



Introduction to the Digital Design Flow

Rodrigo N. Wuerdig

Joint PhD Student @ UFRGS (Brazil) and KU Leuven (Belgium)

August 28th, 2024



UNIC-CASS Page

Do not miss the educational material at

<https://unic-cass.github.io/>

The screenshot shows a Microsoft Edge browser window displaying the UNIC-CASS GitHub page. The URL in the address bar is <https://unic-cass.github.io>. The page title is "Universalization of IC Design in CASS". A sidebar on the left contains a navigation menu with sections like Home, Course introduction, Environment Setup, Prerequisites, Digital design tools, Analog Mixed Signal Design, Analog Design Flow, and Digital Design Flow. The main content area welcomes visitors to the IEEE Circuits and Systems Society (CASS) UNIC-CASS page and provides links to the UNIC-CASS Program homepage, the Slack workspace, and access training materials. It also includes copyright information and a link to edit the page on GitHub.

InPrivate Home | Universalization of IC Des: + https://unic-cass.github.io

Universalization of IC Design in CASS

Search Universalization of IC Design in CASS

UNIC-CASS Homepage

Home

Course introduction

Environment Setup

2.0 Prerequisites

2.1 Digital design tools on Linux or WSL using Docker

2.2 Analog Mixed Signal Design using docker image

2.3 Analog Mixed Signal Design tools on Linux or WSL using Conda

Analog Design Flow

Digital Design Flow

4.1 Openlane overview

4.2 Caravel test harness

4.3 Design setup using Caravel_user_project

4.4 Running example design from Caravel_user_project

4.5 How to integrate your own design

4.6 Running precheck and getting the design ready for tapeout

4.7 Synthesizing VHDL designs

Welcome to the IEEE Circuits and Systems Society (CASS) UNIC-CASS page!

UNIC-CASS Program homepage

Join the UNIC-CASS Slack Workspace

Access training materials

Copyright © 2024 IEEE CASS

Edit this page on GitHub.

This site uses Just the Docs, a documentation theme for Jekyll.

Join the Open-EDA Community

UNIC-CASS Slack

https://join.slack.com/t/unic-cass/shared_invite/zt-1xxifr0ow-n8dpt0qNBxb4J50g8MEvmw

open-source-silicon Slack

https://join.slack.com/t/open-source-silicon/shared_invite/zt-1zopfd1gk-ul2eSINXB54xN9RCowGa4g

Outline

1. Introduction to the RTL2GDSII Flow

- Logical Synthesis
- Physical Synthesis

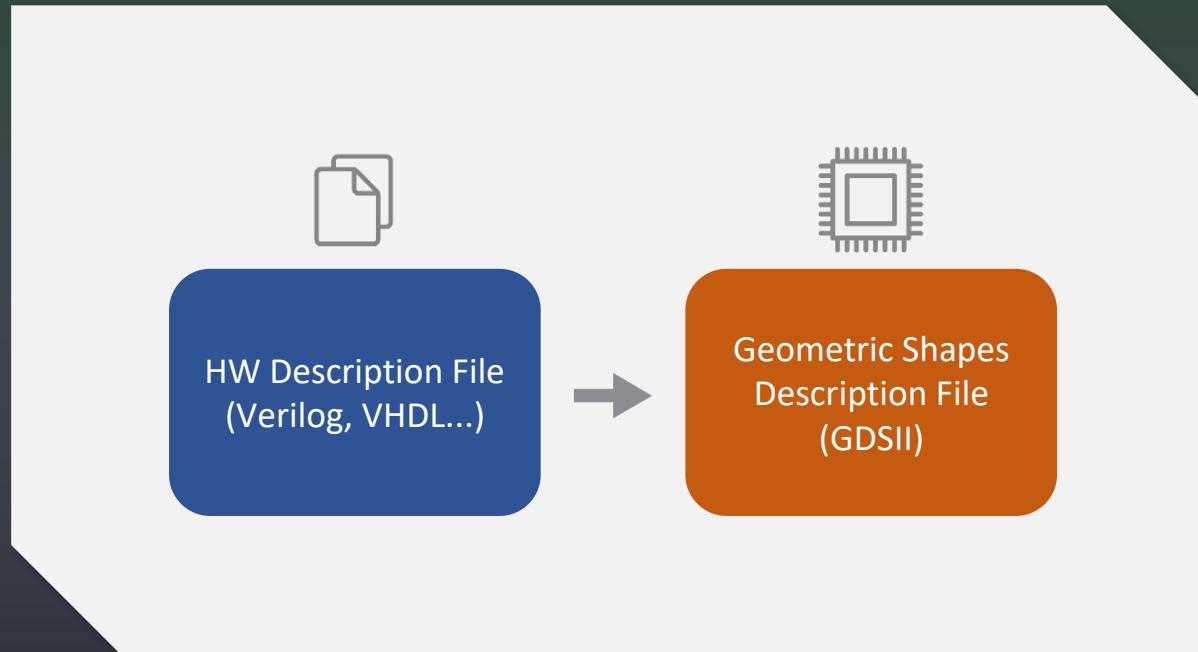
2. Open-PDK Ecosystem

3. Open-EDA Ecosystem

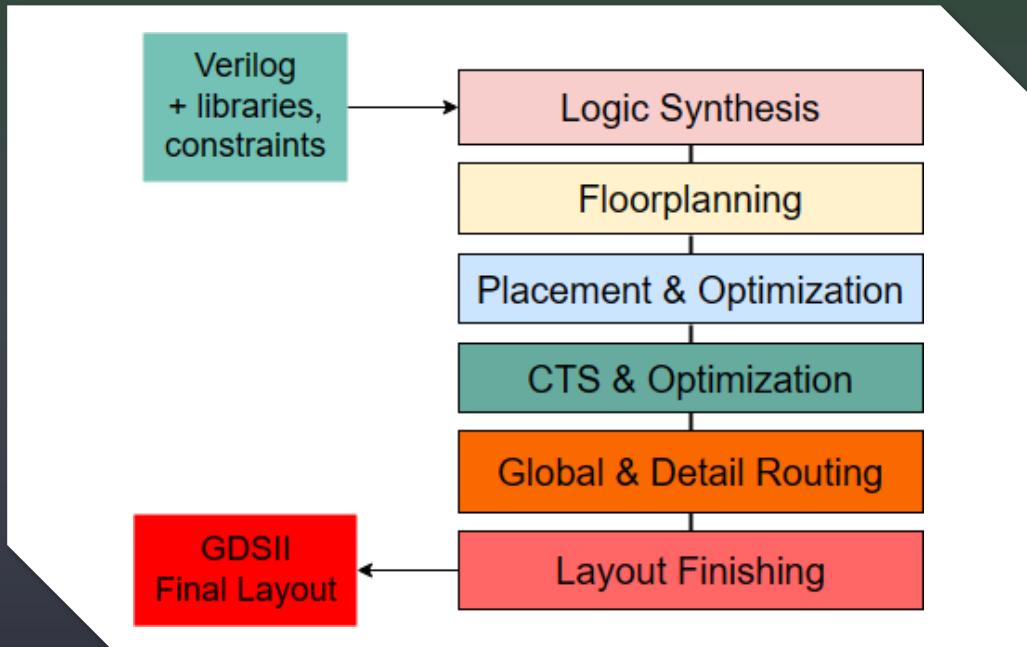
4. Caravel

5. How the top-level integration works

RTL2GDSII



RTL2GDSII

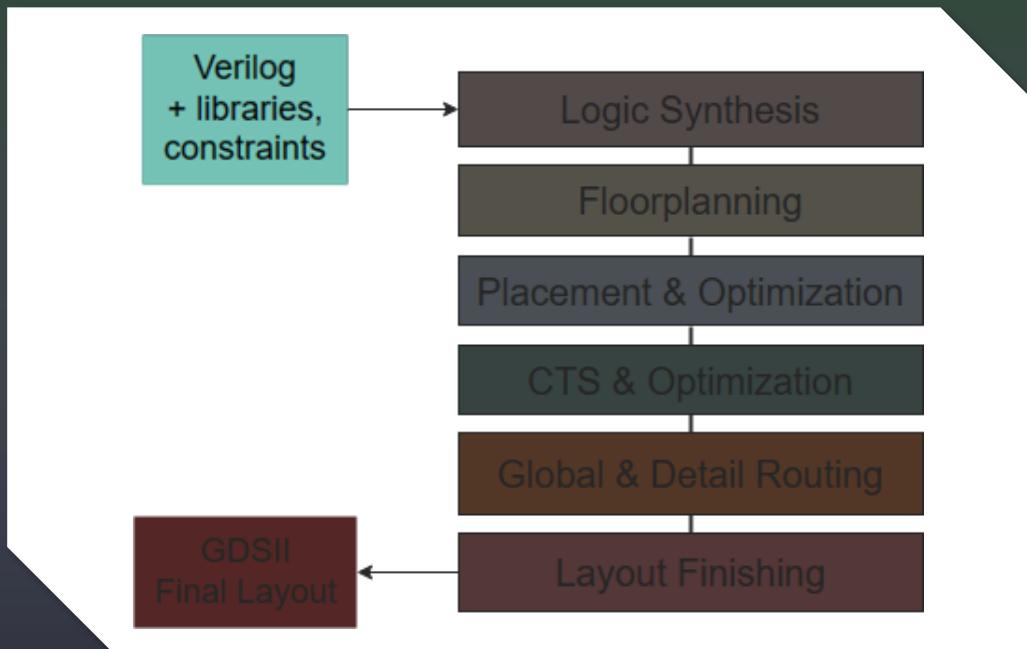


Source: <https://openroad.readthedocs.io/en/latest/main/README.html>

RTL2GDSII

If using VHDL, follow section 4.7:

<https://unic-cass.github.io/training/4.7-synthesis-vhdl-design.html>



Source: <https://openroad.readthedocs.io/en/latest/main/README.html>

Logic Synthesis

Transform a **HW Description** into a “mapped description” / netlist.

```
23 reg [SC_SIZE-1:0] scan_master;
24 reg [SC_SIZE-1:0] scan_slave;
25 wire [SC_SIZE-1:0] scan_next;
26
27 assign scan_next = {scan_data_in, scan_slave[SC_SIZE-1:1]};
28
29 always @ (posedge clk) begin
30   scan_master = scan_next;
31 end
32
33 always @ (negedge clk) begin
34   scan_slave = scan_master;
35 end
36
37 always @ (*)
38 if (scan_load_chip) begin
39   SC_to_the_chip=scan_slave;
40 end
```

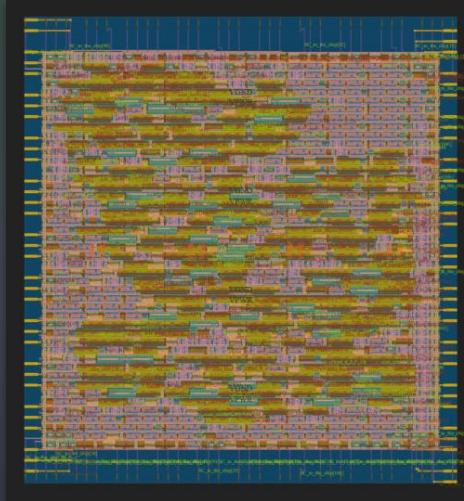


```
4054 .VWN(VWN),
4055 sky130_fd_sc_hd_tapvpwrvgnd_1 TAP_99 (.VGND(VGND),
4056 .VPWR(VPWR));
4057 sky130_fd_sc_hd_buf_1_142_ (.A(clknet_4_9_0_clk),
4058 .VGND(VGND),
4059 .VNB(VGND),
4060 .VPB(VPWR),
4061 .VPWR(VPWR),
4062 .X(_128_));
4063 sky130_fd_sc_hd_buf_1_143_ (.A(clknet_1_1_leaf_128_),
4064 .VGND(VGND),
4065 .VNB(VGND),
4066 .VPB(VPWR),
4067 .VPWR(VPWR),
4068 .X(_129_));
4069 sky130_fd_sc_hd_inv_2_144_9 (.A(clknet_1_1_leaf_129_),
4070 .VGND(VGND),
4071 .VNB(VGND),
```

Physical Synthesis

Transform the mapped netlist into a geometric shapes description (<*.gds>).

```
4654
4655     .VRWV(.VRWV),
4656     sky130_fd_sc_hd_tapvpwrvgnd_1 TAP_99 (.VGND(VGND),
4657     .VPWR(VPWR));
4658     sky130_fd_sc_hd_buf_1 _142_ (.A(clknet_4_9_0_clk),
4659     .VGND(VGND),
4660     .VNB(VGND),
4661     .VPB(VPWR),
4662     .VPWR(VPWR),
4663     .X(_128_));
4664     sky130_fd_sc_hd_buf_1 _143_ (.A(clknet_1_1_leaf_128),
4665     .VGND(VGND),
4666     .VNB(VGND),
4667     .VPB(VPWR),
4668     .VPWR(VPWR),
4669     .X(_129_));
4670     sky130_fd_sc_hd_inv_2 _144_9 (.A(clknet_1_1_leaf_129),
4671     .VGND(VGND),
4672     .VNB(VGND),
```



Open-PDK Ecosystem

Open PDKs

- GlobalFoundries 180
- Skywater 130
- IHP 130 BiCMOS

alpha release:

- SKY90-FD

Predictive PDKs

- ASAP 7
- PTM-MG
- FreePDK45
- ...

Open-PDK Ecosystem

Open PDKs

- GlobalFoundries 180
- **Skywater 130**
- IHP 130 BiCMOS

alpha release:
- SKY90-FD

Predictive PDKs

- ASAP 7
- PTM-MG
- FreePDK45
- ...

Open-EDA Ecosystem

70+ open/free tools for IC Design

Source: <https://semiwiki.com/wikis/industry-wikis/eda-open-source-tools-wiki/>

Frameworks / Software Wrappers

OpenLane

OpenROAD Flow Scripts

SiliconCompiler

...

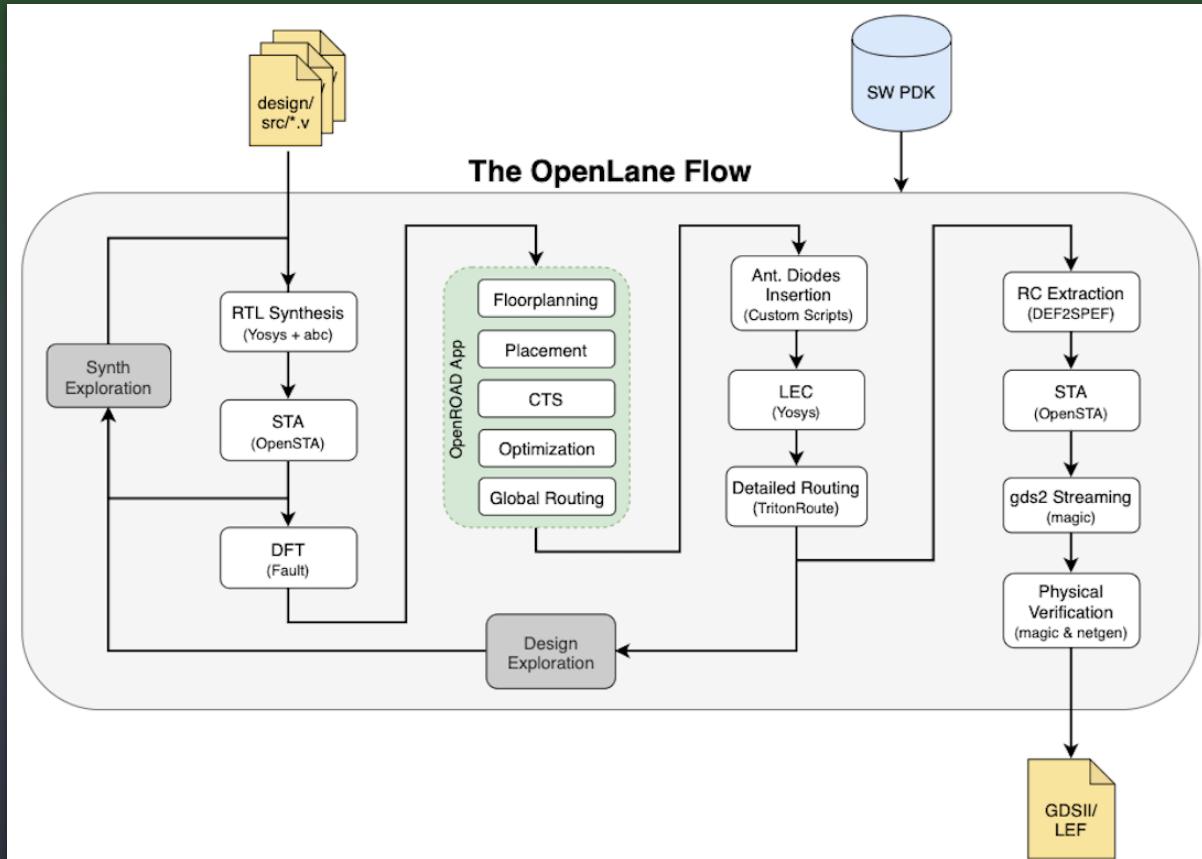
Frameworks / Software Wrappers

OpenLane

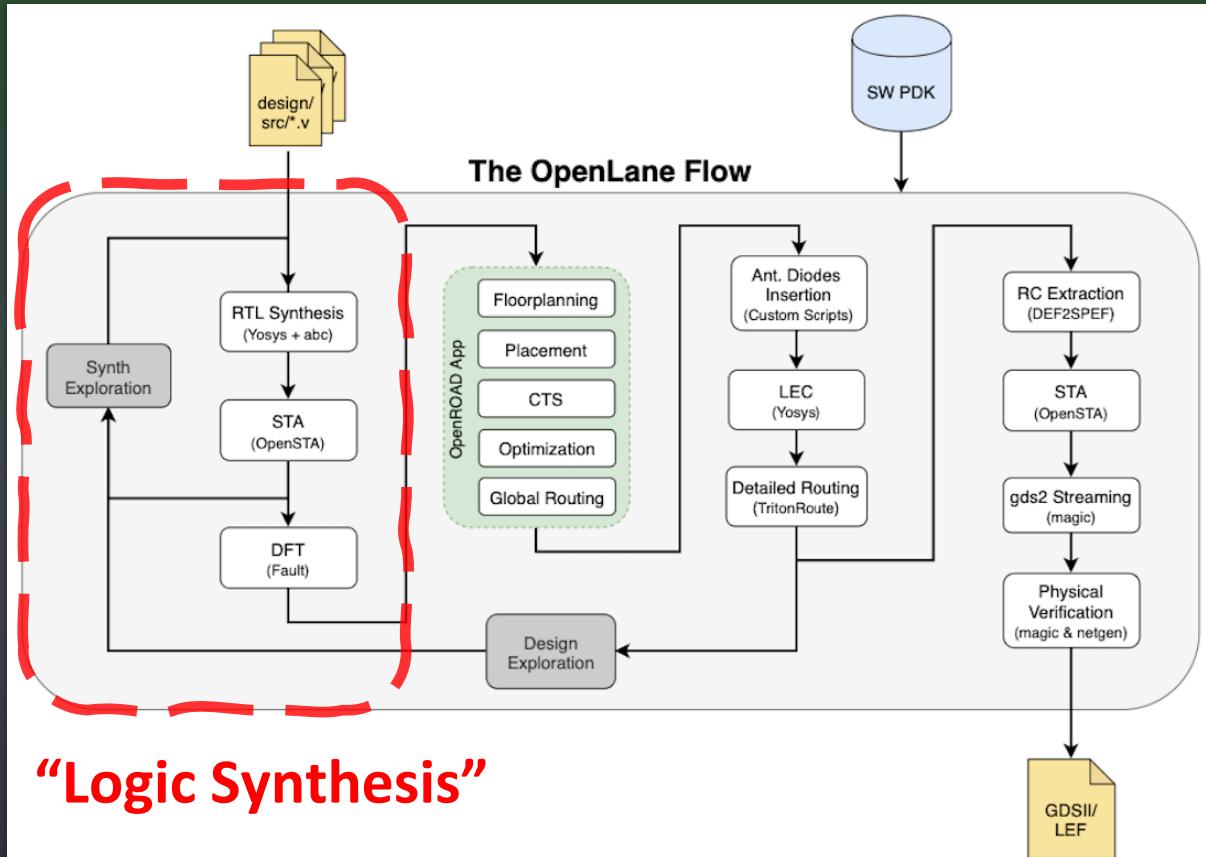
OpenROAD Flow Scripts

SiliconCompiler

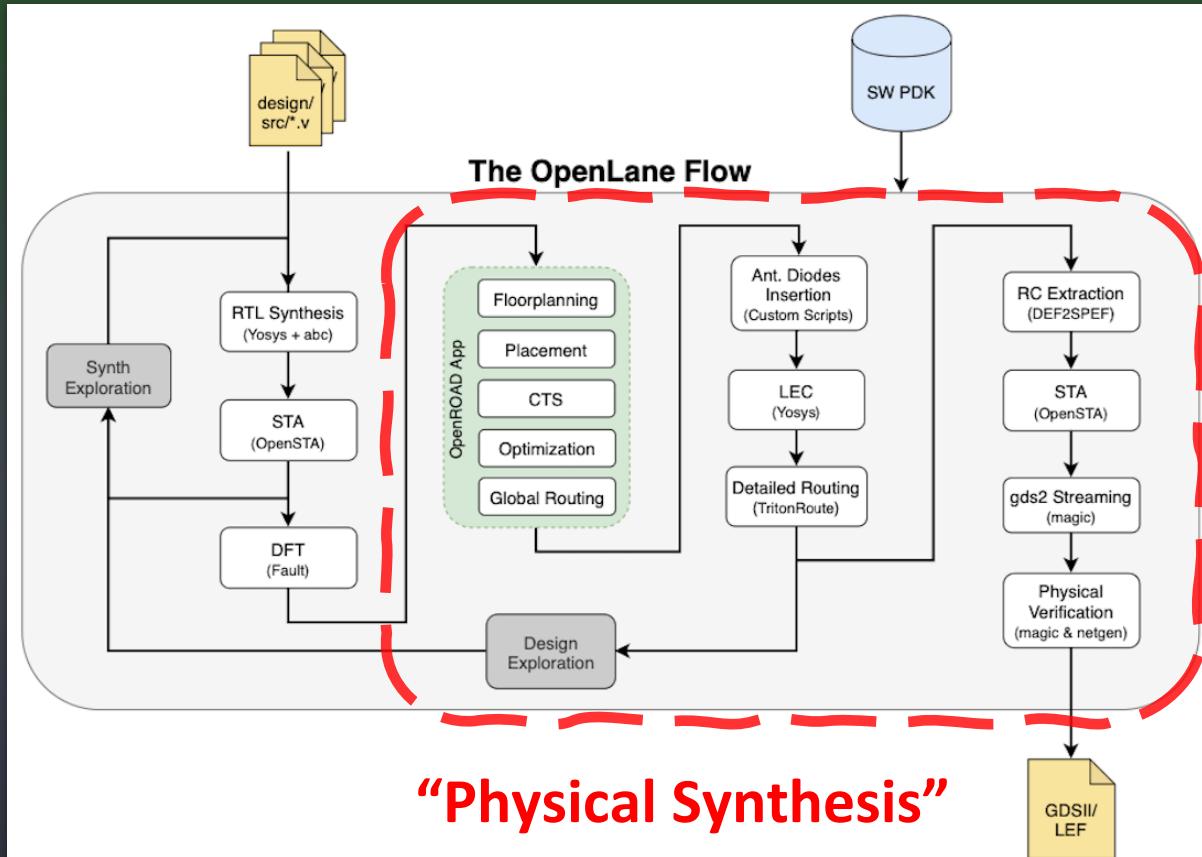
...



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

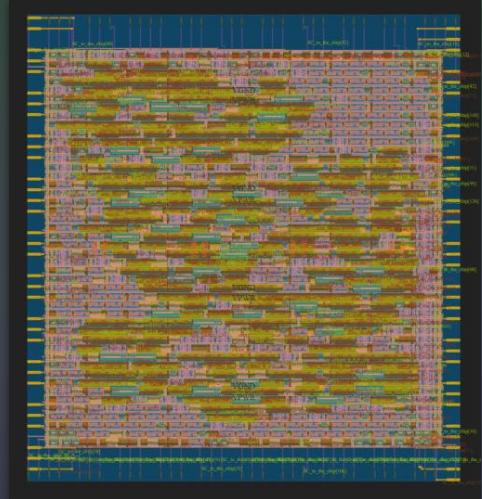


Source: https://openlane.readthedocs.io/en/latest/flow_overview.html



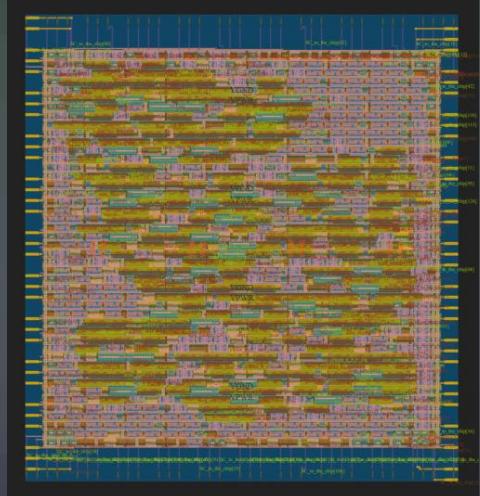
Source: https://openlane.readthedocs.io/en/latest/flow_overview.html

```
23 reg [SC_SIZE-1:0] scan_master;
24 reg [SC_SIZE-1:0] scan_slave;
25 wire [SC_SIZE-1:0] scan_next;
26
27 assign scan_next = {scan_data_in, scan_slave[SC_SIZE-1:1]};
28
29 always @ (posedge clk) begin
30   scan_master = scan_next;
31 end
32
33 always @ (negedge clk) begin
34   scan_slave = scan_master;
35 end
36
37 always @ (*)
38 if (scan_load_chip) begin
39   SC_to_the_chip=scan_slave;
40 end
```

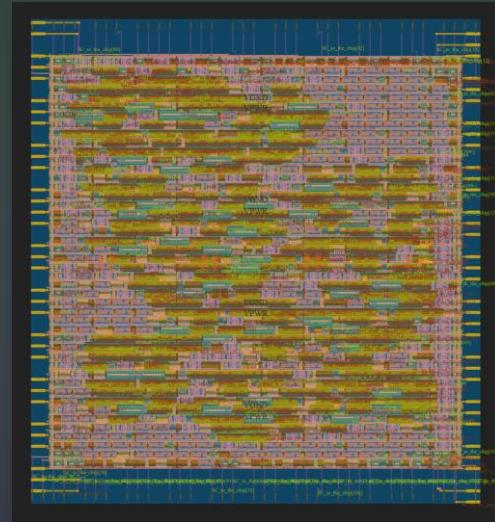
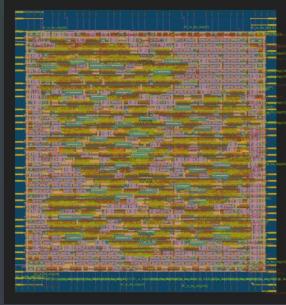


Ok! We can synthesize a block using OpenLane.
But this block lacks context

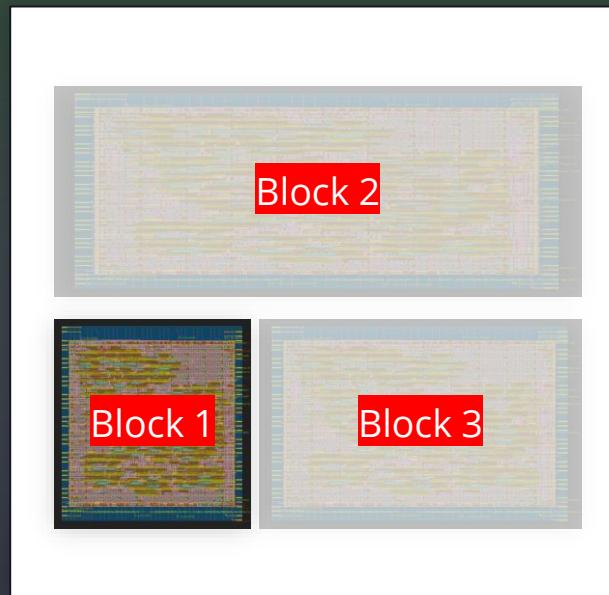
```
23 reg [SC_SIZE-1:0] scan_master;
24 reg [SC_SIZE-1:0] scan_slave;
25 wire [SC_SIZE-1:0] scan_next;
26
27 assign scan_next = {scan_data_in, scan_slave[SC_SIZE-1:1]};
28
29 always @ (posedge clk) begin
30   scan_master = scan_next;
31 end
32
33 always @ (negedge clk) begin
34   scan_slave = scan_master;
35 end
36
37 always @ (*)
38 if (scan_load_chip) begin
39   SC_to_the_chip=scan_slave;
40 end
```



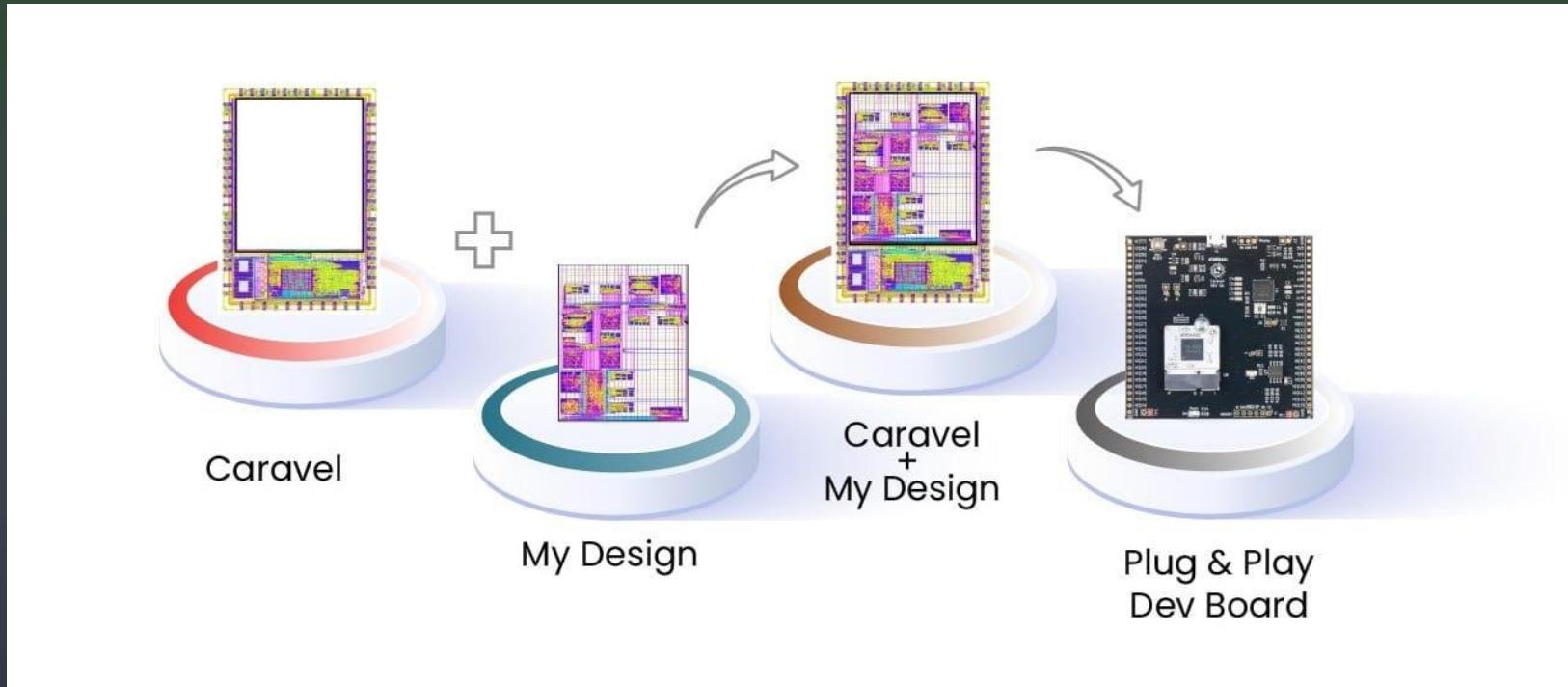
OpenLane gives us freedom for making our block with several shapes and sizes



**But in a real application (ic) this block will be placed under some context
position, size, pin-connection...**



Our context is placing the multiple blocks under the Caravel Wrapper



Source: <https://efabless.com/chipignite>

So it is important to try both:

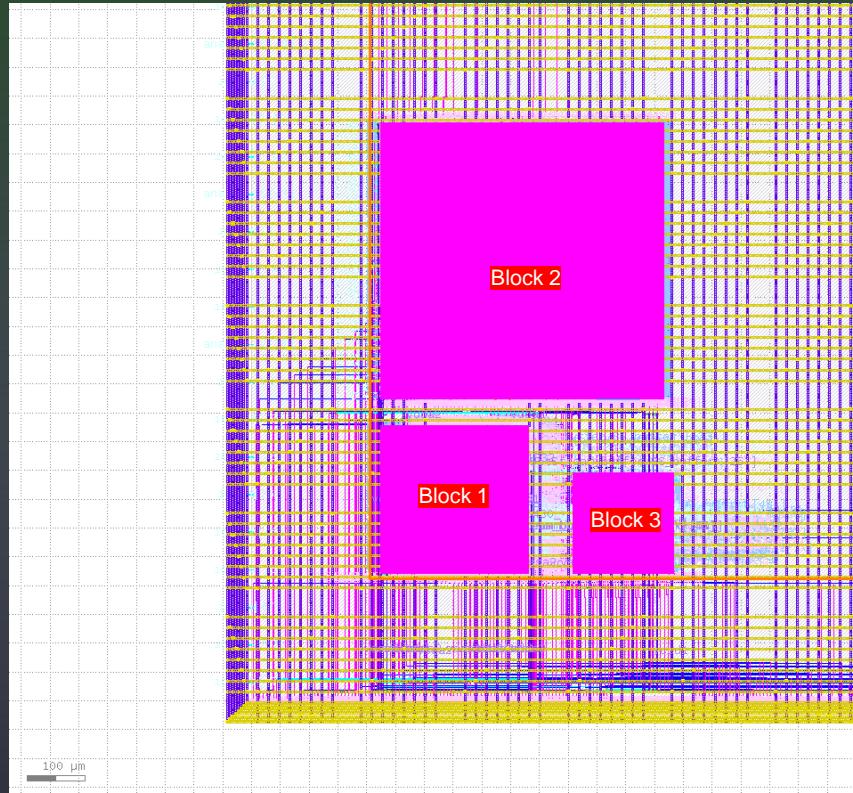
1. The standard OpenLane block synthesis

<https://unic-cass.github.io/training/2.1-digital-design-tool-docker.html>

2. And especially the caravel synthesis flow

<https://unic-cass.github.io/training/4.3-design-setup-caravel-user-project.html>

```
23 reg [SC_SIZE-1:0] scan_master;
24 reg [SC_SIZE-1:0] scan_slave;
25 wire [SC_SIZE-1:0] scan_next;
26
27 assign scan_next = {scan_data_in, scan_slave[SC_SIZE-1:1]};
28
29 always @ (posedge clk) begin
30 scan_master = scan_next;
31 end
32
33 always @ (negedge clk) begin
34 scan_slave = scan_master;
35 end
36
37 always @ (*)
38 if (scan_load_chip) begin
39 SC_to_the_chip=scan_slave;
40 end
```



How the top-level integration works

caravel_user_project Public template

Watch 30 Fork 326

main 32 Branches 40 Tags Go to file Add file Code

jeffdi Merge pull request #368 from efabless/makefile_fix · 05c3eea · 3 weeks ago 752 Commits

Commit	Message	Date
.github	Ci optimization (#330)	9 months ago
def	update user_project_wrapper implementation	last year
docs	Update index.md	3 weeks ago
gds	update user_project_wrapper implementation	last year
lef	update user_project_wrapper implementation	last year
lib	update user_project_wrapper implementation	last year
lvs/user_project_wrapper	fixed path in config	last year
mag	update user_project_wrapper implementation	last year
maglef	update user_project_wrapper implementation	last year
openlane	updated docker mounts to include ~/ipm	last month
sdc	update user_project_wrapper implementation	last year
signoff	Merge branch 'main' into qol	last year
spf	update user_project_wrapper implementation	last year
spi/lvs	update user_proj_example implementation	last year
verilog	Merge pull request #320 from efabless/mattvenn-patch-5	10 months ago

How the top-level integration works

caravel_user_project / openlane / ↗	
 marwaneltoukhy	updated docker mounts to include ~/ipm ✘
Name	Last commit message
..	
user_proj_example	update user_proj_example implementation
user_project_wrapper	update user_project_wrapper implementation
.gitignore	Example of a full run of user_project_wrapper
Makefile	updated docker mounts to include ~/ipm

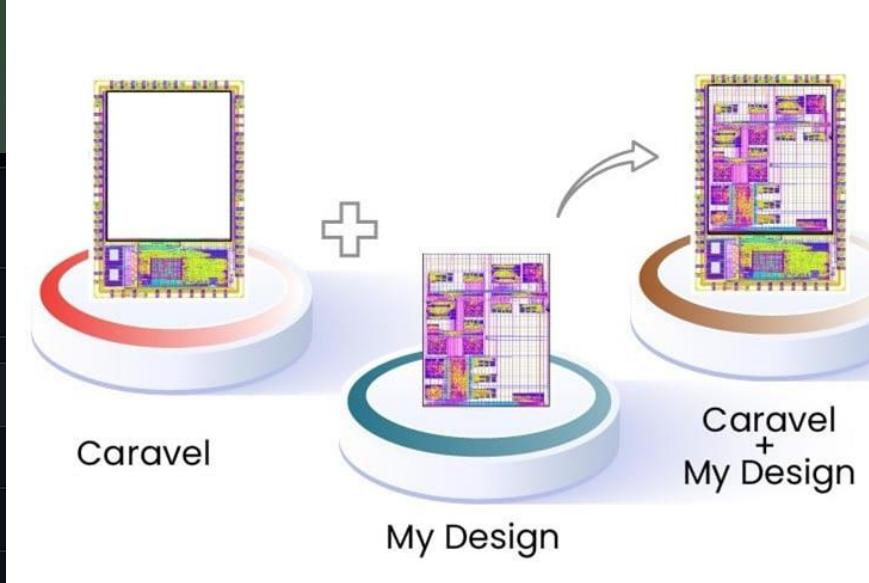
How the top-level integration works

caravel_user_project / openlane / ⌂

 marwaneltoukhy updated docker mounts to include ~/.ipm

Name
..
user_proj_example
user_project_wrapper
.gitignore
Makefile

update user_project_wrapper implementation
Example of a full run of user_project_wrapper
updated docker mounts to include ~/.ipm



How the top-level integration works

caravel_user_project / openlane / ⌂

marwaneltoukhy updated docker mounts to include ~/ipm

Name

..

user_proj_example

user_project_wrapper

.gitignore

Makefile

The diagram illustrates the integration process. It shows two circular stages: 'Caravel' and 'Caravel + My Design'. The 'Caravel' stage contains a small chip icon. An arrow points from this stage to a larger, more complex chip icon labeled 'My Design', which is highlighted with a red box. A plus sign (+) is positioned between the two stages, indicating the combination of Caravel and My Design.

update_user_project_wrapperImplementation

Example of a full run of user_project_wrapper

updated docker mounts to include ~/ipm

How the top-level integration works

caravel_user_project / openlane / ⌂

marwaneltoukhy updated docker mounts to include ~/ipm

Name

..

user_proj_example

user_project_wrapper

.gitignore

Makefile

Caravel

My Design

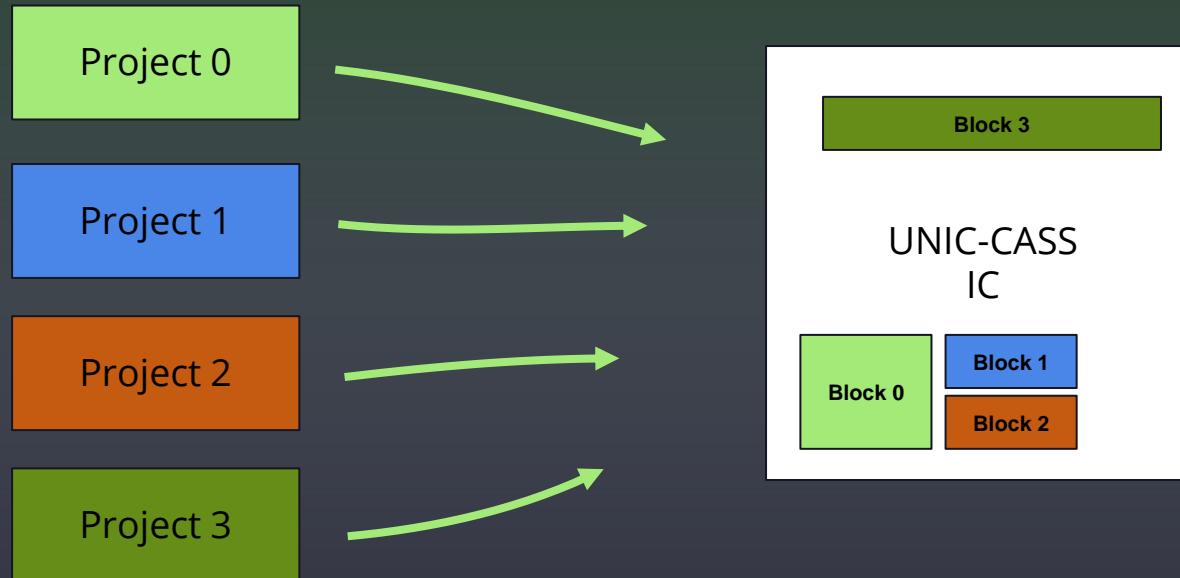
Caravel + My Design

update_user_project_wrapperImplementation

Example of a full run of user_project_wrapper

updated docker mounts to include ~/ipm

How the top-level integration works



How the top-level integration works

Project 0

Project 1

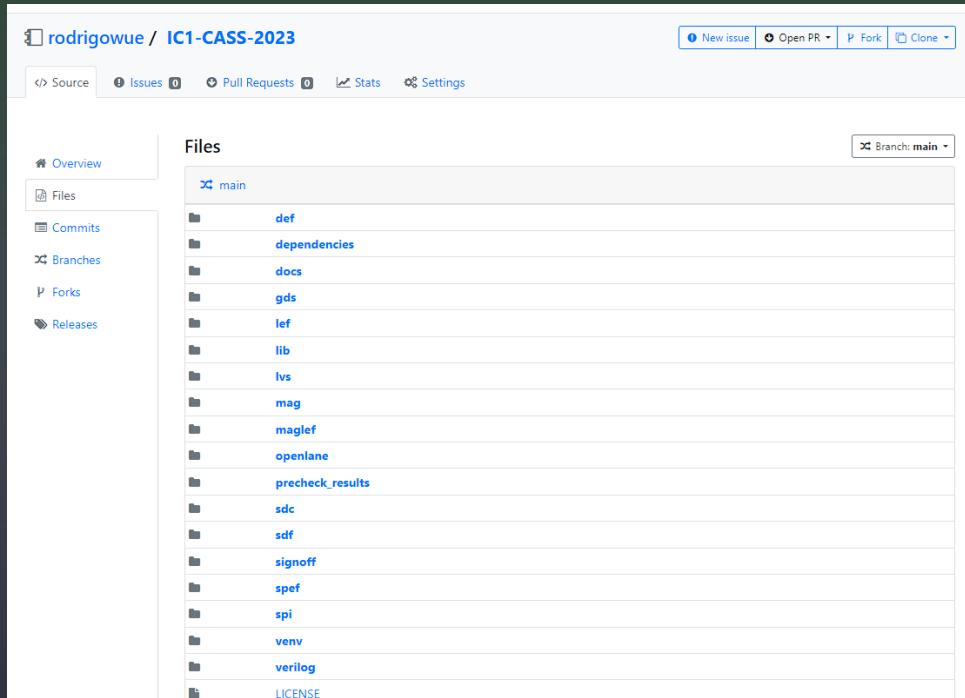
Project 2

Project 3

How the top-level integration works



How the top-level integration works



Efabless git repo for last year IC

A “top-level” git will be created
by me :)

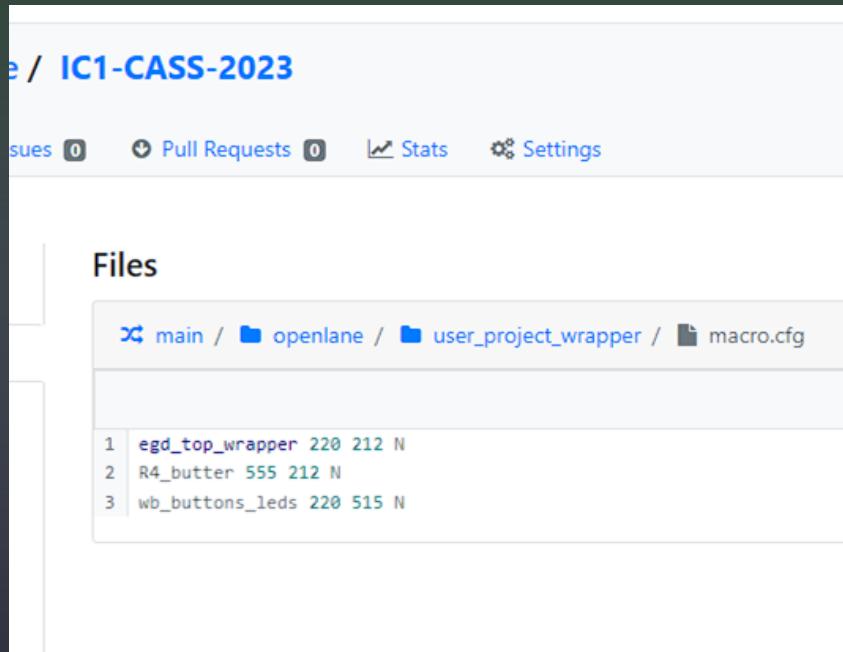
How the top-level integration works

The screenshot shows a GitHub repository interface. The repository name is 'rodrigowue / IC1-CASS-2023'. The 'Files' tab is selected. On the left, there's a sidebar with links: Overview, Files (which is active), Commits, Branches, Forks, and Releases. The main area shows the 'main' branch with a 'main' folder containing an 'openlane' folder. Inside 'openlane', there are several files: 'R4_butter' (labeled 'Block 1'), 'egd_top_wrapper' (labeled 'Block 2'), 'user_project_wrapper' (labeled 'Top-level'), 'wb_buttons_leds' (labeled 'Block 3'), '.gitignore', and 'Makefile'. The 'user_project_wrapper' file is highlighted in red.

Efabless git repo for last year IC

And information from each block composing the wrapper will be inserted there.

How the top-level integration works



Efabless git repo for last year IC

The placement for each block will be defined

How the top-level integration works

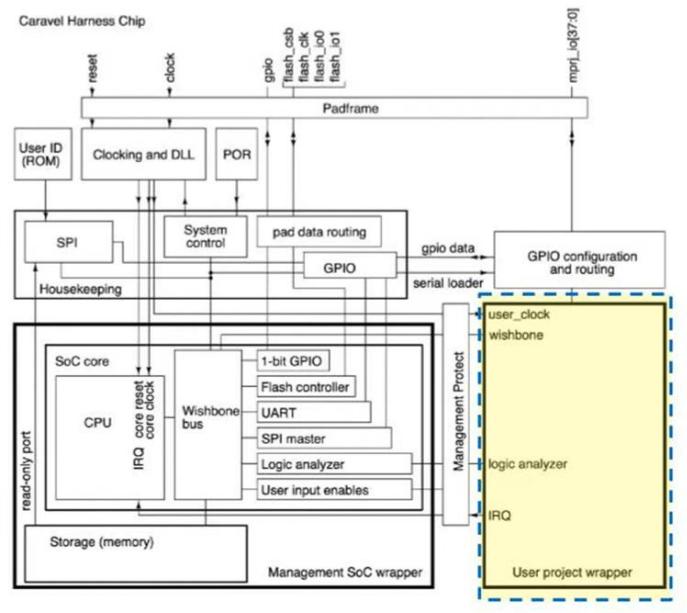
```
80  /*
81  *-----*/
82  /* EGD TOP */
83  /*-----*/
84
85 egd_top_wrapper egd_top_wrapper (
86     `ifdef USE_POWER_PINS
87         .vccd1(vccd1), // User area 1 1.8V power
88         .vssd1(vssd1), // User area 1 digital ground
89     `endif
90
91     // Wishbone Slave ports
92     .wb_clk_i(wb_clk_i),
93
94     // LA Signals
95     // Inputs to egd_top_wrapper
96     .la_data_in_65(la_data_in[65]),
97     .la_data_in_58_43(la_data_in[58:43]),
98     .la_data_in_60_59(la_data_in[60:59]),
99     // Outputs to egd_top_wrapper
100    .la_data_out_23_16(la_data_out[23:16]),
101    .la_data_out_26_24(la_data_out[26:24]),
102    .la_data_out_30_27(la_data_out[30:27])
103 );
104
105
106 /*-----*/
107 /* R4 Butterfly */
108 /*-----*/
109
110 R4_butter R4_butter(
111     `ifdef USE_POWER_PINS
112         .vccd1(vccd1), // User area 1 1.8V power
113         .vssd1(vssd1), // User area 1 digital ground
114     `endif
115
116     .xr0(la_data_in[11:8]),
117     .xr1(la_data_in[15:12]),
118     .xr2(la_data_in[19:16]),
119     .xr3(la_data_in[23:20]),
```

The blocks will be also logically connected to the user_project_wrapper to the assigned pinout. By their instantiation in the wrapper Verilog.

Think about the context! How do you want to connect it with the outside world?!

<https://unic-cass.github.io/training/6.1-design-for-tapeout-Caravel-overview.html>

- 10 mm² silicon area
- 38 User's I/O Pads
- User I/O Pad Configuration
 - FW running on the Management SoC
 - Housekeeping SPI
 - Integration-time configuration
- Access to the Management SoC wishbone bus
- Clocking and Resetting Signals
- 128 R/W Logic Probes



Thanks, any questions? :)

feel free to reach me out

rnwuerdig@inf.ufrgs.br

