Challenge to Fabricate LSI without NDA with Open Method

Naohiko Shimizu

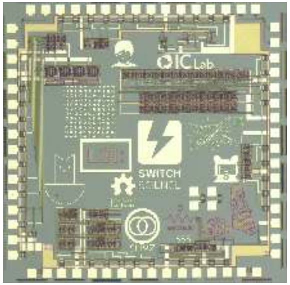
Tokai University, Japan

8th, July, 2022

Agenda

Motivation

- Exchange LSI ideas among people including non-professionals
- Give opportunities to get basic knowledge of LSI making in classrooms
- Minimize efforts to support new processes
- Utilize open tools to make LSI as solid as possible



MakeLSI: multi project chip on FAIS by Prof. Akita. Several non-professional people put their own layout. Left bottom one is mine.

NDA free trials

- Case 1: FAIS 2um 2M CMOS
  FAIS stands for Kitakyushu Foundation for the Advancement of Industry, Science and Technology. The foundation depends mostly on Kitakyushu city financially.
- Case 2: Phenitec 0.6um 3M CMOS with scalable method
  Phenitec process is under their NDA. We use scalable cell library and design flow to make LSI without NDA.
- Case 3: YSS Minimal Fab SOI-CMOS 2M 6um
  YSS is an equipment vendor and now try to establish the design rules with users. `https://web-material3.yokogawa.com/19/24177/files/DevicePrototypingR1.pdf`

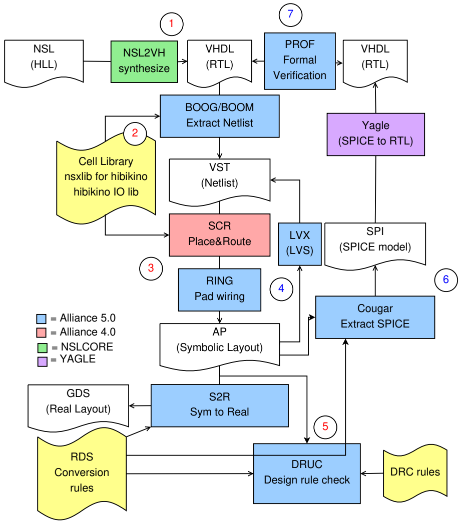
FAIS

Kitakyushu city lends the facilities on hourly basis. They provide human technical supports and chemicals required freely. The facilities were supported by the Ministry of Education Japan, but the support was expired at April 2022.
`https://www.ksrp.or.jp/fais/mic/nano/`

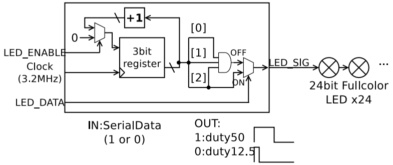
- The community makes test chips to define design rules
- Volunteers who has equipment extract the characteristics
- I intensively use the open tools to make design flow on FAIS. Especially I use LIP6 tools and prefers the scalable rules of Alliance tool set.
- We need to send students two weeks on the facility.

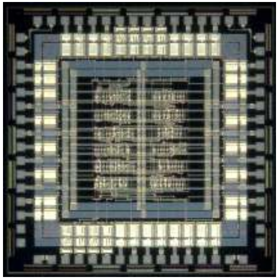
FAIS 2um 2M CMOS design flow

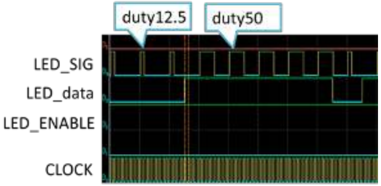
- scalable cell library for core and io designed by me and my student
- Alliance 4.0 for 2 metal place and route
- Alliance 5.0 for logic synthesize, pad wiring, symbolic to real conversion, logic versus schematic check, design rule check, layout to netlist extraction, RTL to RTL formal verification
- Yagle for netlist to RTL extraction

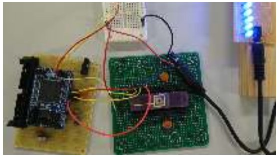


FAIS 2um 2M CMOS versa-writer









NSL source code of the chip

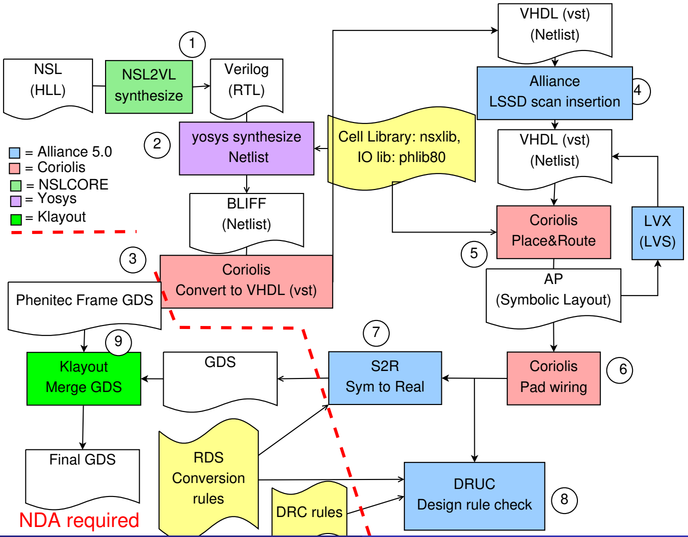
```
declare led_top{
    input     led_enable;
    input     data_i;
    output     led_sig;
}
module led_top{
    reg count[3];
    reg sync_e1=0,sync_e2=0;
    reg sync_d1=0,sync_d2=0;
    func_self c_reset();
    sync_e1:=sync_e2;
    sync_e2:=led_enable;
    sync_d1:=sync_d2;
    sync_d2:=data_i;
if(c_reset){ count:=0;
  }else{ count++; }
if(~sync_e1){ c_reset(); }
  else{
if(count[2]&~count[1]&count[0]){
      led_sig=1;
}else if(sync_d1&
 (count[2]^~(count[1]|count[0]))){
      led_sig=1;
    }else{
      led_sig=0;
      }
.  }
}
```

Phenitec 0.6um 3M CMOS shuttle

- They provide a low cost shuttle on the process.
  `https://www.phenitec.co.jp/pdf/shuttle_ver3.pdf`
- Phenitec itself requests NDA, but they allow us to open the user measured data.
- Volunteers who has equipment extract the characteristics from test chips.
- For NDA free trials, we separate the design users from the real GDS. Prof.Akita uses 1um pseudo process, I use Alliance scalable layout.

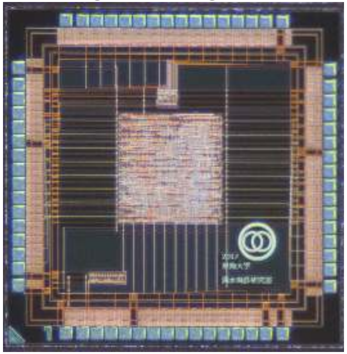
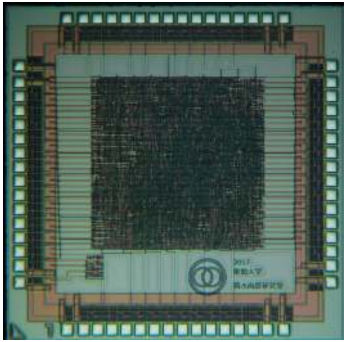
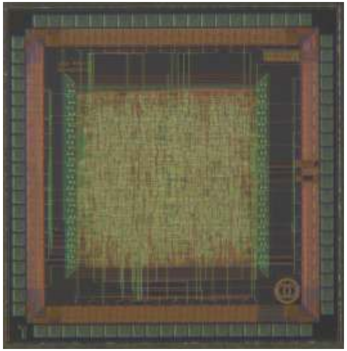
Phenitec 0.6um 3M CMOS design flow

- Scalable cell library NSXLIB and phlib80
- Yosys, Coriolis, Alliance 5.0, Klayout
- RDS conversion rules, DRC rules are under NDA
- End user design with MOSIS rules, who has NDA generate final GDS



Phenitec Chips with our flow

The logic design user don't have to see the Phenitec provided NDA data.

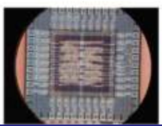
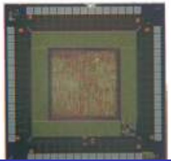
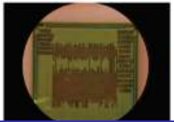




YSS Minimal Fab SOI-CMOS 2M 6um

- We can make CMOS chip with 1000 Tr. in 4 days.
- YSS is a member of Minimal Fab consortium. Currently, they try to make solid design rules with users.
- Volunteers who has equipment extract the characteristics from test chips.
- I planed to use Alliance scalable layout. But they have serious trouble on VIA and it prevent me to use my cell library.
- I will continue to support the process when the VIA trouble resolved.

Scalable Design Rules

- Back to 1980s, Mead and Conway proposed λ-rules. It did work well in '80s with MOSIS SCMOS.
- Alliance has scalable design rules which uses λ for the segment positioning. Unlike M&C rules, Alliance segment is a virtual segment which length and width and layers can be modified with parameter file called RDS.
- With the Alliance sxlib, I had made a few chips with 0.35um, 0.18um, 1.2um technologies.

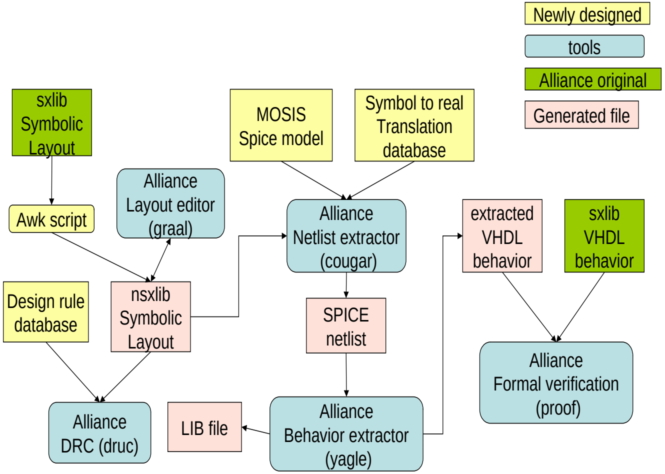
Nsxlib development

Every time when I use a new process, I have to modify Sxlib in Alliance extensively to meet the design rules. Then I decided to make a new library Nsxlib which will be more flexible.

- Change the basic λ as a half of gate length, where Sxlib λ is the gate length.
- Use MOSIS DEEP with $\lambda = 90nm$ as pseudo physical rule.
- Generate initial Nsxlib from SXLIB with λ convert and adjust the layout to meet the rule manually.
- modify Nsxlib to fit with Hibikino
- modify Nsxlib to fit with Phenitec
- modify Nsxlib to fit with FreePDK45

Nsxlib design flow

- scalable cell library NSXLIB designed by myself
- Formal verification against sxlib VHDL
- Yosys, Coriolis, Alliance compatible cell library.



Conclusion

- We made trials to fabricate LSI without NDA
- The key of our efforts is the scalable cell library
- We will adapt our method against other processes.
- Missing feature: I need a test pattern generator for LSSD scan chain.

Future work:

- I will adapt NSXLIB against more processes including Skywater130.
- Now we have 20 days programming course for 10 to 14 years old child.
  We want to make advanced course including chip making

Appendix:HLL NSL designed by me

https://www.ip-arch.jp/unsupported

```
declare gcd {
  input x[16], y[16];
  output z[16];
  func_in start(x,y);
  func_out done(z);
}
module gcd {
  reg a[16],b[16];
  proc_name gcd_ex(a,b);
  func start gcd_ex(x,y);
  proc gcd_ex seq {
  while(a!=b) {
  if(a>b) a:=a-b;
  else    b:=b-a;
  }
  done(a);
  finish();
  }
```

- NSL can compile to VHDL(Alliance, GHDL), VerilogHDL, SystemC(RTL)
- Clock accurate HLL
- Parameterized IP(in VHDL,Verilog) support