An NDA-free oriented Open PDK technology and EDA for small volume LSI developments

Seijiro Moriyama

Anagix Corporation
3-22-27 Numama, Zushi-city,
Kanagawa, 249-004 Japan
seijiro.moriyama@anagix.com

Tadaaki Tsuchiya, Shingo Ura

Logic Research Co., Ltd.
3-8-33-310 Fukuoka Institute of
System LSI Design Industry,
Momochi-Hama, Sawara-ku, Fukuoka, 814-0001 Japan

Abstract - IP sharing and reuse are indispensable for small-scale (multivariate) LSI development. To make this possible, it is desirable that the PDK is open and the EDA tools are also open source. We are developing a technology that splits PDK development, especially Pcell, into process-dependent and process-independent parts. The latter can receive benefits from being open source. By sharing process-independent part among various processes, layout design data can be ported between processes easily, which is the most beneficial in this technology. Our minimal EDA applies the open PDK technology to a wide range of process technologies from Minimal Fab to Skywater 130nm process. This technology can be applied to processes that require NDAs as well. Regardless of NDA, we will be able to develop PDKs at low cost and in a short period of time. We hope to provide high-value-added LSI developers with an environment where they can select semiconductor manufacturing fabs best suited for their needs.

I. Introduction

Domestic semiconductor manufacturing fabs have completely fallen behind in miniaturization, but the need for semiconductors is still alive for around 0.6um-130nm process nodes. For small-lot, high-mix products, we would like to utilize fabs according to needs such as cost and quantity. However, domestic fabs are not always motivated in the fab business, which makes it difficult for fabless companies, ventures and individuals to use domestic fabs at low cost. One of the reasons is that the fab does not have the technical ability to develop PDKs. PDK is what we need to have to develop LSIs using fabs, which includes schematic symbols, SPICE simulation models, device generation PCells for layout, DRC and LVS for layout verification.

Since PDK development is closely related to process technology, it has been developed privately by a limited number of vendors. Therefore, the PDK development is costly, and we would suspect that it is not easy for domestic fabs with uncertain future to invest in PDK development.

When developing small-lot, high-mix products, it is not realistic to develop all parts (IPs) by ourselves. Large mass-production companies can get help from IP vendors, which ventures and individuals cannot afford. In the world of software, open source has become quite common. By utilizing existing source programs, even venture companies and individuals can build a considerably complicated system. But in the hardware world, the move is just beginning.

The Skywater 130nm PDK (hereinafter abbreviated as sky130 PDK) has been released without NDA, and a shuttle service supported by Google is operated by efabless.com, which is one of the movements. It is significant that the PDK was released without the NDA restriction. Moreover, the PDK is open source. However, it should be noted that this sky130 PDK is locked to its Skywater 130nm process. From the user's point of view, the NDA-free PDK should be easy to switch from one fab to another at any time so that a fab can be chosen according to the user's needs. The current sky130 PDK does not meet that expectation.

We are hoping that open source technologies will be applied more and more in the hardware world, and that even venture companies and individuals will be able to develop considerably complicated LSIs. It is essential that IP sharing and reuse be realized, and it is desirable that EDA be open source. With that in mind, we have been working on democratizing LSI design and developing minimal EDA [1,2,3]. Minimal EDA has been applied to a new semiconductor production system, Minimal Fab [4]. Not only that, it also supports the NDA-free OpenRule1um design rule used in the MakeLSI project [5], and has already been used for trial production in the domestic 0.6um CMOS process.

Returning to the PDK technology, even for PDKs that fabs require NDA, most of the PDK source code can be made open to public if the process-dependent part is eliminated. PDK development in this way will improve functionality and quality as more people can be involved in open source part of PDK development. On the other hand, for fabs that require PDK with NDA, PDK can be provided at low-cost and in a short period of time by utilizing the idea.

In the LSI prototype that the author was recently involved in, it was decided to use some of NDA protected design rules that conflict with OpenRule1um in order to bring out the performance of the domestic 0.6um process. Since the tools recommended by the fab are expensive vendor tools, we have developed our own PDK to use the open source layout tool KLayout [6]. At that time, the PCell code developed for OpenRule1um was divided into a process-dependent part and a non-process-dependent part. And we newly created a dedicated PDK using process-dependent code and process-independent open source code. In order to show that this technology (called open PDK technology) can be applied to other processes, we have applied it to the Skywater 130nm

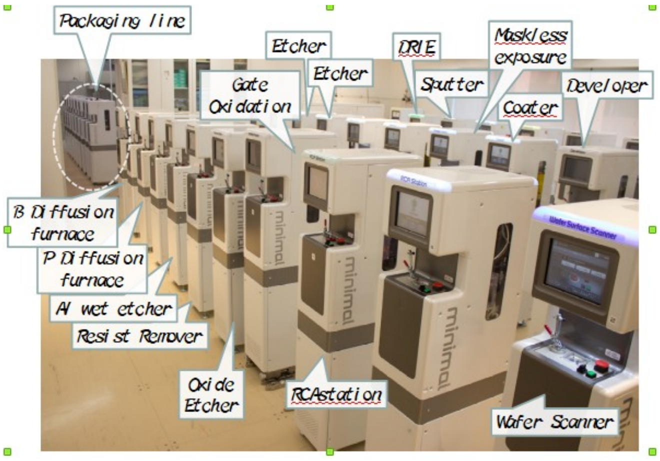


Fig. 1. Minimal Fab manufacturing equipments.

process, and obtained the positive prospect.

Chapter 2 introduces the EDA composed of open source. This has a track record of application to LSI development in Minimal Fab, which is a new LSI manufacturing system. Chapter 3 describes examples of applying PCell, one of the core technologies of PDK, to different process technologies and devising ways to do so. Next, Chapter 4 describes the open PDK technology that will help fabs develop NDA-free PDKs in a soft landing fashion. Chapter 5 introduces the PDK developed by applying this technology, and demonstrates design examples developed in a short period of time by using the PDK.

II. Introduction of minimal EDA

A. Why open-source?

First off, vendor tools are too costly for small volume LSI developers like individuals or small companies who do not have much money.

There are several good free tools for LSI design, but there is no guarantee that we can rely on these tools for a long time. It is okay to use these just to develop something and use the results. In layout's case, we can develop polygon layout using Glade and leave GDSII layout as a result. We can distribute and re-use the GDSII layout (if NDA is not an issue). However, if we use the tool's proprietary PCell, our freedom to re-use IPs is restricted. We need openness in tools to make IPs possible to fulfill various technology requirements.

B. Minimal Fab and Minimal EDA

Minimal Fab [4] developed by the AIST, uses a 0.5 inch wafer, performs direct writing without the need for a mask, and allows small volume production from only one piece. The conventional shuttle service using fabs for mass production takes about 3 months of trial manufacture time, whereas Minimal Fab aims at trial manufacturing for a

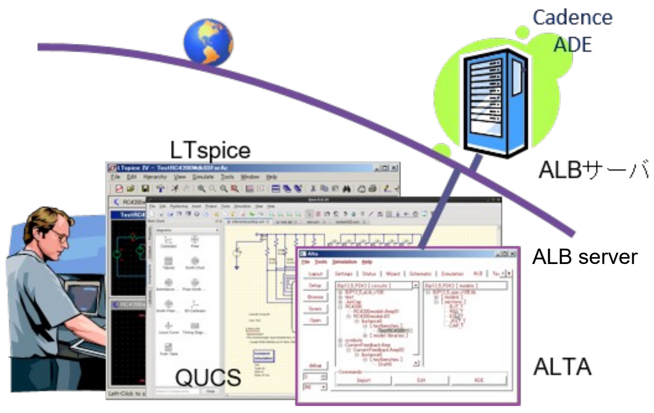


Fig.2 ALB/ALTA overview

minimum of 3 days. In Minimal Fab, the equipment responsible for the semiconductor manufacturing process is housed in a 294 mm wide × 450 mm deep × 1440 mm high case, using a 100 V household power supply and consuming extremely pure water etc. There is little need for a clean room. Therefore, the initial investment and running costs of the factory are minimal. Fig. 1 shows the manufacturing equipment that makes up the minimal fab.

To fulfill Minimal Fab needs, Minimal EDA is designed to integrate free and open source tools with ALB/ALTA as a core [1]. Minimal Fab is targeted to fabricate LSI chips in only 3 days. Minimal EDA provides a total solution to take advantage of Minimal Fab. Minimal EDA servers, where all the open source tools have been installed, are ready for Minimal Fab users.

C. Xschem and Ngspice/XYCE for circuit design

Xschem is an open source schematic editor attracting attention because it has been used in the Skywater 130nm design. The circuit simulator supports NGspice and XYCE.

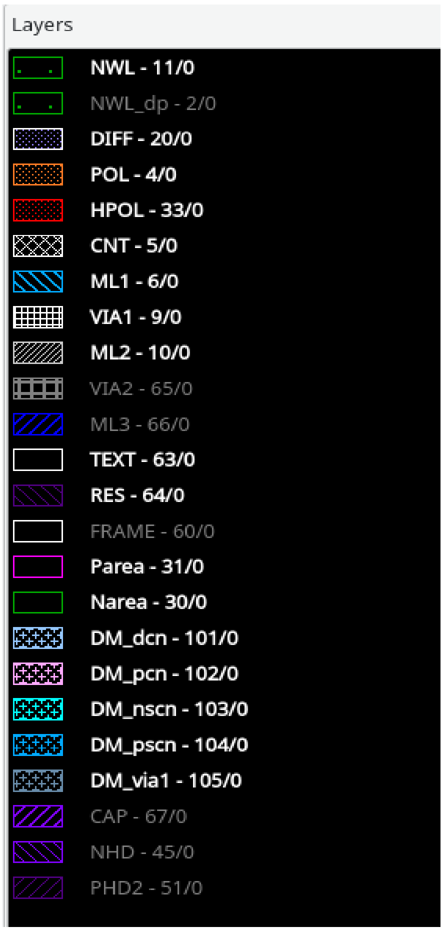
XYCE is an open-source simulator built from scratch by National Sandia Laboratory in the US. It supports various latest compact SPICE models and can handle very large integrated circuits.

D. KLayout for layout design

KLayout is an open-source LSI layout design tool. It has Qt-based GUI and can be extended using Ruby or Python languages. Design Rule Check (DRC) rules can be implemented as Ruby scripts. Layout vs. Schematic (LVS) verification is also available. Most important thing with respect to IP is PCell, which is available as either Ruby or Python scripts.

E. ALB/ALTA and tools integration

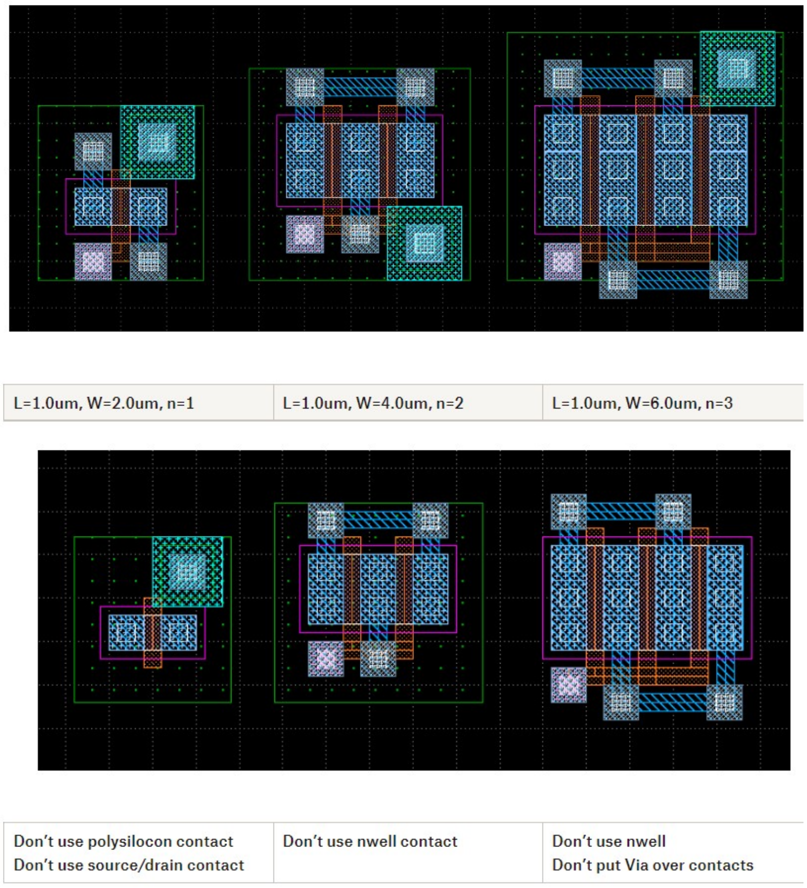
ALB/ALTA is a cloud-based hybrid system, where ALB is a web-server based design data manager aiming at ease of Analog IP development and reuse [1,2]. ALB/ALTA looks like Fig. 2 from user's point of view. ALTA is a cloud



Layers

- NWL - 11/0
- NWL_dp - 2/0
- DIFF - 20/0
- POL - 4/0
- HPOL - 33/0
- CNT - 5/0
- ML1 - 6/0
- VIA1 - 9/0
- ML2 - 10/0
- VIA2 - 65/0
- ML3 - 66/0
- TEXT - 63/0
- RES - 64/0
- FRAME - 60/0
- Parea - 31/0
- Narea - 30/0
- DM_dcn - 101/0
- DM_pcn - 102/0
- DM_nscn - 103/0
- DM_pscn - 104/0
- DM_via1 - 105/0
- CAP - 67/0
- NHD - 45/0
- PHD2 - 51/0

Fig. 3.Layers definition of OpenRule1um design rule for Standardl CMOS process



L=1.0um, W=2.0um, n=1	L=1.0um, W=4.0um, n=2	L=1.0um, W=6.0um, n=3
Don't use polysilicon contact Don't use source/drain contact	Don't use nwell contact	Don't use nwell Don't put Via over contacts

Fig. 4. PCell examples (Standardl CMOS process)

application running on local machines to control various design tools including schematic editors, simulators, PCB design editors and measurement programs. ALB/ALTA facilitates these with consistent mechanism to handle tools' data. ALB/ALTA as a unit is an analog design focused design environment for designers. ALB/ALTA supports LTspice, Xschem, EEschema and Qucs for schematic editor and circuit simulator.

III. PCell commonization for different types of fabs

A. *What is PCell?*

PCell is an abbreviation for parameterized cell, and is a technology that instantly generates different shapes by giving parameters for active elements such as MOSFETs and passive elements used in LSI layout. Fig. 3 shows the layer definition of OpenRule1um based on a standard CMOS process, and Fig. 4 shows examples of its PCells.

By changing the parameters shown in Table 1, the shape of the transistor changes instantly as shown in Fig. 4.

B. *Application to Minimal Fab (SOI CMOS)*

We devised a way to apply PCell, which assumes a standard CMOS process, to SOI CMOS.

Fig. 5 shows an example of a Minimal Fab (SOI CMOS) PMOS transistor. It is similar to Fig. 4, but there are some differences. Fig. 6 shows a conversion of this PCell into manufacturing layers.

TABLE I. PCell parameters example for PMOS device

Parameter name	Description
Pch W	PMOS gate width; default is 2.4 um
Pch L	PMOS gate length; default is 0.6 um
Use nsub contact	Create N well contact
Use nwell	Create N well

There are two major differences between Fig. 5 and the standard bulk CMOS (Fig. 4). (1) In order to prevent the TiN of the gate material from breaking, the height difference at SOI edge is overcome by covering ML1. (2) In standard CMOS, on the contact to the gate/source/drain etc., VIA is placed so that layout beginners can easily connect metal layers. In the case of this SOI CMOS example, VIA is temporarily abolished because only ML1 can be used for wiring layers. This kind of difference can be absorbed by setting the PCell parameter as an optional flag.

In the current SOI CMOS process, the source and drain are created by diffusion. Since the standard CMOS utilizes a self-aligned gate, a diffusion layer (DIFF in Fig. 3) is laid directly under the gate. In the layout design of Minimal Fab, Parea layer is used instead of DIFF in Fig. 5, and in the manufacturing layer, Parea is cut with a little overlap with S /

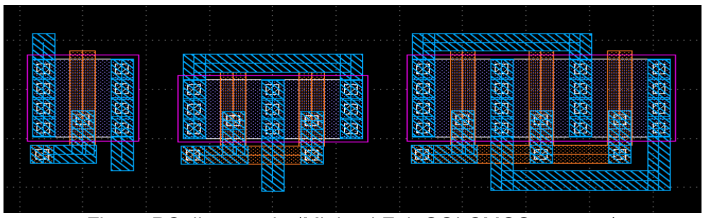


Fig. 5. PCell example (Minimal Fab SOI CMOS process)

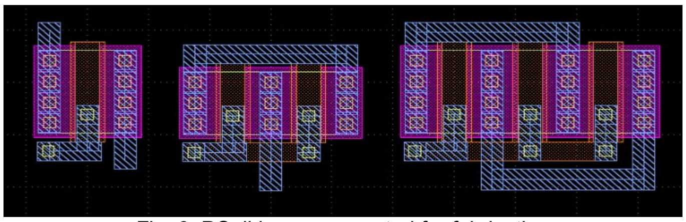


Fig. 6. PCell layers converted for fabrication

D as shown in Fig. 6. Also, the SOI layer is replaced by DIFF. This set of replacements allows users to use PCell in much the same way as standard CMOS, even with Minimal Fab designs.

This example of PCell for Minimal Fab demonstrates that our PDK can use equivalent PCells for different manufacturing processes. Our PDK is open source and NDA-free like the sky130 PDK, but fundamentally different from it since ours is not linked to a specific fab.

Sky130 is NDA-free. However, it is another matter whether the design using sky130 can be easily switched to another company's fab. Whether sky130 requires an NDA or not, this doesn't change. With our open PDK technology, PDKs are made the same with different content, so it's easy to switch fabs because you can use the same Open Source EDA and the same set of PCells. Therefore, users can select semiconductor manufacturing fabs according to their needs.

IV. Open PDK technology

A. *Reconcile with fabs that stick to NDAs*

In the LSI prototype the author involved this time, in order to bring out the maximum performance of the domestic 0.6um process, it was decided to use some design rules that require NDA instead of NDA-free OpenRule1um. Therefore, we have developed a dedicated PDK. At that time, the PCell code developed for OpenRule1um was divided into a process-dependent part and a non-process-dependent part, and a dedicated PDK was realized using process-dependent code and process-independent open source code.

To show that this technology can be applied to other processes, we applied it to the Skywater 130nm process, which is open to the public without NDA, and we got the prospect of realizing our PCell for it. Fig. 8 shows a PCell prototype for the skywater 130nm process using the open PDK technology. This PCell has the same usability as the one for OpenRule1um because the parameters are the same. Even if the fabs are different, if the PCell is the same, it will be much easier for the user to use another fabs properly.

B. *Divide PCell into process-dependent and non-process-dependent parts*

As shown in Fig.7, PCell can be divided into process-dependent and non-process-dependent parts. The dependent part is a part for setting a layer number unique to a process, setting a distance between layers determined by a process, and the like. The independent part (NMOS/PMOS common drawing routine in the figure) is invoked by 'super' method, which draws the NMOS/PMOS figures. This mechanism is easy to implement because KLayout's PCell is written in the object-oriented Ruby language.

As shown in Figs. 4 and 5, even if the processes are different, the MOS transistor is composed of a gate figure, a source figure, and a drain figure, after all.

In Fig.7, PCell parameters (like l, w) are accessible in the drawing routine and process specific parameters are passed by a hash variable named params when invoked from the process specific part. This figure generation part is process independent and can be published as an open source program.

The significance of being open source is not small. Even a PDK protected by an NDA can benefit from open source in its subordinate parts. Since many people can be involved in the open source part development, the functionality and quality are expected to be improved. In addition, the technologies hidden inside a specific company in the past will be improved and further developed in various ways by democratization. It will also have a positive effect on the automatic layout composition technology.

V. Layout design using PTS06 PDK

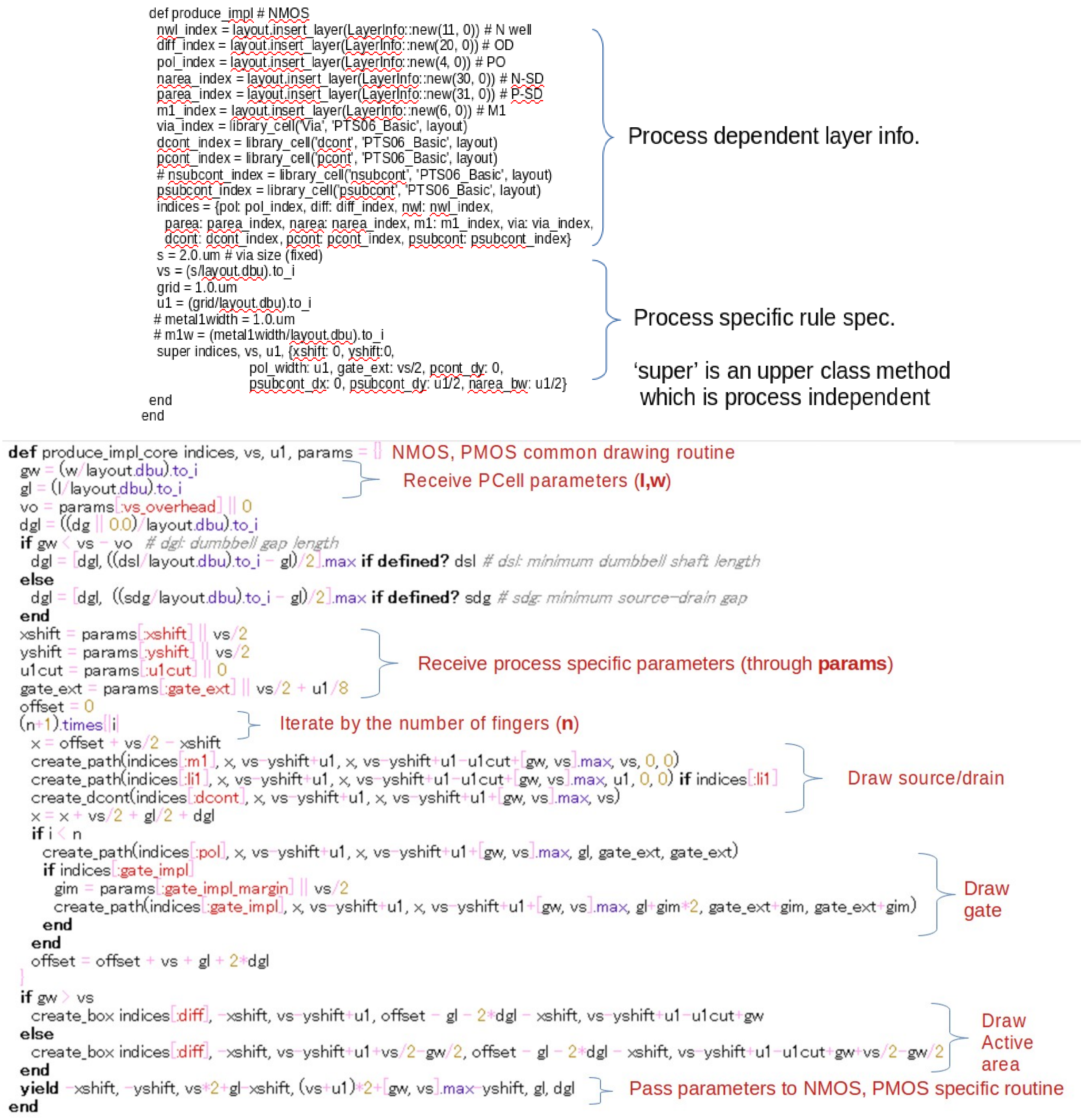
A. *PTS06 PDK*

In order to demonstrate that the PDK developed using the open PDK technology can be used for actual LSI development, we have developed a PDK named PTS06 for the domestic 0.6um process. The process-independent part has been published as part of the open source OpenRule1um PDK [7]. LVS is almost unchanged, but some modifications have been made on DRC from OpenRule1um. It is a future task to realize DRC by dividing it into process-dependent part and non-process-dependent part.

The number of PCell program lines in PTS06 is 599, and the number of process-independent program lines included in OpenRule1um is 596. Polysilicon resistor and high resistivity Polysilicon resistor have been added with a dedicated code to PTS06 in order to realize them with high accuracy. Therefore, the PTS06 PCell code became longer. As for the MOS element alone, the description of PTS06 is 111 lines, and the common code of OpenRule1um is 285 lines, and it can be seen that the process-dependent part is quite short.

B. *Layout design example using PTS06 PDK*

The overall view of the chip is shown in Fig. 9. This chip consists of a dark display part that cannot be shown in detail



```
def produce_impl # NMOS
  nwl_index = layout.insert_layer(LayerInfo::new(11, 0)) # N well
  diff_index = layout.insert_layer(LayerInfo::new(20, 0)) # OD
  pol_index = layout.insert_layer(LayerInfo::new(4, 0)) # PO
  narea_index = layout.insert_layer(LayerInfo::new(30, 0)) # N-SD
  parea_index = layout.insert_layer(LayerInfo::new(31, 0)) # P-SD
  m1_index = layout.insert_layer(LayerInfo::new(6, 0)) # M1
  via_index = library_cell('Via', 'PTS06_Basic', layout)
  dcont_index = library_cell('dcont', 'PTS06_Basic', layout)
  pcont_index = library_cell('pcont', 'PTS06_Basic', layout)
  # nsubcont_index = library_cell('nsubcont', 'PTS06_Basic', layout)
  psubcont_index = library_cell('psubcont', 'PTS06_Basic', layout)
  indices = {pol: pol_index, diff: diff_index, nwl: nwl_index,
    parea: parea_index, narea: narea_index, m1: m1_index, via: via_index,
    dcont: dcont_index, pcont: pcont_index, psubcont: psubcont_index}
  s = 2.0.um # via size (fixed)
  vs = (s/layout.dbu).to_i
  grid = 1.0.um
  u1 = (grid/layout.dbu).to_i
 # metal1width = 1.0.um
 # m1w = (metal1width/layout.dbu).to_i
  super indices, vs, u1, {xshift: 0, yshift:0,
                     pol_width: u1, gate_ext: vs/2, pcont_dy: 0,
                     psubcont_dx: 0, psubcont_dy: u1/2, narea_bw: u1/2}
 end
end
```

Process dependent layer info.
Process specific rule spec.
'super' is an upper class method
which is process independent

```
def produce_impl_core indices, vs, u1, params = {}
  gw = (w/layout.dbu).to_i
  gl = (l/layout.dbu).to_i
  vo = params[:vs_overhead] || 0
  dgl = ((dg || 0.0)/layout.dbu).to_i
  if gw < vs - vo  # dgl: dumbbell gap length
    dgl = [dgl, ((dsl/layout.dbu).to_i - gl)/2].max if defined? dsl # dsl: minimum dumbbell shaft length
  else
    dgl = [dgl,  ((sdg/layout.dbu).to_i - gl)/2].max if defined? sdg # sdg: minimum source-drain gap
  end
  xshift = params[:xshift] || vs/2
  yshift = params[:yshift] || vs/2
  u1cut = params[:u1cut] || 0
  gate_ext = params[:gate_ext] || vs/2 + u1/8
  offset = 0
  (n+1).times{|i|
    x = offset + vs/2 - xshift
    create_path(indices[:m1], x, vs-yshift+u1, x, vs-yshift+u1-u1cut+[gw, vs].max, vs, 0, 0)
    create_path(indices[:li1], x, vs-yshift+u1, x, vs-yshift+u1-u1cut+[gw, vs].max, u1, 0, 0) if indices[:li1]
    create_dcont(indices[:dcont], x, vs-yshift+u1, x, vs-yshift+u1+[gw, vs].max, vs)
    x = x + vs/2 + gl/2 + dgl
    if i < n
      create_path(indices[:pol], x, vs-yshift+u1, x, vs-yshift+u1+[gw, vs].max, gl, gate_ext, gate_ext)
      if indices[:gate_impl]
        gim = params[:gate_impl_margin] || vs/2
        create_path(indices[:gate_impl], x, vs-yshift+u1, x, vs-yshift+u1+[gw, vs].max, gl+gim*2, gate_ext+gim, gate_ext+gim)
      end
    end
    offset = offset + vs + gl + 2*dgl
  }
  if gw > vs
    create_box indices[:diff], -xshift, vs-yshift+u1, offset - gl - 2*dgl - xshift, vs-yshift+u1-u1cut+gw
  else
    create_box indices[:diff], -xshift, vs-yshift+u1+vs/2-gw/2, offset - gl - 2*dgl - xshift, vs-yshift+u1-u1cut+gw+vs/2-gw/2
  end
  yield -xshift, -yshift, vs*2+gl-xshift, (vs+u1)*2+[gw, vs].max-yshift, gl, dgl
end
```

NMOS, PMOS common drawing routine
Receive PCell parameters (**l,w**)
Receive process specific parameters (through **params**)
Iterate by the number of fingers (**n**)
Draw source/drain
Draw gate
Draw Active area
Pass parameters to NMOS, PMOS specific routine

Fig. 7. Split PCell into process-dependent and non-dependant parts

by the NDA, and a bright display part that follows only the OpenRule1um rule. The details of the latter are shown in Fig. 10. They are all analog circuits composed of a class AB audio amplifier, a thermal shutdown circuit, and a bandgap voltage supply circuit in addition to a general-purpose operational amplifier. All elements were generated by PCell. The NDA part is mostly a digital circuit, but all the elements are generated by PCell. The design period was extremely short, about 3 weeks.

Since the DRC is the one basically diverted from

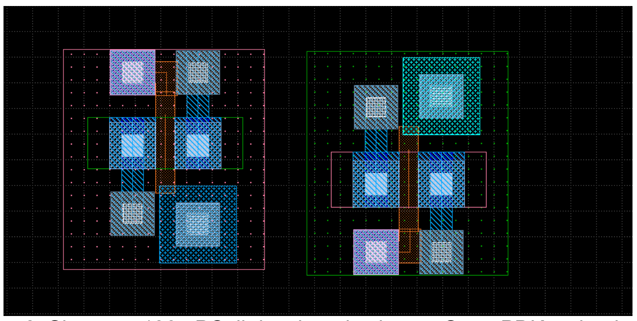


Fig. 8. Skywater 130n PCell developed using an Open PDK technology

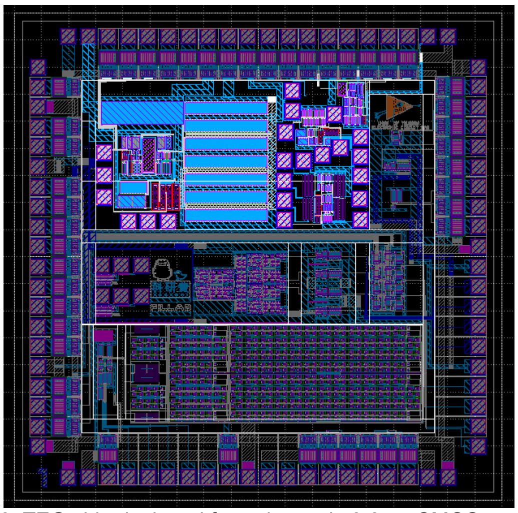


Fig. 9. TEG chip designed for a domestic 0.6um CMOS process

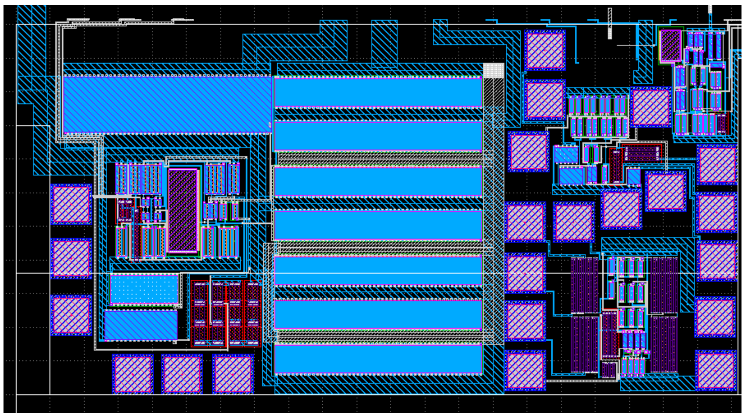


Fig. 10. OpenRule1um part of TEG chip in Fig. 9

OpenRule1um, it couldn't cover process-specific rules such as wide metal rules. Therefore, the final verification was performed using a vendor tool guaranteed by the domestic fab. The final verification was completed in almost a day and most of the design and verification could be performed with Klayout only.

VI. Summary and Conclusions

In order to facilitate and succeed in the development of high-value-added, low-volume, high-mix LSIs, it is necessary to create a situation in which IPs can be shared and reused. For that purpose, it is desirable that the PDK provided by the fab does not require NDA and all the EDA tools are open source. However, it will take some time before it happens. This time, we have developed an open PDK technology that can be used in Minimal EDA. It divides the PDK into process-dependent parts and other parts that can be open sourced. Even for the development of PDKs that require NDA, by increasing the open source part, not only can we expect quality improvement due to openness, but we will be able to develop more PDKs at low cost and in a short period of time. The PDKs for specific fabs that do not require NDA have emerged, but PDK development that applies open PDK technology is effective in enabling users to freely and easily select fabs. Using a fab that has a proven track record in the OpenRule1um rule, we developed a PDK that applies open PDK technology and applied it to actual LSI development. In the future, including the Skywater 130nm process, we will apply open PDK technology regardless of whether NDA is necessary or not, and create a situation where fabs are easy to use for high value-added LSI development users.

Acknowledgements

We would like to thank Dr. Tanimoto, Mr. Takahashi, Dr. Yoshino, Professor Shouno of University of Tsukuba, Professor Sekine of Meiji University, Professor Yasuda of Hosei University, and students of each laboratory, for their help and encouragements. We are grateful to Professor Akita of the University Kanazawa who has led the MakeLSI project. The development at a domestic fab was supported by President Yamada of A.LSI Design Co., Ltd. The development at Minimal Fab was taken care of by the people concerned. Their efforts are much appreciated.

References

[1] Seijiro Moriyama, Manabu Ishibe, Chikau Takahashi, Michitaka Yoshino, Akira Yasuda: "Cloud-based Analog LSI CAD system for cooperative design", AVIC, Boston, 2016.

[2] Seijiro Moriyama, Tadaaki Tsuchiya, Michitaka Yoshino, Akira Yasuda: "EDA for Minimal Fab and dynamic design documentation", AVIC, ChiangMai, 2018.

[3] Seijiro Moriyama, Tadaaki Tsuchiya, Katsuhiko Wakasugi, Michitaka Yoshino and Akira Yasuda: "Minimal EDA for Open-minded LSI developments", AVIC, Taiwan, 2019

[4] Minimal Fab association homepage, June 1st, 2022. <https://www.minimalfab.com/en/>

[5] MakeLSI: FrontPage, June 12, 2012. <http://ifdl.jp/make_lsi/>

[6] Klayout homepage, June 2nd, 2022. <https://klayout.de>

[7] OpenRule1um for Minimal EDA homepage, June 3rd, 2022. <https://github.com/mineda-support/OpenRule1um>