Protecting proprietary data in an open environment

Tim Edwards

efabless.com

June 13, 2019

FOSSi foundation WOSH 2019 Zürich

WOSH

efabless.com

Protecting proprietary data in an open environment

The efabless model:

- Community-based engineering design
- Open Source tools, free to use on the platform
- Design IP in a “Marketplace” catalog, try for free

  IP blocks may be open or closed, free or for purchase/fee for use

- Custom PDKs installed and available on the platform
- No signing NDAs with foundries

- Fast, easy experimentation of ideas without up-front costs
- Full access to enough data and tools to get designs done

efabless.com

Protecting proprietary data in an open environment

Pros and cons

+ Foundries are happy not to sign many NDAs with low ROI
+ Foundries are happy not to support many inexperienced users

– Foundries do not like exposing proprietary data
– Foundries do not like to give away IP for free

Protecting propritary data is an issue both for foundry data and for user (including 3rd-party IP) data. Non-open-source IP should be protected without compromising user access to open-source IP.

efabless.com

Protecting proprietary data in an open environment

What may foundries be willing to expose?

- Behavioral verilog models
- Abstract views of layout (LEF)
- BEOL (metal stack) DRC rules
- Obfuscated primitive devices and layout

more likely

⋮ (gray area)

- Device electrical parameters
- Complete DRC rules
- SPICE models of primitive devices
- Mask-level views of layout (GDS)

less likely

efabless.com

Protecting proprietary data in an open environment

- Behavioral verilog models
- Abstract views of layout (LEF)
- BEOL (metal stack) DRC rules
- Obfuscated primitive devices and layout

← Build the design environment around these.

⋮ (gray area) ← Negotiate

- Device electrical parameters
- Complete DRC rules
- SPICE models of primitive devices
- Mask-level views of layout (GDS)

← Protect as requested by the foundry

efabless.com

Protecting proprietary data in an open environment

Behavioral verilog models

real-valued verilog (simple) vs. verilog-A/AMS (often proprietary)

```
real period, lastedge, refpd;



// Toggle clock at rate determined
// by period
always @(CLK or EN_VCO) begin
    if (EN_VCO == 1'b1) begin
        #(period / 2.0);
    end else if (EN_VCO == 1'b0) begin
        CLK <= 1'b0;
    end else begin
        CLK <= 1'bx;
    end
end
```

```
// Update period on every reference
// rising edge
always @(posedge REF) begin
    if (lastedge > 0.0) begin
        refpd = $realtime - lastedge;
        // Adjust period towards 1/8
        // the reference period
        period = (0.99 * period) +
                 (0.01 * (refpd / 8.0));
    end
    lastedge = $realtime;
end
```

examples here: https://github.com/efabless/picorv32-soc-raven

Protecting proprietary data in an open environment

Managing foundry-proprietary data in layout:

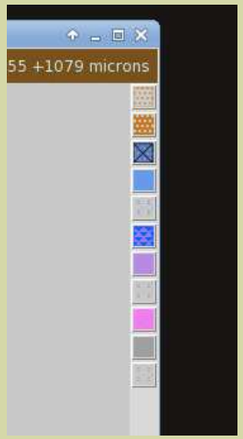
Techfiles made with "privileged" and "non-privileged" versions.

"Non-privileged" versions:

- Cannot generate GDS (no GDS output rules)
- Cannot read GDS (no GDS input rules)
- Cannot run DRC on diffusion / poly / implant layers
- Can enforce no-overlap or exact-overlap rules for layers

All versions can view all defined layers

Users discouraged from drawing layers with no viewable DRC rules by edit locking.

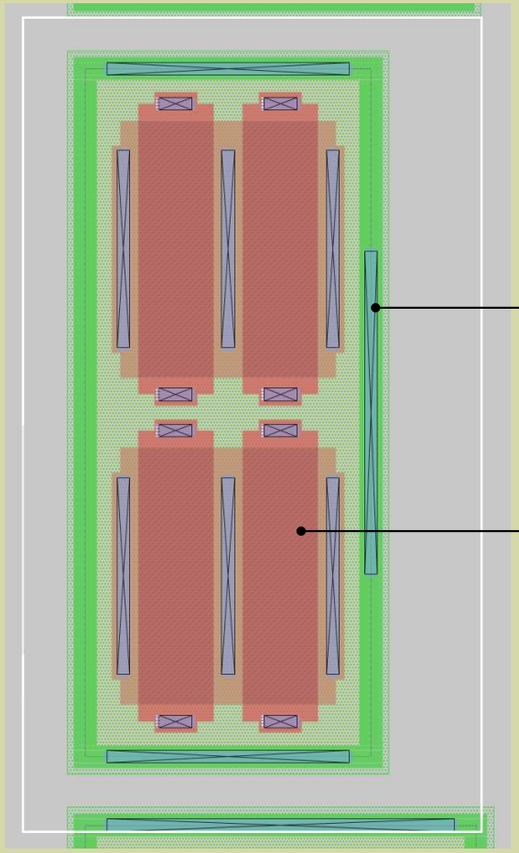


user editable layers

all defined layers

Protecting proprietary data in an open environment

Obfuscated primitive devices and layout

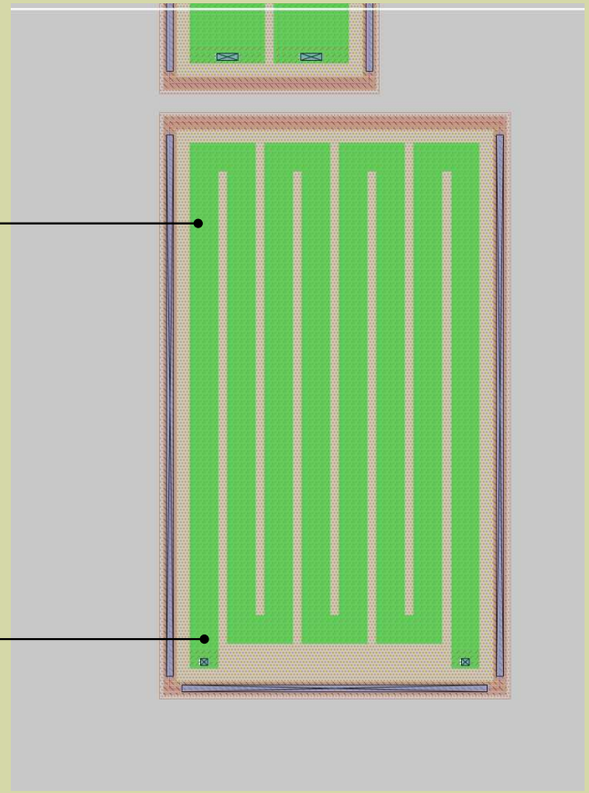


No DRC rules for diffusion and polysilicon: DRC-by-design enforced by required guard ring and no-overlap rules.

Contacts do not show cut layers so cut size, spacing, and overlap are not represented. Cuts generated by algorithm when writing GDS output.

No implant layers shown. All implant layers are automatically generated on output.

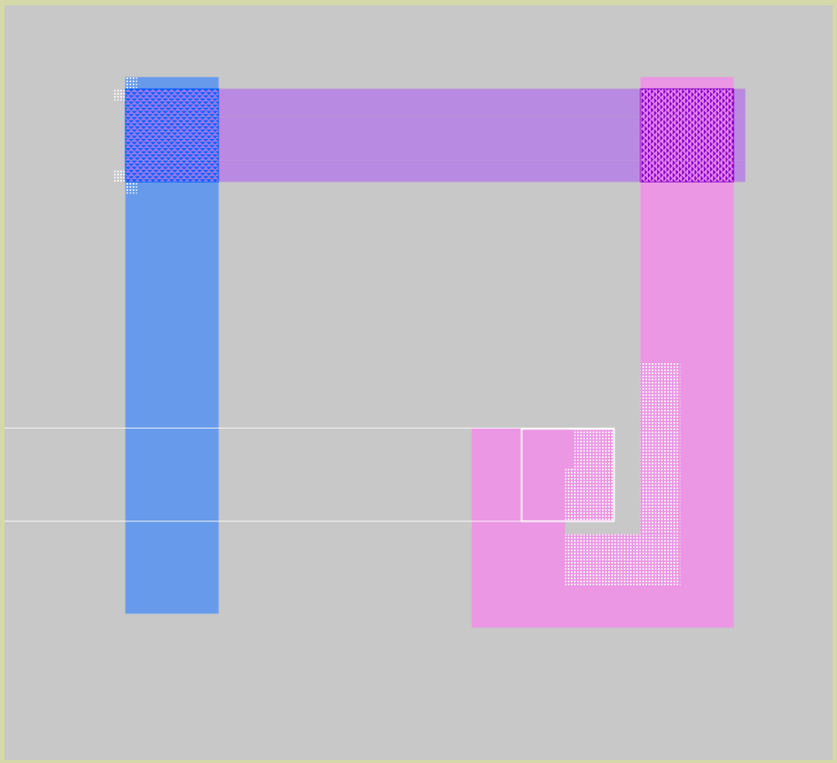
GDS output only generated from scripts running at a privileged user level and able to access the full techfile.



efabless.com

Protecting proprietary data in an open environment

BEOL (metal stack) DRC rules



Real DRC:

width and spacing rules
for all metal layers.

Obfuscated DRC:

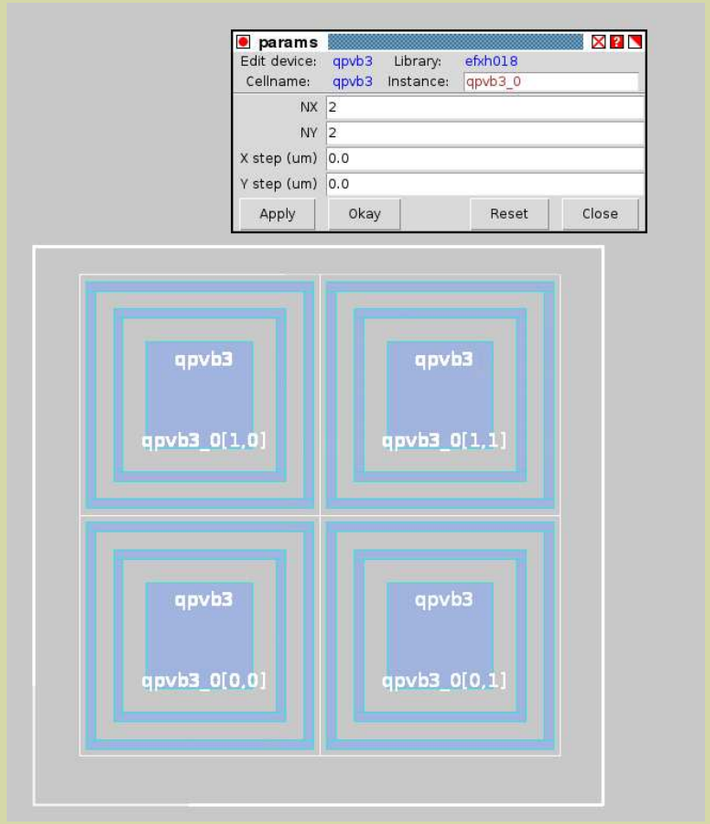
contacts represented like
"via generate" rules in LEF.

contact surround and extend
rules are not directly encoded.

Protecting proprietary data in an open environment

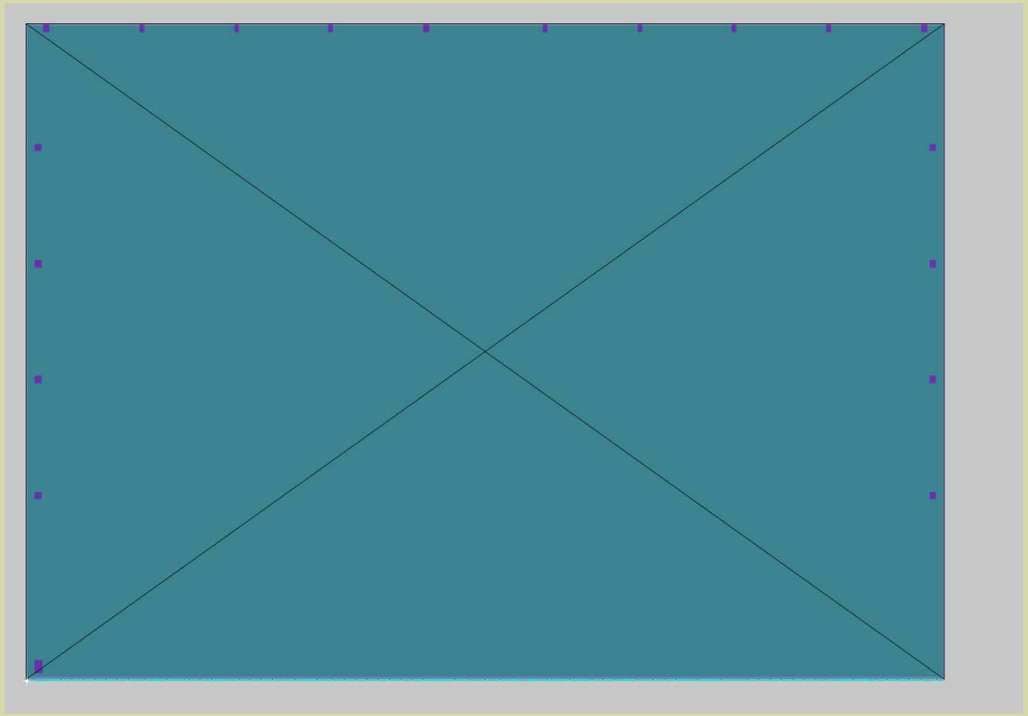
Enforce use of cell abstract views in layout.

- Less detail (like standard cell abstract views—BEOL layers only). Bounding boxes show how to abut cells.
- Often need to add "keep-out" area layers to prevent generation of DRC errors that cannot be seen by the user.
- Not a traditional method, as evidenced by (vendor data) abstract layouts that violate DRC rules
- No consistent definition of obstruction layers across vendors.
- Convert LEF data to "maglef" cells to enforce consistency and remove apparent DRC violations



Protecting proprietary data in an open environment

Abstract views of IP layout (LEF)



SRAM IP block, Magic layout derived from LEF.

Same methods as used for primitive devices.

Entire chips are mainly just collections of abstract blocks routed together.

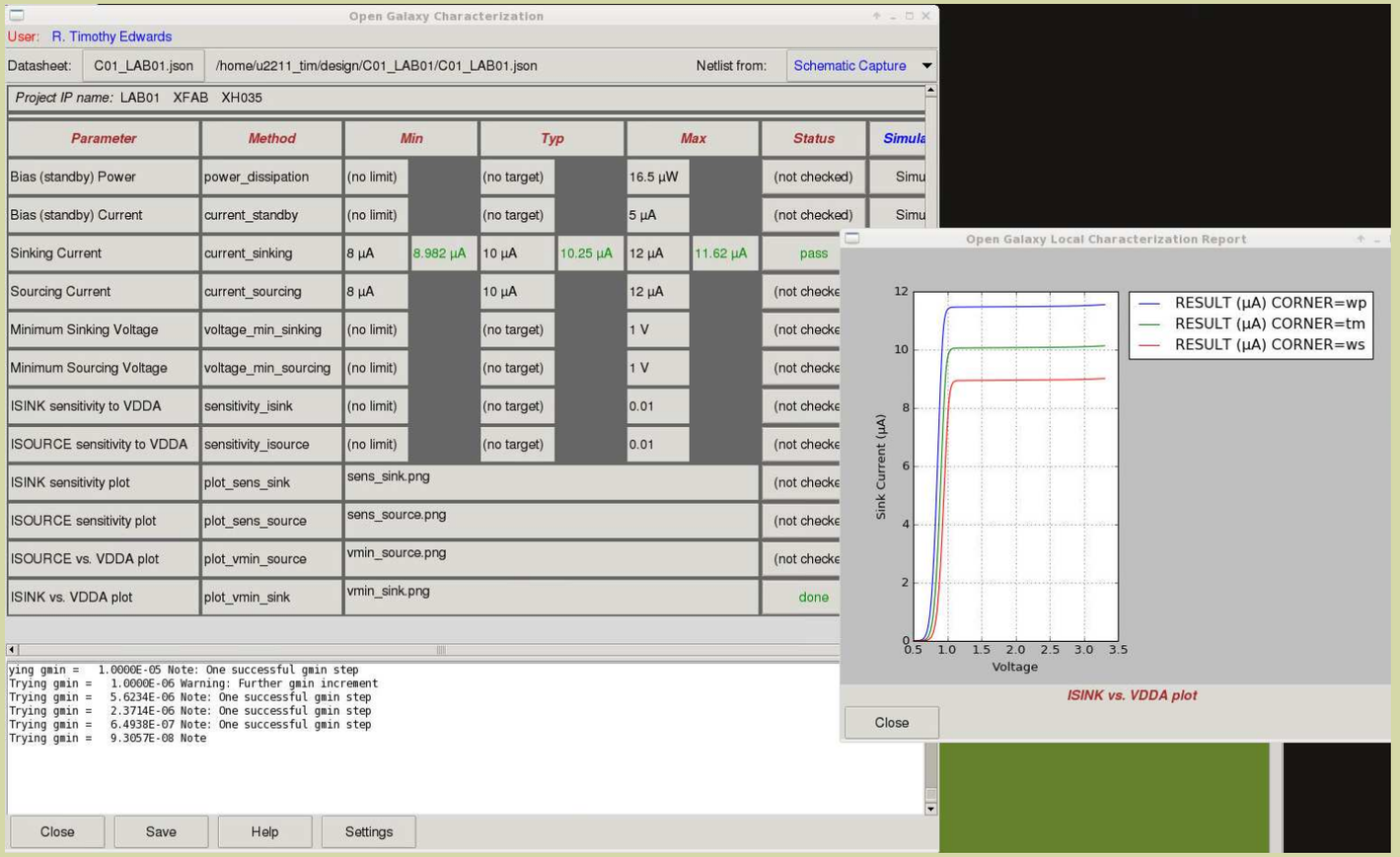
"maglef" view contains pointer to GDS file with start, end byte positions.

GDS write (privileged) manages the GDS data to ensure unique cell names, including all subcell data, etc.

Libraries of IP contain user-readable "maglef" views and read-protected GDS.

Protecting proprietary data in an open environment

Design parameters through simulation



CACE automatic characterization tool on efabless Open Galaxy design platform

efabless.com

Protecting proprietary data in an open environment

Privileged-mode execution of SPICE simulation

Summary: **sudo** ngspice with elevated **group** privilege.

- Only one group can read foundry SPICE models.
- **ngspice** is a wrapper script that launches "exec with-*group* ngspice-inner *arguments*"
- Script **with-***group* runs "sudo -l -n -g *group* asgroup *group* ngspice-inner *arguments*"
- Script **asgroup** limits the applications that can be run (authorization check).
- If authorized (ngspice-inner is authorized), run "**exec ngspice-inner** *arguments*"
- **ngspice-inner** is a script that sets **LD_PRELOAD=**sudo_noexec.so and runs ngspice
- sudo_noexec.so has dummy versions of **execv()** and similar
- See **sudoers** documentation for details.
- Custom version of **ngspice** disables commands to view internal parameters

  Specifically: **show**, **showmod**

Clearly *this is not trivial.*

Protecting proprietary data in an open environment

Uploads and Downloads

- Uploading is less controversial, mainly limited to prevent abuse of the system.
- Currently uploads only allowed in CloudV, considering extension to Open Galaxy.
- Uploads limited to one tarball to be used to seed a new project.
- Downloads prohibited
- Prohibiting uploads and downloads *discourages designers from using the system.*
- Need more flexible use of uploads and downloads using content filtering

  e.g., if it's a verilog file, run it through a parser. If it passes, it's allowed.

  Prevents abuse of the system without limiting users from accessing their own data.

  Certain file types can be prohibited to avoid wholesale download of vendor data, even if it is not strictly proprietary.

Protecting proprietary data in an open environment

User-readable and read-protected IP data

efabless "open galaxy" file system

- /ef/tech/
  - *foundry/*
    - *pdk-name/*
      - .ef-config/
      - libs.ref/
      - libs.tech/

foundry data

- libs.ref/
  - lef/
    - *library-name/*
  - liberty/ — gray area
  - maglef/
  - spi/ — proprietary
  - verilog/ — gray area
  - gds/ — proprietary
  - doc/
  - techLEF/

distributed across all virtual servers

- libs.tech/
  - elec/
  - magic/
    - *pdk-name*.tech
    - *pdk-name*-F.tech — proprietary
  - models/ — proprietary
  - netgen/
  - qflow/

efabless.com

Protecting proprietary data in an open environment

User-readable and read-protected IP data

efablessfile system — user data

- /home/*userid*/design/
  - *project*/ — user data, always readable
  - ip/ — marketplace data pulled by the user
    - *ip-name*/*version*/
      - gds/ — foundry-proprietary data
      - spi/ — user-readable if open source (contains component list)
      - spi-rcx/ — foundry-proprietary data (extracted)
      - spi-stub/ — user-readable (port list only)
      - maglef/ — user-readable (abstract/obfuscated view)
      - verilog/ — user-readable if open source or behavioral

efabless.com

Protecting proprietary data in an open environment

Auto-PDKS

Foundry Data

Libraries

- Digital Standard Cells
- Primitive Devices
- I/O Cells
- Analog IP
- Memory IP

Standard Formats

- Liberty
- LEF
- GDS
- CDL/SPICE
- Verilog
- Documentation

Custom Data

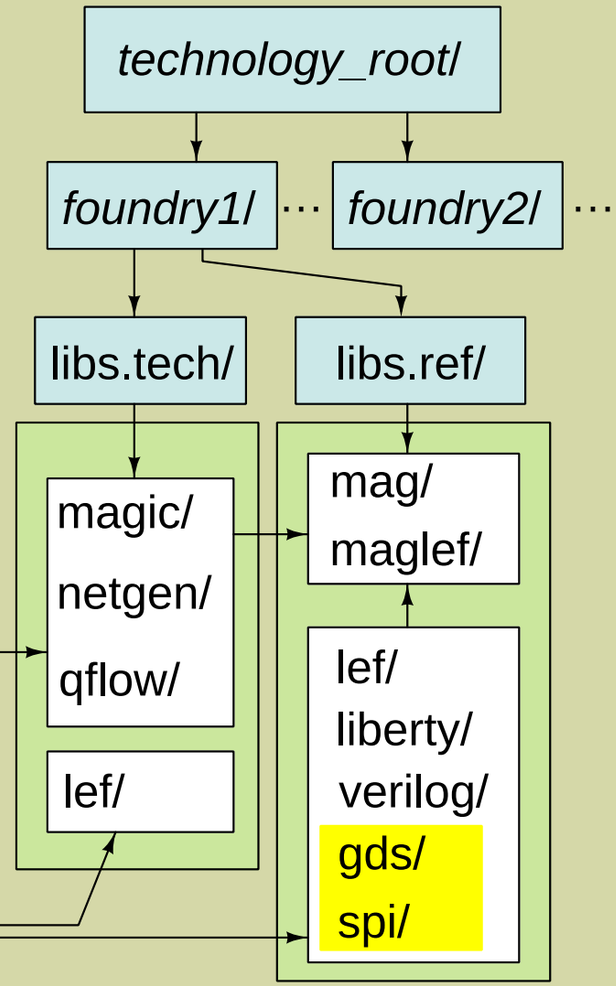
Makefile

- FEOL/BEOL Options

Templates

- Magic techfile (.tech)
- Magic PDK (.tcl)
- Magic startup script (.magicrc)
- Netgen setup file (.tcl)
- Qflow setup file (.sh)
- Graywolf setup file (.par)

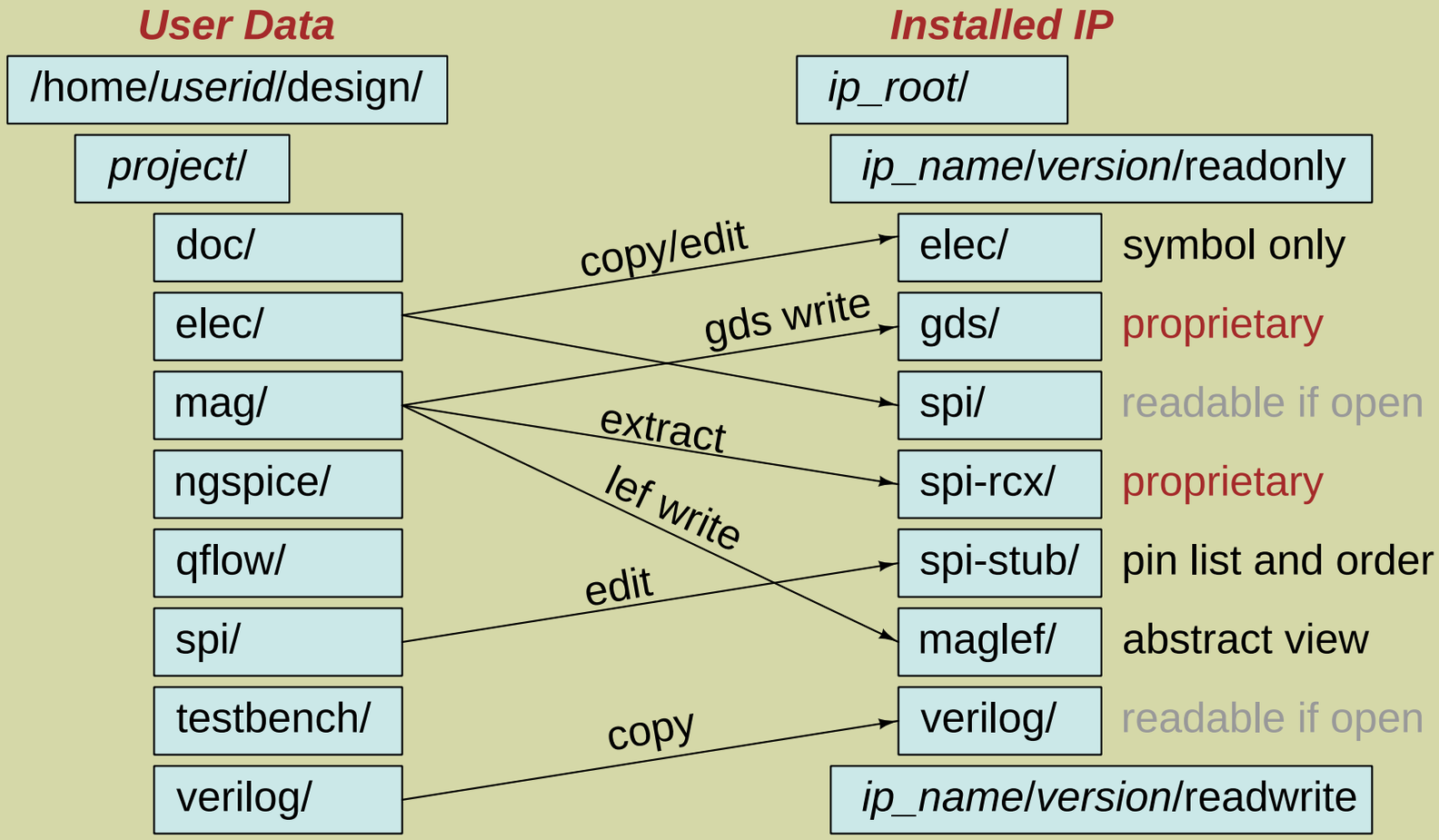
Installed PDKs



efabless.com

Protecting proprietary data in an open environment

IP catalog prep



Available if IP is open source

efabless.com

Protecting proprietary data in an open environment

Conclusions

Open Hardware can coexist with protected foundry IP

Protected foundry IP can coexist with an open design platform

Always better/easier if the foundry is more open and friendly

Foundries seem to be coming around to the idea that open IP is beneficial to them, and it is better to share data than to hide it.