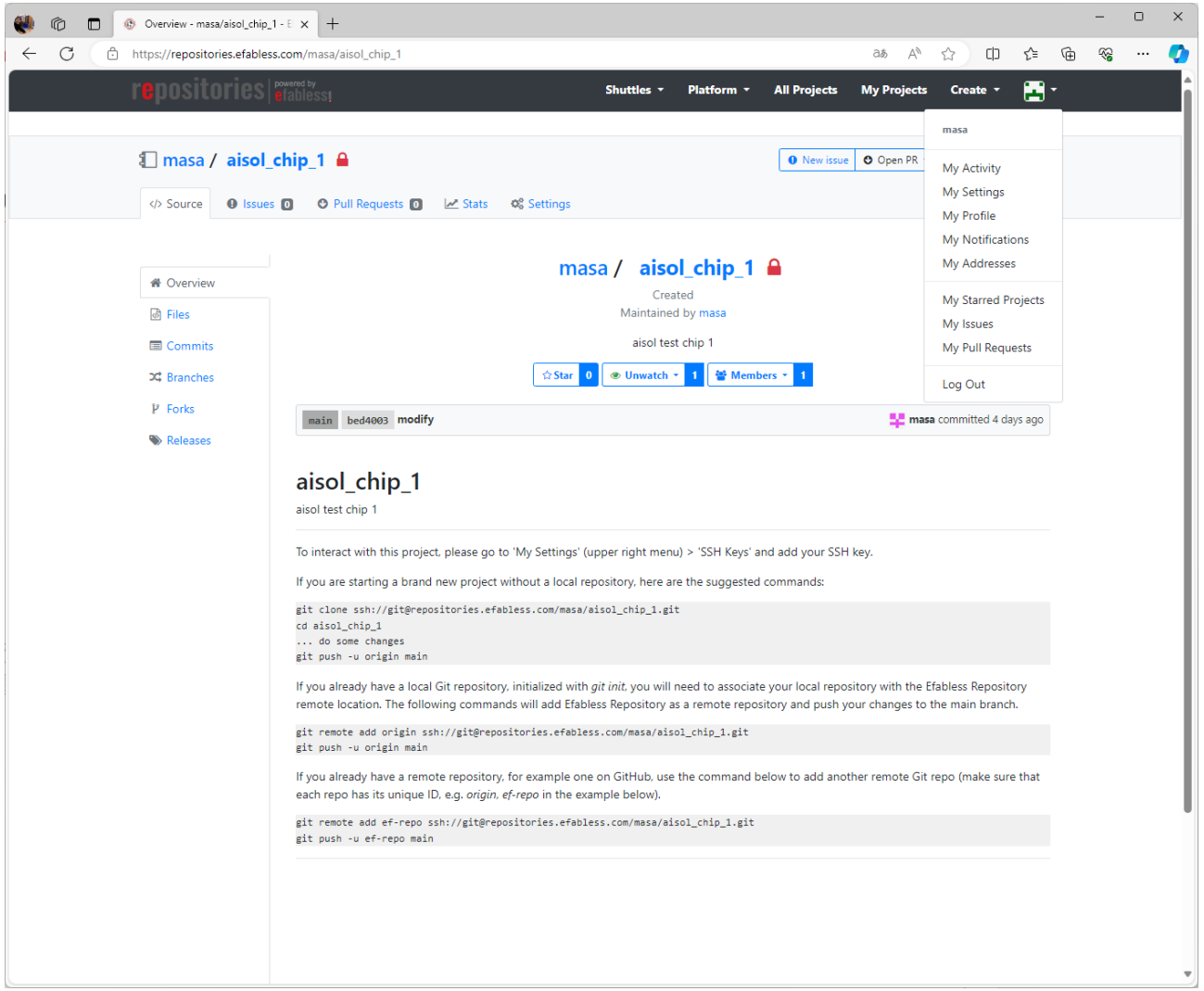
ChipIgniteシャトルにおけるデータサブミットへの道：設定





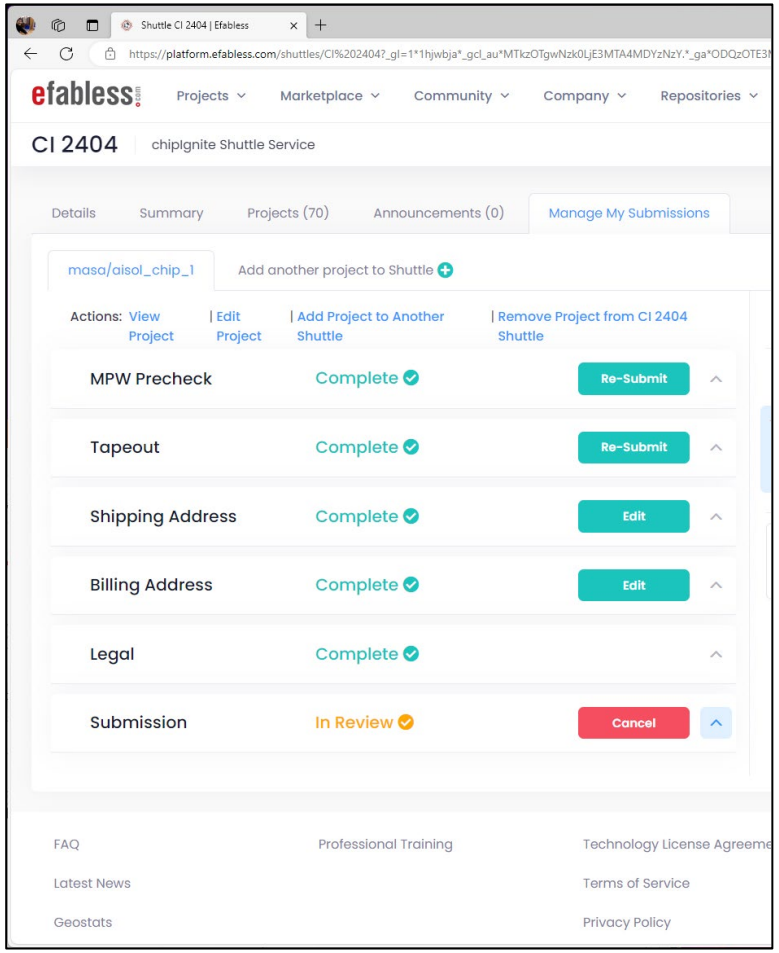
- アカウント作成
  - LinkedInアカウント作成⇒efablessアカウント作成
  - profileとbilling/shippingアドレス記入（ブラウザ右上アイコン）
  - verification実行（ブラウザ右上アイコン）⇒数日で完了
- レポジトリ作成
  - Create repository⇒レポジトリ作成
  - project name, description, privateをチェック、create readmeをチェック、=>create
- SSH key作成
  - SSH key作成と登録（efablessのgit）
- Shuttleタブ⇒シャトルCI2404選択

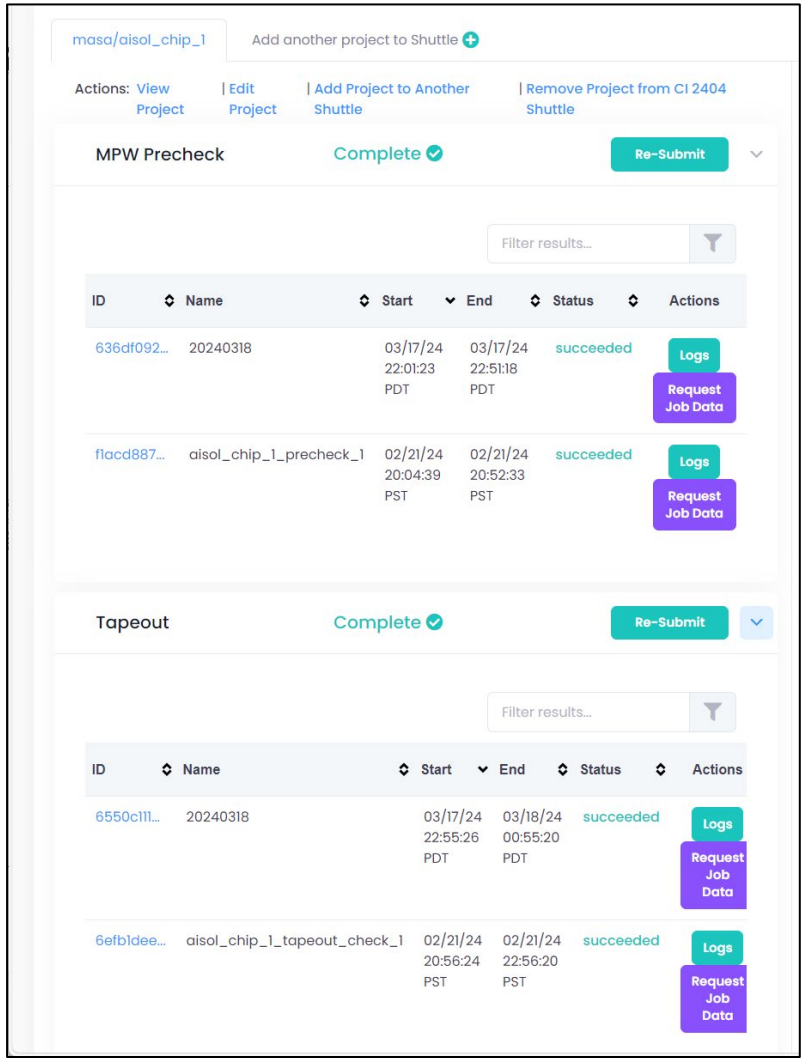
ChipIgniteシャトルにおけるデータサブミットへの道：データアップロード



- My Setting選択（ブラウザ右上アイコン）
- SSH Keysクリック⇒ssh keysページに移動
- ローカルマシンでssh key作成
- ページ右上add ssh keyクリック⇒作成されたid_rsa.pubの中身をボックスにコピペ⇒add。
- 空のレポジトリをローカルマシンにクローン
- クローンしたレポジトリにファイル追加
- git add, git commit, git push

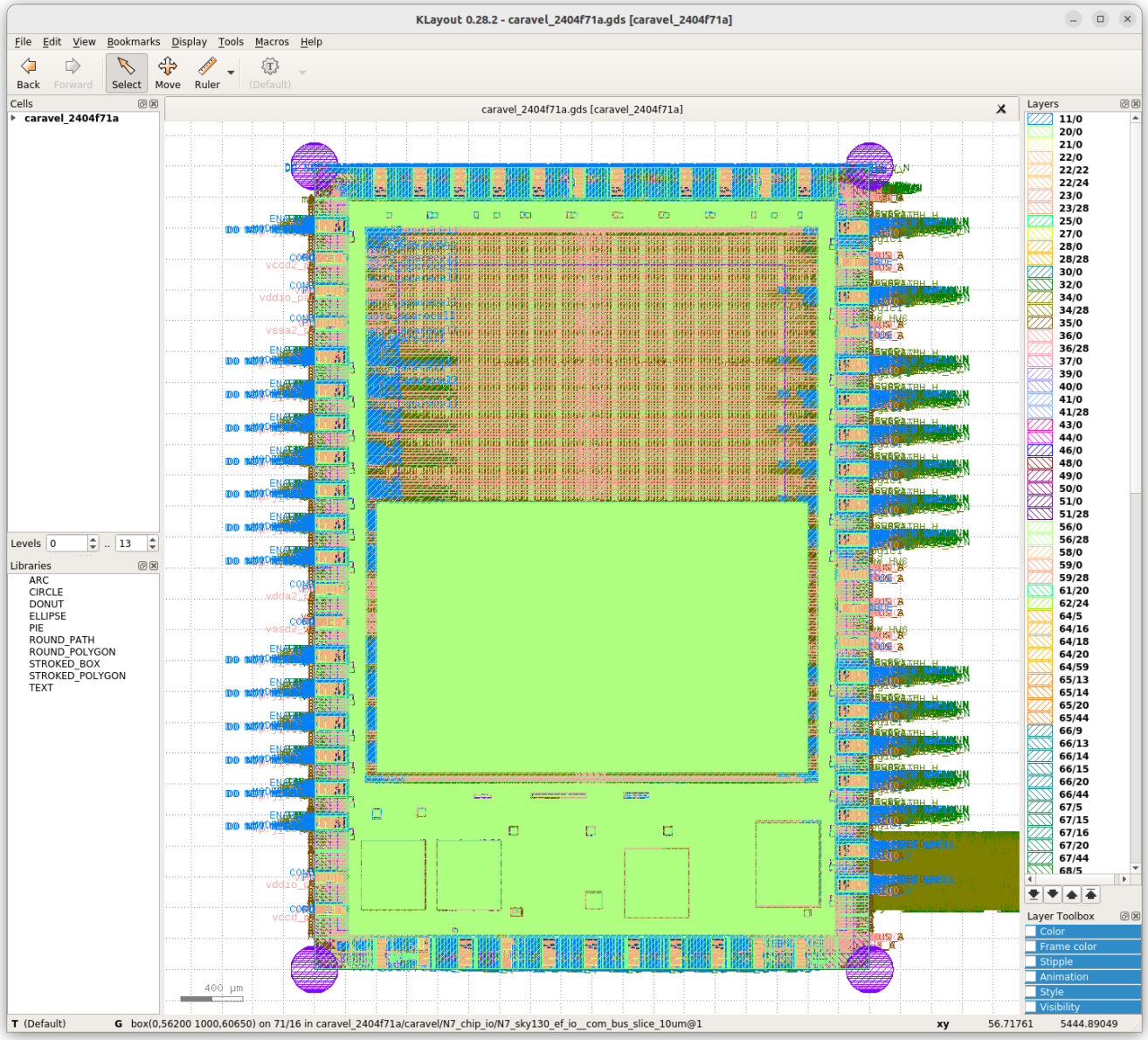
ChipIgniteシャトルにおけるデータサブミットへの道：データサブミッション



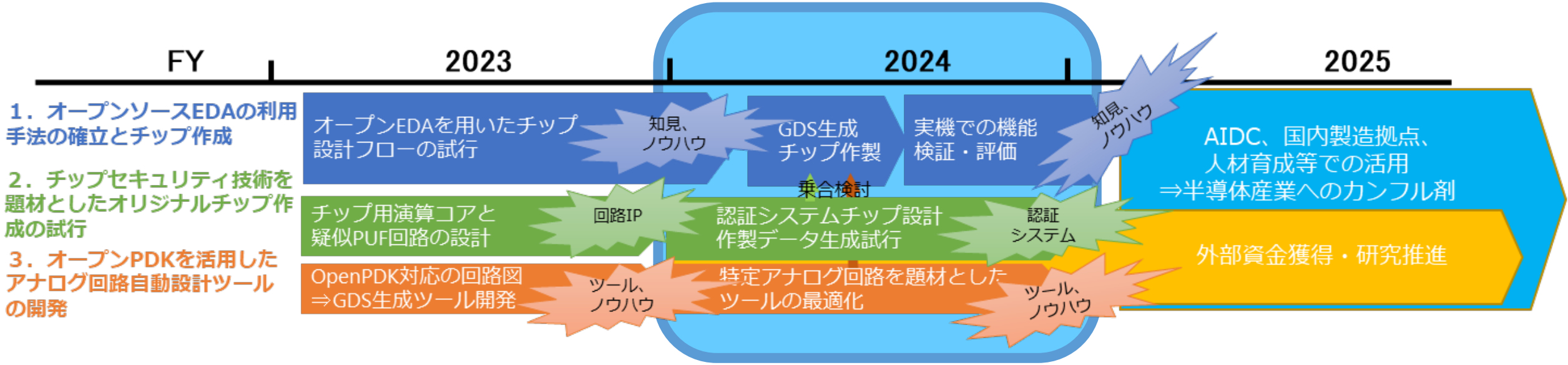


- 事前準備
  - Manage My Submissionsクリック⇒シャトルにレポジトリ追加
    add this project to CI2404 shuttle
- プリチェック
  - ローカルでプリチェック済みデータをプッシュ
  - MPW precheckをサブミット⇒ジョブ名入力⇒ジョブ実行
- テープアウト
  - MPW precheckコンプリートデータ選択⇒ジョブサブミット
- シッピングアドレス、ビリングアドレス選択
- Legal: terms and conditions利用規約とexport compliance輸出管理に同意
- サブミッションで終了
- と思いきや、efablessによるデザインレビューが行われた、、、

ファイナルGDS


KLayout 0.28.2 - caravel_2404f71a.gds [caravel_2404f71a]
File Edit View Bookmarks Display Tools Macros Help
Back Forward Select Move Ruler (Default)
Cells
caravel_2404f71a
Levels 0 .. 13
Libraries
ARC
CIRCLE
DONUT
ELLIPSE
PIE
ROUND_PATH
ROUND_POLYGON
STROKED_BOX
STROKED_POLYGON
TEXT
caravel_2404f71a.gds [caravel_2404f71a]
400 µm
Layers
11/0
20/0
21/0
22/0
22/22
22/24
23/0
23/28
25/0
27/0
28/0
28/28
30/0
32/0
34/0
34/28
35/0
36/0
36/28
37/0
39/0
40/0
41/0
41/28
43/0
44/0
46/0
48/0
49/0
50/0
51/0
51/28
56/0
56/28
58/0
59/0
59/28
61/20
62/24
64/5
64/16
64/18
64/20
64/59
65/13
65/14
65/20
65/44
66/9
66/13
66/14
66/15
66/20
66/44
67/5
67/15
67/16
67/20
67/44
68/5
Layer Toolbox
Color
Frame color
Stipple
Animation
Style
Visibility
T (Default)
G box(0,56200 1000,60650) on 71/16 in caravel_2404f71a/caravel/N7_chip_io/N7_sky130_ef_io__com_bus_slice_10um@1
xy 56.71761 5444.89049

今後の展開



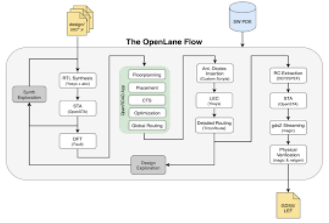
① Caravel統合環境を用いたサンプル回路のテープアウトと実チップでの性能評価
② HWセキュリティチップの個別回路IPの設計完了とそれらの統合、レイアウトデータ生成
③ オペアンプ回路を題材として性能仕様から回路ネットリストを生成するツールの開発
④ 得られた知見の公開
⑤ ChipIgniteシャトルIP試作チップの評価

まとめ

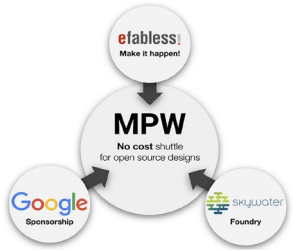
- “誰でも手が届く半導体”に向けたオープンソースEDAの世界的な潮流
- 利用事例がどんどん増え始めてきた
- オープンソースEDAの利用方法等の知見を蓄積・公開
- 国内コミュニティと連携しながらオープンソースEDAの普及促進を図る
- 半導体の活性化や人材育成、PDK開発や設計フロー構築などに裨益することを期待

半導体化のハードルを
下げる取り組み@dapa

Open Source EDA flow



Open Shuttle Program



国内OpenSourceHWの取り組みの
さらなる活性化に助力

期待する効果

- 国内半導体の活性化
- 人材育成

- PDK開発
- 設計フロー構築

以下、参考資料

Ubuntu 22.04 LTS セットアップガイド

動作環境

ホストOS ： Windows11 22H2

使用ソフトウェア ： VMWare workstation pro 17.02, Ubuntu 22.04.4 LTS

（2024/03）VMWare の 17.5.1 は不具合が多く非推奨.

VMWare 仮想マシン作成

仮想マシンをカスタム設定にて作成．仮想マシン形式は 17 とする.

ゲストOSイメージに ubuntu-22.04.4-desktop-amd64.iso を使用し，簡易インストールを行う.

インストール準備完了時に「ハードウェアをカスタマイズ」を開き，USBコントローラ項目にて互換性を2.0 -> 3.1へ変更する．（USB3.1でないとFPGAボードなどの外部機器を接続できない場合がある）．また，bluetoothのチェックボックスを外す.

（VMWare 17 のみ）キーボード・マウスの切断・遅延の不具合解消の設定を行う．プロセッサ項目の仮想化エンジン IOMMUを仮想化 をチェックする．このほか，Windowsのメモリ整合性を無効に設定する.

簡易インストールを選択，インストール途中でキーボード・ロケール設定・ユーザ設定など行う必要がある．なお，簡易インストールによりvmtoolsのインストールや共有フォルダの初期設定は自動的に行われる.

簡易インストール後の設定

アップデートなどパッケージのサーバ変更

デフォルトで日本向けのサーバに設定されている.

(2024/03 現在) Settings → About → Software Update ダイアログ → Ubuntu SoftwareタブのDownload fromでアップデートなどパッケージのサーバを変更できるが，標準の設定を用いること．20.04 と異なり，ミラーサーバの情報では build-essential などパッケージの整合性に問題があり，インストールできない不具合がある.

Windowシステムの設定

Ubuntu 22.04 では Window システムが Wayland であり，X window システムとの互換性により gtk や Xilinx vivado の表示に不具合が生じることがあり，その場合は Window システムを xorg に変更する．/etc/gdm3/custom.conf ファイルの下記の行のコメントアウトを外すことで変更できる.

```
#WaylandEnable=false
```

再起動後，Settings → About にて Windowシステムが X11 に変更されていることを確認できる.

ターミナルの文字の色

ターミナルウィンドウの Preferences から (Profile)Unnamed を選択し，ColorsタブにてPalette/Built-in schemesを選択することで変更できる.

20.04 LTS と同様の配色は Tango を選択することで設定できる.

日本語設定

Settings → Region & Language → Japanese を選択する．また，インストールされている言語の管理を選択し，選択時に現れる追加インストールを行う.

20.04 と比べ，メニューの日本語化やターミナルのフォントの設定が異なる．そこで，日本語入力などのパッケージを導入して近しいものに設定する．（2024/03）下記の中で既にインストールされているパッケージもある.

```
sudo apt install language-pack-ja-base
sudo apt install task-japanese-gnome-desktop
sudo apt install language-pack-gnome-ja-base
sudo apt install fonts-noto-cjk-extra
sudo apt install fonts-takao
```

再起動後，ターミナルの設定 → プロファイル → 文字 にて フォントを指定チェックボックスをチェックし，フォント名をUbuntu Monoとすることで20.04に近いターミナル表示となる.

vmware17_ubuntu22_04_4_setup_2024_03.md

閉じる

追加の日本語フォント

Ubuntuにて利用されている日本語フォントとして，Noto Sans CJK (Google)，Source han code CJK (Adobe)，takao (IPA派生) など挙げられる．見た目を20.04に近しい表示として，ターミナルではUbuntu Mono（Google），Visual Studio CodeではSource hans code CJK (Adobe) を利用することとした．

追加フォントとして，Source han code CJK(Adobe，源の角ゴシック)を導入する．

導入では，font-managerを利用する．

```
sudo apt install font-manager
```

Source han code CJKフォントを導入する．

```
cd Downloads
wget https://github.com/adobe-fonts/source-han-code-jp/releases/download/2.012R/SourceHanCodeJP.ttc
font-manager
```

font-manager上でAdd fonts「+」をクリックし，SourceHanCodeJP.ttcを選択する．

なお，font-managerにてフォントを選択し，Propertiesを参照することでフォントを呼び出す際の名前が確認できる．

共有フォルダ設定

VM をシャットダウンし，VMwareツール上の共有フォルダ設定を行う．再起動後，/mnt/hgfs フォルダ下に共有フォルダが現れない場合は，下記の設定を行う．
 参照：
https://sakimika.hateblo.jp/entry/2022/02/24/153940

マウントしていないが，ツール上認識されているか確認．下記コマンドで共有フォルダが表示される．

```
vmware-hgfsclient
```

マウントポイント作成～共有フォルダマウントまで

```
sudo mkdir /mnt/hgfs
sudo /usr/bin/vmhgfs-fuse .host:/ /mnt/hgfs -o subtype=vmhgfs-fuse,allow_other
```

上記手順により共有フォルダ /mnt/hgfs/(フォルダ名) にアクセスできるようになるが，再起動すると vmhgfs-fuse コマンドを実行し直す必要がある．
 常に共有フォルダをマウントするには，下記の行を /etc/fstab に加える．

```
.host:/ /mnt/hgfs fuse.vmhgfs-fuse allow_other,auto_unmount,defaults 0 0
```

htop, emacs, visual studio code 導入

htop, emacs

```
sudo apt install htop
sudo apt install emacs
```

visual studio code

```
sudo apt install wget gpg
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > packages.microsoft.gpg
sudo install -D -o root -g root -m 644 packages.microsoft.gpg /etc/apt/keyrings/packages.microsoft.gpg
sudo sh -c 'echo "deb [arch=amd64,arm64,armhf signed-by=/etc/apt/keyrings/packages.microsoft.gpg] https://packages.microsoft.com/repos/code stable main" > /etc/apt/sources.list.d/vscode.list'
rm -f packages.microsoft.gpg
sudo apt install apt-transport-https
sudo apt update
sudo apt install code
```

その他設定など

gui復帰

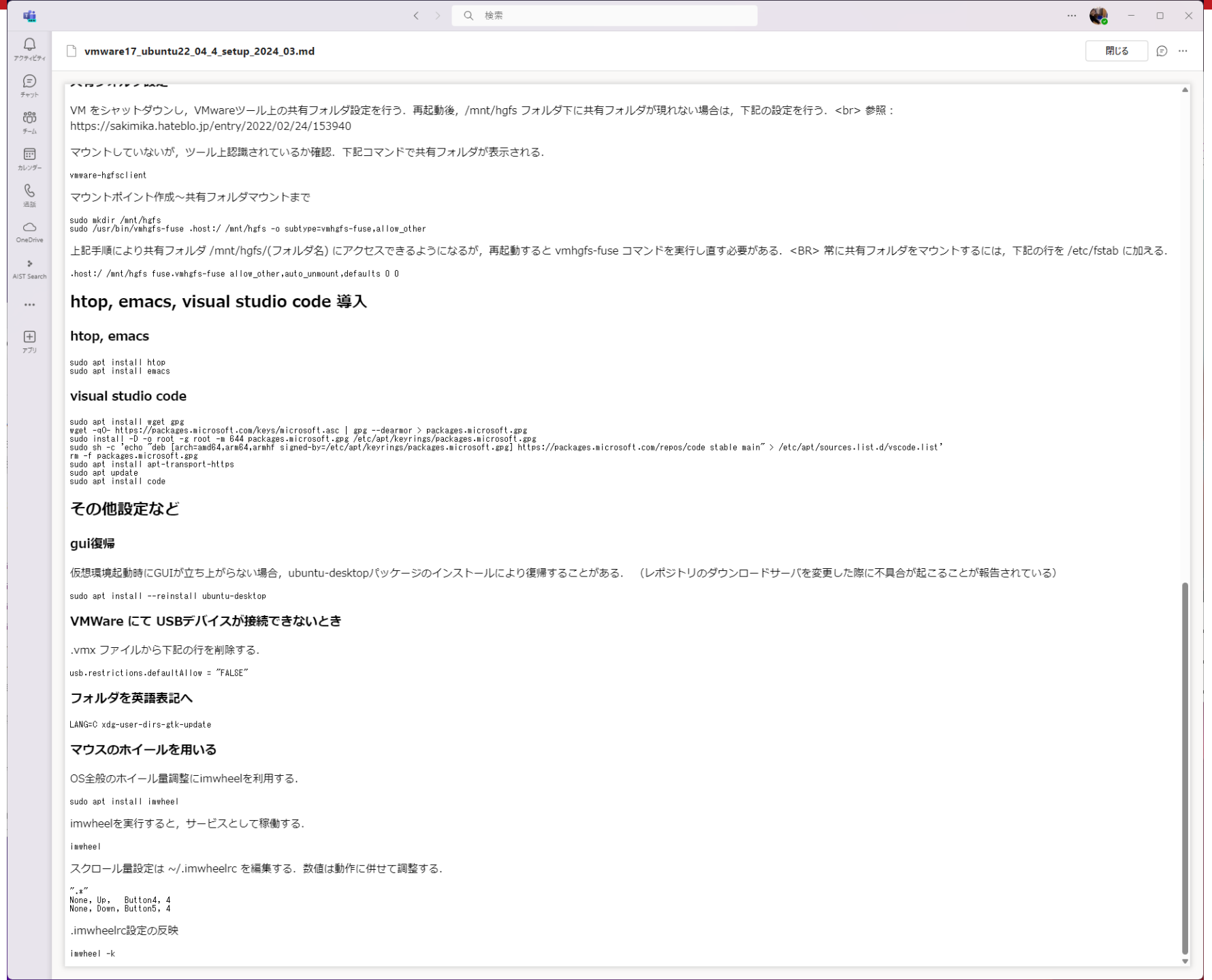
仮想環境起動時にGUIが立ち上がらない場合，ubuntu-desktopパッケージのインストールにより復帰することがある．（レポジトリのダウンロードサーバを変更した際に不具合が起こることが報告されている）

```
sudo apt install --reinstall ubuntu-desktop
```

VMWare にて USBデバイスが接続できないとき

.vmx ファイルから下記の行を削除する．

```
usb.restrictions.defaultAllow = "FALSE"
```



vmware17_ubuntu22_04_4_setup_2024_03.md

閉じる

VM をシャットダウンし，VMwareツール上の共有フォルダ設定を行う．再起動後，/mnt/hgfs フォルダ下に共有フォルダが現れない場合は，下記の設定を行う．
 参照：
https://sakimika.hateblo.jp/entry/2022/02/24/153940

マウントしていないが，ツール上認識されているか確認．下記コマンドで共有フォルダが表示される．

```
vmware-hgfsclient
```

マウントポイント作成～共有フォルダマウントまで

```
sudo mkdir /mnt/hgfs
sudo /usr/bin/vmhgfs-fuse .host:/ /mnt/hgfs -o subtype=vmhgfs-fuse,allow_other
```

上記手順により共有フォルダ /mnt/hgfs/(フォルダ名) にアクセスできるようになるが，再起動すると vmhgfs-fuse コマンドを実行し直す必要がある．
 常に共有フォルダをマウントするには，下記の行を /etc/fstab に加える．

```
.host:/ /mnt/hgfs fuse.vmhgfs-fuse allow_other,auto_unmount,defaults 0 0
```

htop, emacs, visual studio code 導入

htop, emacs

```
sudo apt install htop
sudo apt install emacs
```

visual studio code

```
sudo apt install wget gpg
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > packages.microsoft.gpg
sudo install -D -o root -g root -m 644 packages.microsoft.gpg /etc/apt/keyrings/packages.microsoft.gpg
sudo sh -c 'echo "deb [arch=amd64,arm64,armhf signed-by=/etc/apt/keyrings/packages.microsoft.gpg] https://packages.microsoft.com/repos/code stable main" > /etc/apt/sources.list.d/vscode.list'
rm -f packages.microsoft.gpg
sudo apt install apt-transport-https
sudo apt update
sudo apt install code
```

その他設定など

gui復帰

仮想環境起動時にGUIが立ち上がらない場合，ubuntu-desktopパッケージのインストールにより復帰することがある． （レポジトリのダウンロードサーバを変更した際に不具合が起こることが報告されている）

```
sudo apt install --reinstall ubuntu-desktop
```

VMWare にて USBデバイスが接続できないとき

.vmx ファイルから下記の行を削除する．

```
usb.restrictions.defaultAllow = "FALSE"
```

フォルダを英語表記へ

```
LANG=C xdg-user-dirs-gtk-update
```

マウスのホイールを用いる

OS全般のホイール量調整にimwheelを利用する．

```
sudo apt install imwheel
```

imwheelを実行すると，サービスとして稼働する．

```
imwheel
```

スクロール量設定は ~/.imwheelrc を編集する．数値は動作に併せて調整する．

```
".*"
None, Up,   Button4, 4
None, Down, Button5, 4
```

.imwheelrc設定の反映

```
imwheel -k
```

OpenLane セットアップガイド（2024年1月版）

（2024/01版）efabless 社 chipignite サービスを利用したチップ作製を目指す．

動作環境

動作環境 ： Windows11 22H2, VMWare workstation pro 17.0.2, Ubuntu 22.04.3 LTS

使用ソフトウェア ： OpenLane 2024.01.12

２０２４年１月現在，OpenLaneは頻繁に更新されており，導入手順がそのまま利用できない場合も予想される．

必要パッケージの導入

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install build-essential python3 python3-venv python3-pip make git
python3 -m pip install pyyaml click
```

Docker-ce, community edition の導入

docker は 無償の商用利用可能な CLI を導入する． （docker desktopは有償）

２０２４年１月現在，無償のDockerエンジンのみの導入方法は英語ドキュメントのみであり，そちらを参照する．

公式情報：https://docs.docker.com/engine/install/ubuntu/

まず，Dockerの公開鍵を設定する．ca-certificates 導入時にエラーなど発生する場合は，sudo apt autoremove を実行する．

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

つぎに，Dockerのレポジトリを設定する．

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

エンジンのインストールを行う．

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

dockerの動作確認を行う．

```
sudo docker run hello-world
```

ローカル計算機でdockerをユーザ権限により動作させる設定

グループ docker にユーザを登録してユーザ権限で実行できるように設定する．なお，hello-world 実行時に設定が反映されない場合は再起動する．

```
sudo groupadd docker
sudo usermod -aG docker ${USER}
docker run hello-world
```

参照： https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04

dockerの動作確認を行う．

```
sudo docker run hello-world
```

ローカル計算機でdockerをユーザ権限により動作させる設定

グループ docker にユーザを登録してユーザ権限で実行できるように設定する．なお，hello-world 実行時に設定が反映されない場合は再起動する．

```
sudo groupadd docker
sudo usermod -aG docker ${USER}
docker run hello-world
```

参照：　https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04

Docker導入のトラブルシューティング

パッケージ元の追加を実行すると，apt 実行時に「複数回設定されています」など表示されインストールなどできない状態となる．

これは，/etc/apt/sources.list.d/ 下のパッケージ記述が重複しているため．grep /etc/apt/sources.list.d/* docker など実行した際に複数表示される場合は，当該の *.list からdocker記載を「#」でコメントアウトすることで回避できる．

Docker導入のトラブルシューティング２

日本語翻訳のDockerバージョン19など古いインストール方法では，GPGの公開鍵が無効である不具合が生じることがある．この場合，下記の手順によりdockerエンジン，レポジトリの削除により回避する．

```
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt remove docker-ce
sudo apt autoremove

sudo rm /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-jammy.list
sudo rm /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-jammy.list.save
```

OpenLane 本体（dockerコンテナ）

サイズが大きいため，gitではdepthにて最新のコミットを取得する様に指定する．

```
git clone --depth 1 https://github.com/The-OpenROAD-Project/OpenLane.git
cd OpenLane
make
```

動作テスト．designs/spm/runs/openlane_test_results 下にテスト回路がビルドされる．

```
make test
```

VMのDISPLAY設定変更

OpenLaneのコンテナ起動スクリプトmake mountでは，hostネットワークが利用される．このとき，XwindowのGUIを接続することができない場合がある．

（恐らくWindows上のVMwareの場合，Microsoft Defenderにてブロックされている）

このため，Makefile の34行DOCKER_OPTIONSにて，--network hostを--network bridgeに変更する．

```
DOCKER_OPTIONS += -e DISPLAY=$(DISPLAY) -v /tmp/.X11-unix:/tmp/.X11-unix -v $(HOME)/.Xauthority:/root/.Xauthority --network bridge --security-opt seccomp=unconfined
```

最後に，make test にて合成した回路を表示させ，正しく表示されることを確認する．

```
make mount
klayout -e -nn $PDK_ROOT/sky130A/libs.tech/klayout/tech/sky130A.lyt ¥
   -l $PDK_ROOT/sky130A/libs.tech/klayout/tech/sky130A.lyp ¥
   ./designs/spm/runs/openlane_test/results/final/gds/spm.gds
```

検索

Caravel_trial_1st_mpw-9g_2024_03.md

閉じる

Chipignite チップ作製に向けた，Caravel の試用（mpw-9g）

（2024/03版）efabless 社 chipignite サービスを利用したチップ作製を目指す．

チップ作製データの生成では，あらかじめ用意されたクロック・リセット生成やIOパッド・制御回路を統合した Caravel ハーネスを利用しすることでデジタル回路の設計のみ注力できる手法を用いる．

Caravel ハーネスは efabless 社から提供されるが，chipignite 用のものはefablessへの登録が必要となる．チップ作製前に手順の確認や手法に慣れるため，オープンソースの OpenMPW プロジェクト用の Caravel を試用する．

Caravelのサンプルプロジェクトではlitexを元とした回路が初期設定されているが，2024/03現在，ゲートレベルシミュレーションが動作しない不具合がある．

※参考資料

efabless社：　https://efabless.com

ユーザ向け公開Caravelハーネス：　https://github.com/efabless/caravel_user_project

日本有志による分かり易い情報：　https://vlsi.jp/OpenMPW.html

動作環境

動作環境 ： Windows11 22H2, VMWare workstation pro 17.0.2, Ubuntu 22.04.4 LTS

シミュレーションでは大きくメモリを要する点に注意．（サンプルで 10GB~ 程度）

Caravel と Caravel 向け OpenLane・PDK のセットアップ

必要パッケージの導入

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install build-essential python3 python3-venv python3-pip
sudo apt-get install make git
python3 -m pip install pyyaml click
```

Docker-ce, community edition の導入

docker は 無償の商用利用可能な CLI を用いる．（docker desktopは有償）

下記は公式の手順を掲載している．

公式：　https://docs.docker.com/engine/install/ubuntu/

まず，必要なパッケージと公開鍵を設定する．

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

安定版のリポジトリを設定し，docker をインストールする．

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get install docker-ce
```

docker の動作確認を行う．

```
sudo docker run hello-world
```

ローカル計算機でdockerをユーザ権限により動作させる設定

グループ docker にユーザを登録してユーザ権限で実行できるように設定する．

```
sudo groupadd docker
sudo usermod -aG docker ${USER}
```

再起動（ログアウトでも可）して設定を反映させ，ユーザ権限での動作を確認する

再起動（ログアウトでも可）して設定を反映させ，ユーザ権限での動作を確認する．

```
docker run hello-world
```

参照：　https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04

Caravel インストールフォルダの構成

本手順では，下記ディレクトリ構成にてインストールを行う．

```
$HOME/
|-- Caravel_mpw-9g/
    |-- caravel_mpw-9g/
    |-- dependencies/
    |   |-- openlane_src/
    |   |-- pdks/
    |-- mpw-9g_precheck/
    |setenv.sh
```

Caravel (mpw-9g) ディレクトリを作成する．

```
mkdir Caravel_mpw-9g
cd Caravel_mpw-9g
```

Caravel ハーネスのパッケージ導入とセットアップ

git レポジトリから mpw-9g タグのものをクローンする．

```
git clone -b mpw-9g https://github.com/efabless/caravel_user_project.git caravel_mpw-9g
```

EDAの環境変数設定のファイルを作成する．（setenv.shというファイル名とする）

```
export OPENLANE_ROOT=$HOME/Caravel_mpw-9g/dependencies/openlane_src
export PDK_ROOT=$HOME/Caravel_mpw-9g/dependencies/pdks
export PDK=sky130A
export CARAVEL_ROOTT=$HOME/Caravel_mpw-9f/caravel_mpw-9g/caravel
```

<details><summary>コマンドラインからファイルを生成する方法</summary>

```
touch setenv.sh
echo "export OPENLANE_ROOT=¥$HOME/Caravel_mpw-9g/dependencies/openlane_src" | tee -a setenv.sh
echo "export PDK_ROOT=¥$HOME/Caravel_mpw-9g/dependencies/pdks" | tee -a setenv.sh
echo "export PDK=sky130A" | tee -a setenv.sh
echo "export CARAVEL_ROOTT=¥$HOME/Caravel_mpw-9f/caravel_mpw-9g/caravel" | tee -a setenv.sh
```

</details>

 次に，環境変数の設定，dependencies ディレクトリを作成する．

```
source setenv.sh
mkdir dependencies
cd caravel_mpw-9g
```

mpw_precheck のインストールフォルダを変更するため Makefile の PRECHECK 行を修正する．

```
# PRECHECK_ROOT?=${HOME}/mpw_precheck
PRECHECK_ROOT?=${HOME}/Caravel_mpw-9g/mpw-9g_precheck
```

セットアップ実行する．

```
make setup
```

Caravel環境によるビルド

レポジトリに登録されているサンプル回路をビルドし，ユーザ回路と Caravel を含めた一連のビルド手順を確認する．なお，ユーザ回路/Caravelを含んだ回路のビルドに１～２時間程度要する．

まず，サンプル回路では GPIO の電源投入時の設定の記述 (verilog/rtl/user_define.v) の修正を行う．本手順では，全てのピンについて 初期値のGPIO_MODE_INVALID を GPIO_MODE_MGMT_STD_OUTPUT に修正する．

```
`define USER_CONFIG_GPIO_5_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_6_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_7_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_8_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_9_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_10_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_11_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_12_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_13_INIT `GPIO_MODE_MGMT_STD_OUTPUT

// Configurations of GPIO 14 to 24 are used on caravel but not caravan.
```

Caravel_trial_1st_mpw-9g_2024_03.md

閉じる

レポジトリに登録されているサンプル回路をビルドし，ユーザ回路と Caravel を含めた一連のビルド手順を確認する．なお，ユーザ回路/Caravelを含んだ回路のビルドに１～２時間程度要する．

まず，サンプル回路では GPIO の電源投入時の設定の記述 (verilog/rtl/user_define.v) の修正を行う．本手順では，全てのピンについて 初期値のGPIO_MODE_INVALID を GPIO_MODE_MGMT_STD_OUTPUT に修正する．

```
`define USER_CONFIG_GPIO_5_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_6_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_7_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_8_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_9_INIT  `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_10_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_11_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_12_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_13_INIT `GPIO_MODE_MGMT_STD_OUTPUT

// Configurations of GPIO 14 to 24 are used on caravel but not caravan.
`define USER_CONFIG_GPIO_14_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_15_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_16_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_17_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_18_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_19_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_20_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_21_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_22_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_23_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_24_INIT `GPIO_MODE_MGMT_STD_OUTPUT

`define USER_CONFIG_GPIO_25_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_26_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_27_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_28_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_29_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_30_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_31_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_32_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_33_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_34_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_35_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_36_INIT `GPIO_MODE_MGMT_STD_OUTPUT
`define USER_CONFIG_GPIO_37_INIT `GPIO_MODE_MGMT_STD_OUTPUT
```

書き換え後，Caravel_mpw-9g/caravel_mpw-9g ディレクトリに移動し，ビルドを実行する．

```
make user_proj_example
make user_project_wrapper
```

エラーなど表示されずビルドが成功すると，gdsフォルダ下に user_progect_wrapper.gds ファイルが生成される．

デザインのローカルチェック

デザインチェックではチップ製造の工程との整合性が確認される．例えば，デザインの README.md の内容が更新についてもチェックされる．ローカルチェックに適合するよう，README.mdを下記の内容に書き換える．なお，元の文章を残した場合やREADME.bak などとしてオリジナルのファイルを残した場合，チェックでエラーが報告されるため注意．

```
This project is minimally modified to mpw-8c caravel template.
The function is 32-bit counter. All of 38-bit IOs are set to OUTPUT.
```

デザインチェックの環境セットアップ～実行を行う．run-precheck は１時間弱程度かかる．

```
make precheck
make run-precheck
```

デザインチェックでエラーが報告されないことを確認する．以上の手順によりサンプルユーザデザインのビルド～Caravelを含めたデザインのビルド～出力チェックまでできることを確認できる．

Caravelのファイル構造

・gds/：GDSIIファイルが生成される
・openlane/：OpenLANEの設定ファイル
・verilog/dv/：シミュレーションとテストベンチ用ファイル
・verilog/rtl/：ユーザデザインのverilogファイル

全ディレクトリの説明は下記を参照：https://caravel-harness.readthedocs.io/en/latest/getting-started.html#required-directory-structure

Docker導入のトラブルシューティング

パッケージ元の追加を実行すると，apt 実行時に「複数回設定されています」など表示されインストールなどできない状態となる．

これは，/etc/apt/sources.list.d/ 下のパッケージ記述が重複していることが原因と考えられる．grep /etc/apt/sources.list.d/* docker と確認した際に複数の行が表示される場合はパッケージ記述が重複しており，該当する *.list からdocker記載を「#」でコメントアウトすることで回避できる．

Chipignite チップ作製に向けた，Caravel を用いた回路のシミュレーション（mpw-9g）

（2023/12版）efabless 社 chipignite サービスを利用したチップ作製を目指す．

チップ作製データの生成では，あらかじめ用意されたクロック・リセット生成やIOパッド・制御回路を統合した Caravel ハーネスを利用しすることでデジタル回路の設計のみ注力できる手法を用いる．

本ドキュメントでは，「Caravel 試用 1st (mpw-9g)」のセットアップ後に，Caravelを用いたシミュレーションの手順を示す．

※参考資料

efabless社：　https://efabless.com

ユーザ向け公開Caravelハーネス：　https://github.com/efabless/caravel_user_project

日本有志による分かり易い情報：　https://vlsi.jp/OpenMPW.html

動作環境

動作環境 ： Windows11 22H2, VMWare workstation pro 17.0.2, Ubuntu 22.04.3 LTS

シミュレーションでは大きくメモリを要する点に注意．（サンプルで 10GB~ 程度）

Caravel フォルダの構成

```
$HOME/
|-- Caravel_mpw-9g/
    |-- caravel_mpw-9g/
    |-- dependencies/
    |   |-- openlane_src/
    |   |-- pdks/
    |-- mpw-9g_precheck/
    |setenv.sh
```

Caravel 環境の設定

環境変数を設定する．

```
cd Caravel_mpw-9g/caravel_mpw-9g
source ../setenv.sh
```

シミュレーション環境の設定

シミュレーション用dockerコンテナ，波形表示ソフトウェアを導入する．

```
make simenv
sudo apt-get install gtkwave
```

Caravel (mpw-9g) シミュレーション種類について

Caravel のサンプル回路では，GPIO・wishbone/バス・logic analyzer それぞれを対象とし，rtl・ゲートレベル・遅延情報付きゲートレベルの方式でシミュレーションできる．

シミュレーション対象・方式の選択は，make オプションにて verify-(シミュレーション対象)-(方式) として指定する．

GPIOのRTLシミュレーション

ユーザ回路（カウンタ）の値がユーザ用IOに反映されるか確認を行う．

```
make verify-io_ports-rtl
```

下記のように，IOにカウンタ値が反映されている様子をシミュレーションにて確認できる．

```
vvp  io_ports.vvp
Reading io_ports.hex
 io_ports.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
FST info: dumpfile io_ports.vcd opened for output.
```

```
cd Caravel_mpw-9g/caravel_mpw-9g
source ../setenv.sh
```

シミュレーション環境の設定

シミュレーション用dockerコンテナ，波形表示ソフトウェアを導入する．

```
make simenv
sudo apt-get install gtkwave
```

Caravel (mpw-9g) シミュレーション種類について

Caravel のサンプル回路では，GPIO・wishbone/バス・logic analyzer それぞれを対象とし，rtl・ゲートレベル・遅延情報付きゲートレベルの方式でシミュレーションできる．

シミュレーション対象・方式の選択は，make オプションにて verify-(シミュレーション対象)-(方式) として指定する．

GPIOのRTLシミュレーション

ユーザ回路（カウンタ）の値がユーザ用IOに反映されるか確認を行う．

```
make verify-io_ports-rtl
```

下記のように，IOにカウンタ値が反映されている様子をシミュレーションにて確認できる．

```
vvp  io_ports.vvp
Reading io_ports.hex
io_ports.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
FST info: dumpfile io_ports.vcd opened for output.
MPRJ-IO state = zzzzlzzz
MPRJ-IO state = zzzzlzz0
MPRJ-IO state = zzzzlz00
MPRJ-IO state = 01001111
MPRJ-IO state = 01010000
MPRJ-IO state = 01010001
MPRJ-IO state = 01010010
MPRJ-IO state = 01010011
...
MPRJ-IO state = 11111011
MPRJ-IO state = 11111100
MPRJ-IO state = 11111101
MPRJ-IO state = 11111110
MPRJ-IO state = 11111111
Monitor: Test 1 Mega-Project IO (RTL) Passed
io_ports_tb.v:182: $finish called at 360462500 (1ps)
mv io_ports.vcd RTL-io_ports.vcd
rm io_ports.vvp
```

動作波形は，verilog/dv/io_ports/RTL-io_ports.vcd に保存される．

gtkwaveにより動作波形を確認すると，およそ 350us 後にmprj_ioへカウンタの値が反映されている様子が確認できる．

```
gtkwave verilog/dv/io_ports/RTL-io_ports.vcd
```

画面のSST中のio_ports_tbを選択し，Type/Signals に表示される信号 clock, mprj_io など選択してAppendボタンを押し，表示するSignalsへ登録する．

Caravelでは制御用RISC-VコアによりIOなど設定がなされるが，IO設定まで350us程度要すると見られる．

その他のシミュレーションについて

ゲートレベルシミュレーションは2023/12現在のmpw-9gでは実行できない．参考として，シミュレーションを行う手順を下記に示すが，Caravelのハーネス種類を変更することから非推奨．（また，mpw-9cではシミュレーション実行が可能であった経緯から，修正・更新により今後解消されると推測される）

```
・Makefile中のCARAVEL_LITE?=1を0に設定し，make setup からやり直す（caravelや関連ツールの入れ直し）
・make user_project_wrapper 実行によりGDSファイルが生成されることを確認
・make precheck によりチェックを確認
・make verify-io_ports-rtl により動作確認
（以上でcaravelを入れ替えても同様にビルドできることを確認，以下，ゲートレベルシミュレーションの手順）
・環境変数 CARAVEL_ROOT を設定．export CARAVEL_ROOT=$HOME/Caravel_mpw-9f/caravel_mpw-9f/caravel
・シミュレーション用ファイルの生成 caravel/script/gen_gpio_defaults.py
・verilog/gl 以下に caravel_core.v や gpio_devaults_block_****.v が生成されてることを確認
・verilog/includes/includes.gl.caravel_user_project を編集
 -v $(USER_PROJECT_VERILOG)/gl/caravel_core.v
 のほか，gpio_defaults_block_****.v を追加
  （例）
  -v $(USER_PROJECT_VERILOG)/gl/gpio_defaults_block_0403.v
  -v $(USER_PROJECT_VERILOG)/gl/gpio_defaults_block_0801.v
  -v $(USER_PROJECT_VERILOG)/gl/gpio_defaults_block_1803.v
  -v $(USER_PROJECT_VERILOG)/gl/gpio_defaults_block_1809.v
・make verify_io_ports-gl を実行
```

Chipignite チップ作製に向けた，Caravel のサンプル回路へ自作回路の付加 その１（mpw-9g）

（2023/12版）efabless 社 chipignite サービスを利用したチップ作製を目指す．

チップ作製データの生成では，あらかじめ用意されたクロック・リセット生成やIOパッド・制御回路を統合した Caravel ハーネスを利用しすることでデジタル回路の設計のみ注力できる手法を用いる．

本ドキュメントでは，「Caravel 試用 1st (mpw-9g)」のセットアップ後に，Caravelのサンプル回路に自作回路を追加する．

※参考資料

efabless社：　https://efabless.com

ユーザ向け公開Caravelハーネス：　https://github.com/efabless/caravel_user_project

ユーザ向け公開のチップテストボード：　https://github.com/efabless/caravel_board

日本有志による分かり易い情報：　https://vlsi.jp/OpenMPW.html

動作環境

動作環境 ： Windows11 22H2, VMWare workstation pro 17.0.2, Ubuntu 22.04.3 LTS

シミュレーションでは大きくメモリを要する点に注意．（サンプルで 10GB~ 程度）

Caravel フォルダの構成

```
$HOME/
|-- Caravel_mpw-9g/
    |-- caravel_mpw-9g/
    |-- dependencies/
    |   |-- openlane_src/
    |   |-- pdks/
    |-- mpw-9g_precheck/
    |setenv.sh
```

Caravel 環境の設定

環境変数を設定する．

```
cd Caravel_mpw-9g/caravel_mpw-9g
source ../setenv.sh
```

追加する回路について

Caravelのサンプル回路は 16-bit カウンタであり，GPIOへの出力やWishboneバス・内部ロジックアナライザに接続された構成となっている．サンプル回路に一般的なテーブル型のAES SBOX回路を接続し，カウンタの下位8-bitがSBOXに入力される回路を構成する．[1]

本ドキュメントの回路では，2023年12月時点のefabless社chipigniteでのチップ製造時にサービスされる動作確認用ボードに合わせてGPIOのピンを割り当てている．具体的には，GPIOの1~4ピットがボードからのチップ制御，5,6ピットがUARTに割り当てられており，回路では8~15ピットにカウンタ値，16~23ピットにSBOX出力を割り当てた．

サンプルプロジェクトの修正

verilog/rtl 下のソースコード，verilog/dv/io_ports 下のシミュレーションファイル，openlane/user_proj_example 下のビルド設定の３か所を下記のとおり修正する．

１）verilog/rtl 下のソースコード：AES_SBOX_ENC.vを追加し，user_proj_example.v，user_project_wrapper.v，user_defines.vを修正する．修正部分が多いため，ドキュメントに付随したコードを参照．

２）verilog/dv/io_ports/ 下のシミュレーションファイル：テストベクタ io_ports_tv.b とRISC-V 用コード io_ports.c を修正する．また，verilog/includes 下の includes.rtl.caravel_user_project に追加ファイルについて記載する．修正部分はドキュメントに付随したコードを参照．

３）openlane/user_proj_example/ 下のビルド設定 config.json に，追加ファイルについて記載する．

```
   "VERILOG_FILES": [
      "dir::../../verilog/rtl/defines.v",
      "dir::../../verilog/rtl/user_proj_example.v",
```

本ドキュメントの回路では，2023年12月時点のefabless社chipigniteでのチップ製造時にサービスされる動作確認用ボードに合わせてGPIOのピンを割り当てている．具体的には，GPIOの1~4ビットがボードからのチップ制御，5,6ビットがUARTに割り当てられており，回路では8~15ビットにカウンタ値，16~23ビットにSBOX出力を割り当てた．

サンプルプロジェクトの修正

verilog/rtl 下のソースコード，verilog/dv/io_ports 下のシミュレーションファイル，openlane/user_proj_example 下のビルド設定の３か所を下記のとおり修正する．

1）verilog/rtl 下のソースコード：AES_SBOX_ENC.vを追加し，user_proj_example.v，user_project_wrapper.v，user_defines.vを修正する．修正部分が多いため，ドキュメントに付随したコードを参照．

2）verilog/dv/io_ports/ 下のシミュレーションファイル：テストベクタ io_ports_tv.b とRISC-V 用コード io_ports.c を修正する．また，verilog/includes 下の includes.rtl.caravel_user_project に追加ファイルについて記載する．修正部分はドキュメントに付随したコードを参照．

3）openlane/user_proj_example/ 下のビルド設定 config.json に，追加ファイルについて記載する．

```
   "VERILOG_FILES": [
       "dir::../../verilog/rtl/defines.v",
       "dir::../../verilog/rtl/user_proj_example.v",
       "dir::../../verilog/rtl/AES_SBOX_ENC.v"
   ]
```

GPIOのRTLシミュレーション

RTLシミュレーションを行い，上記修正が反映されているか確認を行う．

```
make verify-io_ports-rtl
```

下記のように，SBOX の入出力信号が反映されている様子を確認でれる．（例）入力が 0xA0 のとき出力は 0xE0 となる．入出力信号は指定した8~23ビット目に正しく出力されていることがわかる．

（補足）3ビット目はチップ制御用SPIのCSB信号であり，efabless社のボードではFTDIチップのIOに接続されている．

```
vvp  io_ports.vvp
Reading io_ports.hex
io_ports.hex loaded into memory
Memory 5 bytes = 0x6f 0x00 0x00 0x0b 0x13
FST info: dumpfile io_ports.vcd opened for output.
MPRJ-IO state = zz zz : zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz1zzz
MPRJ-IO state = db 9f : zzzzzzzzzzzzz1001111111011011zzzz1zzz
MPRJ-IO state = e0 a0 : zzzzzzzzzzzzz1010000011100000zzzz1zzz
MPRJ-IO state = 32 a1 : zzzzzzzzzzzzz1010000100110010zzzz1zzz
...
MPRJ-IO state = b0 fc : zzzzzzzzzzzzz1111110010110000zzzz1zzz
MPRJ-IO state = 54 fd : zzzzzzzzzzzzz1111110101010100zzzz1zzz
MPRJ-IO state = bb fe : zzzzzzzzzzzzz1111111010111011zzzz1zzz
MPRJ-IO state = 18 ff : zzzzzzzzzzzzz1111111000101100zzzz1zzz
Monitor: Test 1 Mega-Project IO (RTL) Passed
io_ports_tb.v:208: $finish called at 494862500 (1ps)
mv io_ports.vcd RTL-io_ports.vcd
rm io_ports.vvp
```

動作波形は，verilog/dv/io_ports/RTL-io_ports.vcd に保存される．

gtkwaveにより動作波形を確認すると，およそ 350us 後にmprj_ioへ値が反映されている様子が確認できる．

```
gtkwave verilog/dv/io_ports/RTL-io_ports.vcd
```

ビルドと回路検証

caravel_mpw-9g ディレクトリにてビルドを実行する．

```
make user_proj_example
make user_project_wrapper
```

ローカルでのデザインチェックを実行する．

```
make run-precheck
```

まとめ

以上の手順にて，サンプルユーザデザインに簡単な回路の追加を行い，そのシミュレーション~ビルド~Caravelを含めたデザインのビルドまで実施した．

---

1. なお，サンプル回路ではWishboneバス・内部ロジックアナライザの読み書きの値の検証は難しい構成である．これは，RISC-V から Wishboneバス・内部ロジックアナライザを通じて値を書き込んだ後にカウンタがフリーランし続けるため．[-^1] ↩

検索

ubuntu22_xls_1st_setup_2024_02.md

閉じる

Google XLS 高位合成ツール セットアップガイド（2024年2月版）

（2024/02版）Google 社 XLS 高位合成ツールを用いてCコードからHDL生成を目指す.

動作環境

動作環境 ： Windows11 22H2, VMWare workstation pro 17.0.2, Ubuntu 22.04.3 LTS

MiniForge の導入

```
curl -L -O "https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-$(uname)-$(uname -m).sh"
bash Miniforge3-$(uname)-$(uname -m).sh -p conda-env/ -b
```

XLS の導入（１）

XLS のインストール.

```
source conda-env/bin/activate
conda install --yes -c litex-hub xls
```

インストールの確認.

```
interpreter_main --version
ir_converter_main --version
opt_main --version
codegen_main --version
```

Bazel の導入

```
sudo apt install apt-transport-https curl gnupg
curl -fsSL https://bazel.build/bazel-release.pub.gpg | gpg --dearmor >bazel-archive-keyring.gpg
sudo mv bazel-archive-keyring.gpg /usr/share/keyrings
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/bazel-archive-keyring.gpg] https://storage.googleapis.com/bazel-apt stable jdk1.8" | sudo tee /etc/apt/sources.list.d/bazel.list
sudo apt update
sudo apt install bazel
sudo apt install bazel-6.4.0
```

XLS の導入（２）

XLS の本体をビルドする．ビルドは時間がかかるが複数スレッド実行が有効であり，仮想環境のコア数を８程度に増やすこと推奨.

```
sudo apt-get install python3-distutils python3-dev libtinfo5 python-is-python3
git clone https://github.com/google/xls.git
cd xls
bazel test -c opt -- //xls/...
```

C++コードの高位合成テスト

下記のコード（test01.ccとする）を準備する.

```
#pragma hls_top
int add(int a, int b) { return a + b; }
```

XLSによる中間コードの生成

```
(xlsフォルダ)/bazel-bin/xls/contrib/xlscc/xlscc test01.cc > test01.ir
(xlsフォルダ)/bazel-bin/xls/tools/opt_main test01.ir > test01.opt.ir
```

Verilog HDL コードの生成

```
./bazel-bin/xls/tools/codegen_main test01.opt.ir --generator=combinational --delay_model="unit" --output_verilog_path=test01.v --module_name=xls_test --top=add
```