Tutorial: Implementing a Macro for Caravel

> **Note**
>
> If you do not have prior experience with OpenLane, please make sure to go through the Getting Started: Newcomers tutorial first.

AES stands for Advanced Encryption Standard which is a symmetric encryption algorithm widely used across the globe to secure data. It operates on blocks of data using keys of 128, 192, or 256 bits to encrypt and decrypt information, providing a high level of security and efficiency for electronic data protection. In this tutorial, we are going to harden an `AES` core and have it as a Caravel User Project macro to serve as an accelerator for the chip Caravel.

> **About Caravel**
>
> The Efabless Caravel chip is a ready-to-use test harness for creating designs with the Google/Skywater 130nm Open PDK. The Caravel harness comprises of base functions supporting IO, power, and configuration as well as drop-in modules for a management SoC core, and an approximately 3000um x 3600um open project area for the placement of user IP blocks.
>
> See Caravel's documentation for more information.

Creating your own project repository

1. Start by creating a new repository from the Caravel user project OpenLane 2 template. Let's call it `caravel_aes_accelerator`.
2. Open a terminal and clone your repository as follows:

```
$ git clone git@github.com:<github_user_name>/caravel_aes_accelerator.git ~/
```

RTL integration

We begin by using the open-source RTLs for AES by Joachim Strömbergson and adding a Wishbone bus wrapper for Caravel. Since the RTL from secworks provides a generic memory interface, we only need to add the `ack`, `write_enable`, and `read_enable` logic to the Wishbone wrapper.

1. Clone the `secworks/aes` Git repository

```
$ git clone git@github.com:secworks/aes.git ~/secworks_aes
```

2. Create the Verilog file `~/caravel_aes_accelerator/verilog/rtl/aes_wb_wrapper.v` and add the Wishbone wrapper to the RTL:

aes_wb_wrapper.v

3. Instantiate the `aes_wb_wrapper` in the `user_project_wrapper` Verilog file under `~/caravel_aes_accelerator/verilog/rtl`

user_project_wrapper.v

> **See also**
>
> Check out Caravel User Project's documentation on Verilog Integration for information about the changes that were done to the RTL.

Hardening strategies

There are 3 options for implementing a Caravel User Project design using

OpenLane:

1. `Macro-First Hardening`: Harden the user macro(s) initially and incorporate them into the user project wrapper without top-level standard cells. Ideal for smaller designs, as this approach significantly reduces Placement and Routing (PnR) and signoff time.
2. `Full-Wrapper Flattening`: Merge the user macro(s) with the user_project_wrapper, covering the entire wrapper area. While this method demands more time and iterations for PnR and signoff, it ultimately enhances performance, making it suitable for designs requiring the full wrapper area.
3. `Top-Level Integration`: Place the user macro(s) within the wrapper alongside standard cells at the top level. This method is typically chosen to introduce buffering at the top level, fitting scenarios where such an approach is necessary.

> **See also**
>
> See the Caravel Integration & Power Routing document for more information about these options.

1. Macro-First Hardening strategy

We will start with the first hardening option. That entails:

1. Hardening the `aes` core with the Wishbone wrapper as a macro
2. Hardening the `user_project_wrapper` with the `aes` macro hardened in the previous step

...the latter of which, will be integrated into the Caravel harness.

AES Wishbone Wrapper Hardening

Configuration

1. Create a design directory to add our source files to:

```
$ mkdir -p ~/caravel_aes_accelerator/openlane/aes_wb_wrapper
```

2. Create the file `~/caravel_aes_accelerator/openlane/aes_wb_wrapper/config.json` and add the following simple configuration to it

```
{
    "DESIGN_NAME": "aes_wb_wrapper",
    "FP_PDN_MULTILAYER": false,
    "CLOCK_PORT": "wb_clk_i",
    "CLOCK_PERIOD": 25,
    "VERILOG_FILES": [
        "dir::../../../secworks_aes/src/rtl/aes.v",
        "dir::../../../secworks_aes/src/rtl/aes_core.v",
        "dir::../../../secworks_aes/src/rtl/aes_decipher_block.v",
        "dir::../../../secworks_aes/src/rtl/aes_encipher_block.v",
        "dir::../../../secworks_aes/src/rtl/aes_inv_sbox.v",
        "dir::../../../secworks_aes/src/rtl/aes_key_mem.v",
        "dir::../../../secworks_aes/src/rtl/aes_sbox.v",
        "dir::../../verilog/rtl/aes_wb_wrapper.v"
    ],
    "FP_CORE_UTIL": 40
}
```

This is a basic configuration file which has only these variables:

- `DESIGN_NAME`: the name of the design, which is equal to the name of the top module in Verilog.
- `CLOCK_PORT`: the name of the clock port in said top module.
- `CLOCK_PERIOD`: the period of the primary clock port in nanoseconds, used to determine the chip frequency. Generally, the lowest you can get away with is the best. $\mathrm{f} = 1/(\texttt{CLOCK\_PERIOD}ns) = 1/(25\mathrm{ns}) = 25\mathrm{MHz}$
- `VERILOG_FILES`: List of input Verilog files.
- `FP_CORE_UTIL`: The core utilization. Typical values for the core utilization range from 25% to 60%. 40% is a good starting value - we can adjust it later if we need to (i.e. one of the tools complains.)
- `FP_PDN_MULTILAYER`: We set this to `false` as we are hardening a chip for integration into Caravel. You may review Power Distribution Networks for more information on this.

Running the flow

To harden macros with OpenLane, we use the default flow, `Classic`.

Let's try running the flow from OpenLane:

```
[nix-shell:~]$ openlane ~/caravel_aes_accelerator/openlane/aes_wb_wrapper/conf
```

Tip

Double-checking: are you inside a `nix-shell`? Your terminal prompt should look like this:

```
[nix-shell:~]$
```

If not, enter the following command in your terminal:

```
$ nix-shell --pure ~/openlane2/shell.nix
```

The flow will finish successfully in ~20 minutes (depending on the speed of your computer) and we will see:


```
Flow complete.
```

Viewing the layout

To open the final GDSII layout run this command:

```
[nix-shell:~/openlane2]$ openlane --last-run --flow openinklayout ~/caravel_ae
```

This opens KLayout and you should be able to see the following:

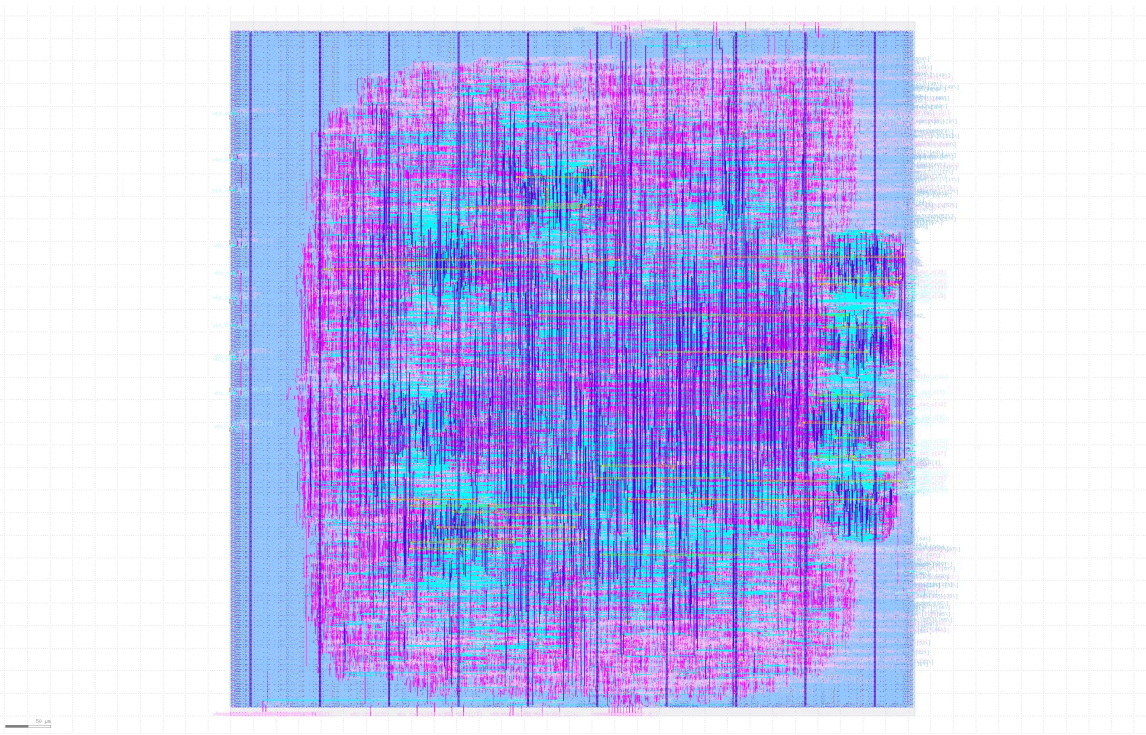

Fig. 26 Final layout of aes_wb_wrapper

Tip

You can control the visible layers in KLayout by double-clicking on the layers you want to hide/unhide. In this figure, the layers `areaid.lowTapDensity` and `areaid.standardc` were hidden to view the layout more clearly.

Checking the reports

You'll find that a run directory (named something like `runs/RUN_2024_02_05_16_46_01`) was created when you ran OpenLane. Under this physical views will be located. It is always a good idea to review all logs and

reports for all the steps in your run. However, in this guide, we will only review the main signoff reports from a couple of steps.

> **Tip**
>
> The names of step directories are constructed as follows:
>
> `{ordinal}-{step_id_slugified}`
>
> ...where `ordinal` is a counter showing in what order a step was run, and `step_id_slugified` is, broadly, the step's ID converted to lowercase and dots replaced with dashes.

`OpenROAD.CheckAntennas`

There are 2 `OpenROAD.CheckAntennas` steps. One after `OpenROAD.GlobalRouting` and the other after `OpenROAD.DetailedRouting`. We are interested in the one after `OpenROAD.DetailedRouting` as this is the final antenna check.

> **See also**
>
> For more information about antenna violations, check this section again in the newcomers' guide.

Inside the step directory of `OpenROAD.CheckAntennas`, there is a `reports` directory that contains two files; the full antenna check report from `OpenROAD` and a summary table of antenna violations:

Partial/Required	Required	Partial	Net
8.43	400.00	3373.21	net337
4.06	400.00	1624.33	_06003_
3.84	400.00	1534.48	net40
3.68	400.00	1471.09	_09365_
3.51	400.00	1402.54	aes.core.dec_block.block_w0_reg[12\]
3.33	400.00	1330.55	_13932_
3.33	400.00	1330.55	_13932_

⋮

As seen in the report, there are around 110 antenna violations that have high ratios up to 8. It is recommended to fix all antenna violations with ratios higher than 3 and the higher the ratio `Partial/Required` the more severely

it might affect the chip. In order to fix those antenna violations, one or more of the following solutions can be applied:

1. Increase the number of iterations for antenna repair using `GRT_ANTENNA_ITERS`. The default value is `3`. We can increase it to `10` by adding this to our `config.json` file.

```
    "GRT_ANTENNA_ITERS": 10,
```

2. Increase the margin for antenna repair using `GRT_ANTENNA_MARGIN`. The default value is `10`. We can increase it to `15`.

```
    "GRT_ANTENNA_MARGIN": 15,
```

3. Enable heuristic diode insertion using `RUN_HEURISTIC_DIODE_INSERTION`:

```
    "RUN_HEURISTIC_DIODE_INSERTION": true,
```

4. Constrain the max wire length (in µm) using `DESIGN_REPAIR_MAX_WIRE_LENGTH`.

```
    "DESIGN_REPAIR_MAX_WIRE_LENGTH": 800,
```

5. Optimize the global placement for minimum wire length using `PL_WIRE_LENGTH_COEF`.

```
    "PL_WIRE_LENGTH_COEF": 0.05,
```

`OpenROAD.STAPostPNR`

Under `xx-openroad-stapostpnr` there should be a file called `summary.rpt`:

Corner/Group	Hold Worst Slack	Reg to Reg Paths	Hold TNS	Hold Violations	of wh to Reg
Overall	0.1045	0.1045	0.0000	0	0
nom_tt_025C…	0.3111	0.3111	0.0000	0	0
nom_ss_100C…	0.8728	0.8728	0.0000	0	0
nom_ff_n40C…	0.1058	0.1058	0.0000	0	0
min_tt_025C…	0.3098	0.3098	0.0000	0	0
min_ss_100C…	0.8712	0.8712	0.0000	0	0
min_ff_n40C…	0.1045	0.1045	0.0000	0	0
max_tt_025C…	0.3131	0.3131	0.0000	0	0
max_ss_100C…	0.8762	0.8762	0.0000	0	0
max_ff_n40C…	0.1073	0.1073	0.0000	0	0

As seen in the report, there are no hold or setup violations. There are only Max Cap and Max Slew violations. To see the violations:

1. Open the report `checks` under `xx-openroad-stapostpnr/max_ss_100C_1v60` since this corner has the highest number of Max Cap and Max Slew violations.
2. Search for `max slew` and you will find the violations listed as follows:

```
Pin                                        Limit        Slew       Slack
------------------------------------------------------------------------
_31022_/B1                                 0.750000    2.000525   -1.250525 (VIOLA
_34084_/A2                                 0.750000    1.999526   -1.249526 (VIOLA
_31021_/Y                                  0.750000    1.998896   -1.248896 (VIOLA
_29818_/B                                  0.750000    1.815682   -1.065682 (VIOLA
_32128_/A2                                 0.750000    1.815665   -1.065665 (VIOLA
_30041_/A                                  0.750000    1.815654   -1.065654 (VIOLA
_29817_/Y                                  0.750000    1.814748   -1.064748 (VIOLA
wire109/A                                  0.750000    1.773218   -1.023218 (VIOLA
_21294_/Y                                  0.750000    1.773215   -1.023215 (VIOLA
wire91/A                                   0.750000    1.683392   -0.933392 (VIOLA
⋮
```

In order to fix the maximum slew/cap violations, **one or more** of the following solutions can be applied:

1. Relax the `MAX_TRANSITION_CONSTRAINT` to 1.5ns as tl
   sky130 lib files

```
    "MAX_TRANSITION_CONSTRAINT": 1.5,
```

2. Increase the slew/cap repair margins using `DESIGN_REPAIR_MAX_SLEW_PCT` and `DESIGN_REPAIR_MAX_CAP_PCT`. The default value is 20%. We can increase it to 30%:

```
    "DESIGN_REPAIR_MAX_SLEW_PCT": 30,
    "DESIGN_REPAIR_MAX_CAP_PCT": 30,
```

3. Change the default timing corner using `DEFAULT_CORNER` for the corner with the most violations which will be `max_ss_100C_1v60` in our case:

```
    "DEFAULT_CORNER": "max_ss_100C_1v60",
```

4. Enable post-global routing design optimizations using `RUN_POST_GRT_DESIGN_REPAIR`:

```
    "RUN_POST_GRT_DESIGN_REPAIR": true,
```

5. Most importantly, it is recommended to use design-specific SDC files for your design using the `PNR_SDC_FILE` and `SIGNOFF_SDC_FILE` variables.

```
    "PNR_SDC_FILE": "dir::cons.sdc",
    "SIGNOFF_SDC_FILE": "dir::cons.sdc",
```

Magic.DRC

Under the directory `xx-magic-drc`, you will find a file named `reports/drc.rpt` that summarizes the DRC violations reported by Magic. The design is DRC clean so the report will look like this:

```
aes_wb_wrapper
----------------------------------------
[INFO] COUNT: 0
[INFO] Should be divided by 3 or 4
```

`KLayout.DRC`

Under the directory `xx-klayout-drc`, you will find a file named `violations.json` that summarizes the DRC violations reported by KLayout. The design is DRC clean so the report will look like this with `"total": 0` at the end:

```
{
  ⋮
  "total": 0
}
```

`Netgen.LVS`

Under the directory `xx-netgen-lvs`, you will find a file named `lvs.rpt` that summarizes the LVS violations reported by netgen. The design is LVS clean so the last part of the report will look like this:

```
Cell pin lists are equivalent.
Device classes aes_wb_wrapper and aes_wb_wrapper are equivalent.

Final result: Circuits match uniquely.
```

Re-running the flow with a modified configuration

To fix the previous issues in the implementation, the following was added to the config file:

```
    "GRT_ANTENNA_ITERS": 10,
    "RUN_HEURISTIC_DIODE_INSERTION": true,
    "HEURISTIC_ANTENNA_THRESHOLD": 200,
    "DESIGN_REPAIR_MAX_WIRE_LENGTH": 800,
    "DEFAULT_CORNER": "max_ss_100C_1v60",
    "RUN_POST_GRT_DESIGN_REPAIR": true,
    "PNR_SDC_FILE": "dir::pnr.sdc",
    "SIGNOFF_SDC_FILE": "dir::signoff.sdc"
```

...and the following 2 constraints files `pnr.sdc` and `s`
`~/caravel_aes_accelerator/openlane/aes_wb_wrapper/`:

pnr.sdc

signoff.sdc

The `Design Constraints` part has to do with the design itself. The `Retrieved Constraints` part is retrieved from the Caravel chip boundary constraints with the `user_project_wrapper`. These constraints can be found here. The PnR constraints file has more aggressive constraints than the signoff one, this is done to accommodate the gap between the optimization tool estimation of parasitics and the final extractions on the layout.

> **See also**
>
> For the most comprehensive guide available on making SDC files, we recommend this excellent book by Sridhar Gangadharan and Sanjay Churiwala:
>
> Constraining Designs for Synthesis and Timing Analysis: A Practical Guide to Synopsys Design Constraints (SDC)

So, the final `config.json` is as follows:

```
{
    "DESIGN_NAME": "aes_wb_wrapper",
    "FP_PDN_MULTILAYER": false,
    "CLOCK_PORT": "wb_clk_i",
    "CLOCK_PERIOD": 25,
    "VERILOG_FILES": [
        "dir::../../../secworks_aes/src/rtl/aes.v",
        "dir::../../../secworks_aes/src/rtl/aes_core.v",
        "dir::../../../secworks_aes/src/rtl/aes_decipher_block.v",
        "dir::../../../secworks_aes/src/rtl/aes_encipher_block.v",
        "dir::../../../secworks_aes/src/rtl/aes_inv_sbox.v",
        "dir::../../../secworks_aes/src/rtl/aes_key_mem.v",
        "dir::../../../secworks_aes/src/rtl/aes_sbox.v",
        "dir::../../verilog/rtl/aes_wb_wrapper.v"
    ],
    "FP_CORE_UTIL": 40,
    "GRT_ANTENNA_ITERS": 10,
    "RUN_HEURISTIC_DIODE_INSERTION": true,
    "HEURISTIC_ANTENNA_THRESHOLD": 200,
    "DESIGN_REPAIR_MAX_WIRE_LENGTH": 800,
    "DEFAULT_CORNER": "max_ss_100C_1v60",
    "RUN_POST_GRT_DESIGN_REPAIR": true,
    "PNR_SDC_FILE": "dir::pnr.sdc",
    "SIGNOFF_SDC_FILE": "dir::signoff.sdc"
}
```

Now let's try re-running the flow:

```
[nix-shell:~/openlane2]$ openlane ~/caravel_aes_accelerator/openlane/aes_wb_wr
```

Re-checking the reports

Now, the antenna report under `xx-openroad-checkantennas-1/reports/antenna_summary.rpt` has much less violations:

Note

The number of antenna violations may vary wildly depending on the configuration variables AND environment variables (such as the operating system) as the detailed router's initial heuristic.

On macOS, around a dozen violations are returned.

Partial/Required	Required	Partial	Net	Pin	Layer
1.24	400.00	495.45	_12832_	_27256_/B2	met3
1.23	400.00	493.06	_13527_	_26076_/A2	met3

Also, the STA report at `xx-openroad-stapostpnr/summary.rpt` should have no issues:

Corner/G…	Hold Worst Slack	Reg to Reg Paths	Hold TNS	Hold Violatio…	of which Reg to Reg	Setup Worst Slack
Overall	0.1601	0.1601	0.0000	0	0	4.466
nom_tt_0…	0.2973	0.3282	0.0000	0	0	10.41
nom_ss_1…	0.7765	0.7803	0.0000	0	0	4.641
nom_ff_n…	0.1650	0.1650	0.0000	0	0	11.13
min_tt_0…	0.3215	0.3215	0.0000	0	0	10.56
min_ss_1…	0.7678	0.7678	0.0000	0	0	4.846
min_ff_n…	0.1601	0.1601	0.0000	0	0	11.14
max_tt_0…	0.2648	0.3330	0.0000	0	0	10.25
max_ss_1…	0.7331	0.7868	0.0000	0	0	4.466
max_ff_n…	0.1656	0.1656	0.0000	0	0	11.10

Saving the views

To save the views, run the following script with the following arguments in order:

1. The directory of the project
2. The macro name
3. The successful run tag

```
[nix-shell:~/openlane2]$ bash ~/caravel_aes_accelerator/openlane/copy_views.sh
```


This will copy the physical views of the macro in the folder.

User Project Wrapper Hardening

Configuration

The User Project Wrapper is a macro inside the Caravel chip which will include our design. To be able to use any design as a Caravel User Project, it has to match the footprint that Caravel is expecting. Also, the top-level design Caravel is expecting any Caravel User Project to have the IO pins at specific locations and with specific dimensions. So, we need a fixed floorplan, fixed I/Os pin shapes and locations, and fixed power rings. The fixed configuration section can be found at the end of the configurations file `openlane/user_project_wrapper/config.json`:

```
    "//": "Fixed configurations for Caravel. You should NOT edit this section",
    "DESIGN_NAME": "user_project_wrapper",
    "FP_SIZING": "absolute",
    "DIE_AREA": [0, 0, 2920, 3520],
    "FP_DEF_TEMPLATE": "dir::fixed_dont_change/user_project_wrapper.def",
    "VDD_NETS": [
        "vccd1",
        "vccd2",
        "vdda1",
        "vdda2"
    ],
    "GND_NETS": [
        "vssd1",
        "vssd2",
        "vssa1",
        "vssa2"
    ],
    "FP_PDN_CORE_RING": 1,
    "FP_PDN_CORE_RING_VWIDTH": 3.1,
    "FP_PDN_CORE_RING_HWIDTH": 3.1,
    "FP_PDN_CORE_RING_VOFFSET": 12.45,
    "FP_PDN_CORE_RING_HOFFSET": 12.45,
    "FP_PDN_CORE_RING_VSPACING": 1.7,
    "FP_PDN_CORE_RING_HSPACING": 1.7,
    "CLOCK_PORT": "wb_clk_i",
    "SIGNOFF_SDC_FILE": "dir::signoff.sdc",
    "MAGIC_DEF_LABELS": 0,
    "CLOCK_PERIOD": 25,
    "MAGIC_ZEROIZE_ORIGIN": 0
```

The rest of the configuration file can be edited. Now, We need the following edits for the `openlane/user_project_wrapper/config.json` in order to integrate our macro inside the `user_project_wrapper`:

1. Replace the `user_proj_example` in the `MACROS` variable with our macro. First, we change the physical views to `aes_wb_wrapper`. Second, we can modify the macro location to `[1500, 1500]` to be in the middle of the chip. The new macro variable will be:

```
    "MACROS": {
        "aes_wb_wrapper": {
            "gds": [
                "dir::../../gds/aes_wb_wrapper.gds"
            ],
            "lef": [
                "dir::../../lef/aes_wb_wrapper.lef"
            ],
            "instances": {
                "mprj": {
                    "location": [1500, 1500],
                    "orientation": "N"
                }
            },
            "nl": [
                "dir::../../verilog/gl/aes_wb_wrapper.v"
            ],
            "spef": {
                "min_*": [
                    "dir::../../spef/multicorner/aes_wb_wrapper.min.spef"
                ],
                "nom_*": [
                    "dir::../../spef/multicorner/aes_wb_wrapper.nom.spef"
                ],
                "max_*": [
                    "dir::../../spef/multicorner/aes_wb_wrapper.max.spef"
                ]
            },
            "lib": {
                "*": "dir::../../lib/aes_wb_wrapper.lib"
            }
        }
    },
```


2. Update the power pins in `PDN_MACRO_CONNECTIONS` to the macro power pins

```
    "PDN_MACRO_CONNECTIONS": ["mprj vccd2 vssd2 VPWR VGND"],
```

Note

If we have multiple macros, we can add more entries to the variable `MACROS`.

Running the flow

```
[nix-shell:~/openlane2]$ openlane ~/caravel_aes_accelerator/openlane/user_proje
```

Tip

Double-checking: are you inside a `nix-shell`? Your terminal prompt should look like this:

```
[nix-shell:~/openlane2]$
```

If not, enter the following command in your terminal:

```
$ nix-shell --pure ~/openlane2/shell.nix
```

The flow will finish successfully in ~7 minutes and we will see:

```
Flow complete.
```

Viewing the layout

To open the final GDSII layout run this command:

```
[nix-shell:~/openlane2]$ openlane --last-run --flow openinklayout ~/caravel_aes
```

This opens KLayout and you should be able to see the following:

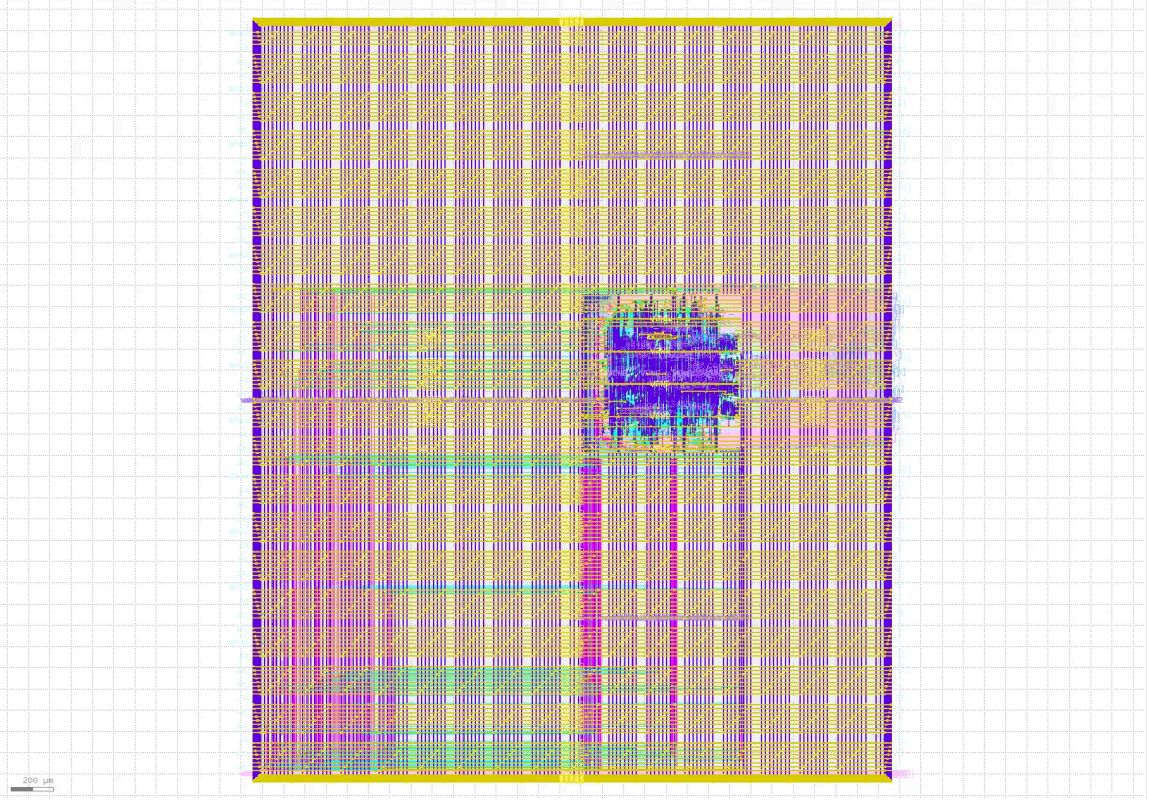

Fig. 27 Final layout of the user_project_wrapper

Tip

You can control the visible layers in KLayout by double-clicking on the layers you want to hide/unhide. In this figure, the layers `areaid.lowTapDensity`, `areaid.diode`, and `areaid.standardc` were hidden to view the layout more clearly.

As seen in the layout, we have our aes macro placed around the middle and if we only show the layers: `prBoundary.boundary`, `met1.drawing`, `met2.drawing`, and `met3.drawing`. We will see long and unnecessary routes because of 2 things:

1. The AES macro is placed very far from its connections. It should be placed at the bottom left corner.
2. The pins of the AES macro should be on the south only.

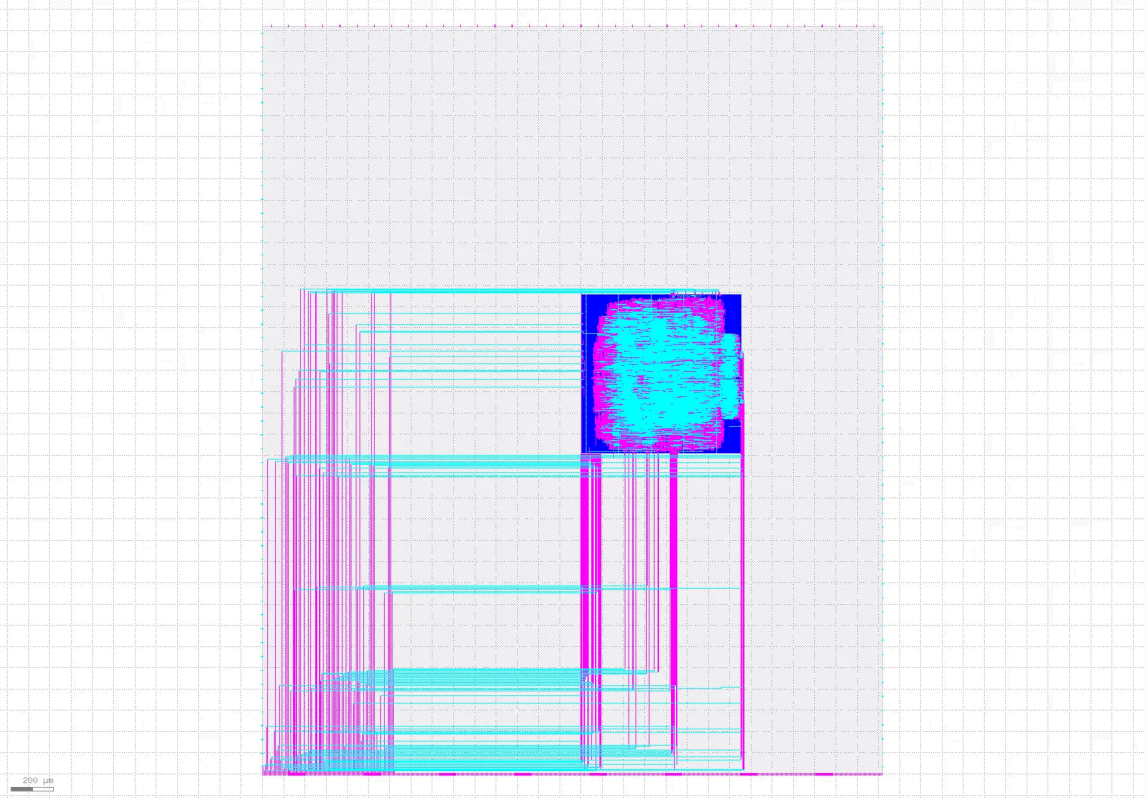

Fig. 28 Long routes in the user_project_wrapper

Checking the reports

`OpenROAD.CheckAntennas`

There should be no antenna violations.

Partial/Required	Required	Partial	Net	Pin	Layer

`OpenROAD.STAPostPNR`

Looking at `xx-openroad-stapostpnr/summary.rpt` and the `Max Slew` section in `xx-openroad-stapostpnr/max_ss_100C_1v60/checks.rpt`, there are max transition violations. If we look at the nets with violations, we v Read the Docs latest long nets we saw in the GDS.

Corner/Group	Hold Worst Slack	Reg to Reg Paths	Hold TNS	Hold Violations
Overall	0.0108	0.0108	0.0000	0
nom_tt_025C_1…	0.1371	0.1596	0.0000	0
nom_ss_100C_1…	0.0830	0.5879	0.0000	0
nom_ff_n40C_1…	0.0120	0.0120	0.0000	0
min_tt_025C_1…	0.1579	0.1579	0.0000	0
min_ss_100C_1…	0.1465	0.5847	0.0000	0
min_ff_n40C_1…	0.0108	0.0108	0.0000	0
max_tt_025C_1…	0.1038	0.1618	0.0000	0
max_ss_100C_1…	0.0268	0.5914	0.0000	0
max_ff_n40C_1…	0.0135	0.0135	0.0000	0

```
Max Slew

Pin                                      Limit        Slew       Slack
-------------------------------------------------------------------------
wbs_dat_o[2]                          1.500000    5.265976   -3.765976 (VIOL
wbs_dat_o[26]                         1.500000    5.214964   -3.714964 (VIOL
wbs_dat_o[25]                         1.500000    4.767642   -3.267642 (VIOL
wbs_dat_o[8]                          1.500000    4.650988   -3.150988 (VIOL
wbs_dat_o[23]                         1.500000    4.362167   -2.862167 (VIOL
wbs_dat_o[29]                         1.500000    3.906245   -2.406245 (VIOL
wbs_dat_o[28]                         1.500000    3.703813   -2.203813 (VIOL
wbs_dat_o[27]                         1.500000    3.586008   -2.086008 (VIOL
wbs_dat_o[31]                         1.500000    3.301759   -1.801759 (VIOL
wbs_dat_o[17]                         1.500000    2.757454   -1.257454 (VIOL
```

Magic.DRC

Under the directory `xx-magic-drc`, you will find a file named `reports/drc.rpt` that summarizes the DRC violations reported by magic. The design is DRC clean so the report will look like this:

```
aes_wb_wrapper
----------------------------------------
[INFO] COUNT: 0
[INFO] Should be divided by 3 or 4
```

`KLayout.DRC`

Under the directory `xx-klayout-drc`, you will find a file named `violations.json` file that summarizes the DRC violations reported by KLayout. The design is DRC clean so the report will look like this with `"total": 0` at the end:

```
{
  ⋮
  "total": 0
}
```

`Netgen.LVS`

Under the directory `xx-netgen-lvs`, you will find a file named `lvs.rpt` that summarizes the LVS violations reported by netgen. The design is LVS clean so the last part of the report will look like this:

```
Cell pin lists are equivalent.
Device classes user_project_wrapper and user_project_wrapper are equivalent.

Final result: Circuits match uniquely.
```

Re-running the flow with a modified configuration

To fix the long routes issue that causes maximum transition violations, 3 things should be done:

1. Create the pin order configuration file for `aes_wb_wrapper` in `openlane/aes_wb_wrapper/pin_order.cfg`:

pin_order.cfg

2. Add the `Odb.CustomIOPlacement::FP_PIN_ORDER_CFG` variable to `openlane/aes_wb_wrapper/config.json`

config.json

3. Update the location of the macro in the `openlane/user_project_wrapper/config.json` to `[10, 20]`

config.json

Now let's re-run the flow for the `aes_wb_wrapper`:

```
[nix-shell:~/openlane2]$ openlane ~/caravel_aes_accelerator/openlane/aes_wb_wr
```

Then, after checking the `aes_wb_wrapper` reports, save the physical views using:

```
[nix-shell:~/openlane2]$ bash ~/caravel_aes_accelerator/openlane/copy_views.sh
```

Then rerun the `user_project_wrapper`

```
[nix-shell:~/openlane2]$ openlane ~/caravel_aes_accelerator/openlane/user_proje
```

Re-checking the layout

To open the final GDSII layout run this command:

```
[nix-shell:~/openlane2]$ openlane --last-run --flow openinklayout ~/caravel_ae
```

Now our macro is placed at the bottom left corner close to the wishbone pins.

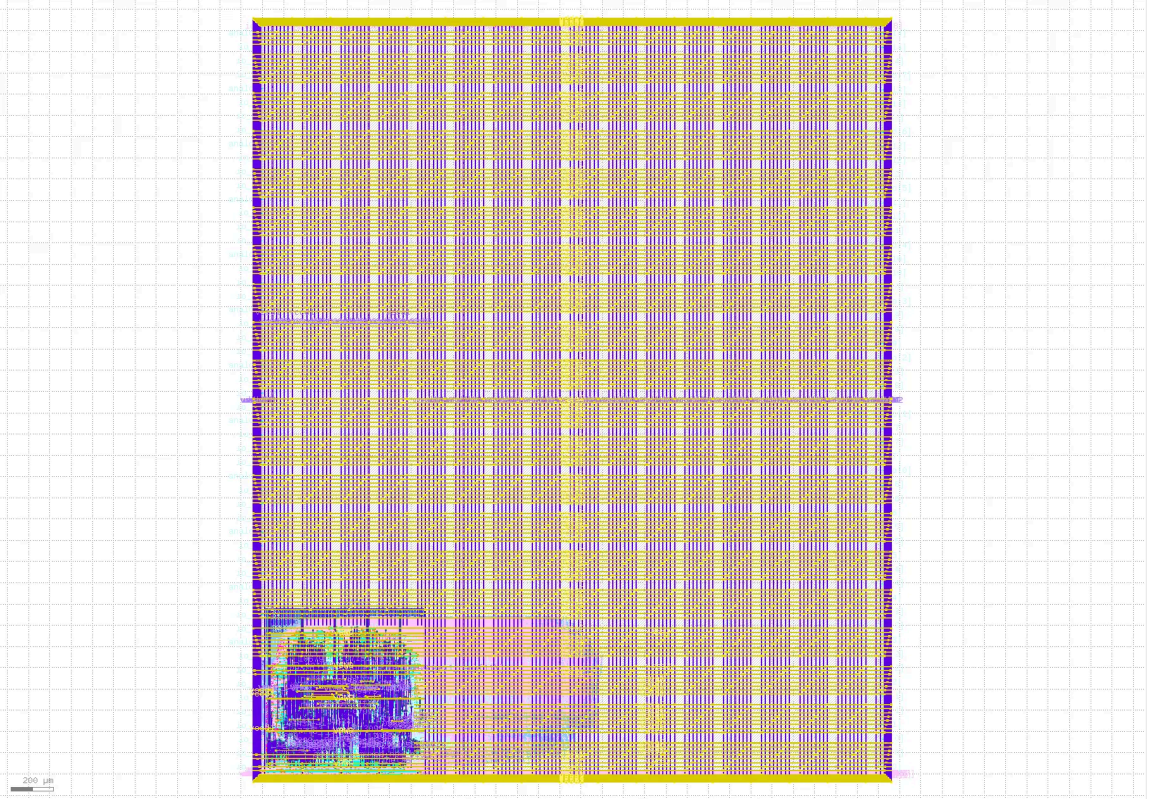

Fig. 29 Final layout of the user_project_wrapper

And if we zoom to the AES macro and view only `prBoundary.boundary`, `met1.drawing`, `met2.drawing`, and `met3.drawing`, there are no long routes anymore.

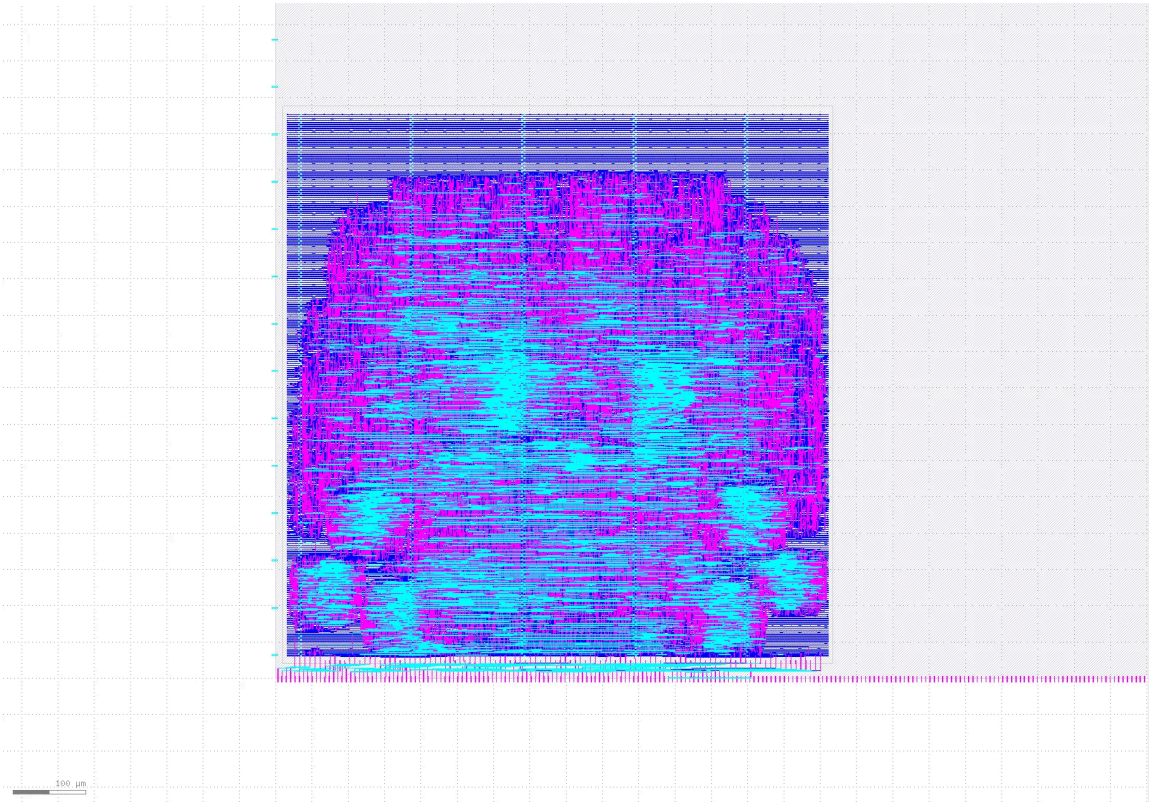

Fig. 30 Shorter routes in the user_project_wrapper

Re-checking the reports

The STA report `xx-openroad-stapostpnr/summary.rpt` now has no issues:

Corner/Group	Hold Worst Slack	Reg to Reg Paths	Hold TNS	Hold Violations
Overall	0.0502	0.0502	0.0000	0
nom_tt_025C_1…	0.2279	0.2279	0.0000	0
nom_ss_100C_1…	0.3832	0.7152	0.0000	0
nom_ff_n40C_1…	0.0519	0.0519	0.0000	0
min_tt_025C_1…	0.2256	0.2256	0.0000	0
min_ss_100C_1…	0.4091	0.7107	0.0000	0
min_ff_n40C_1…	0.0502	0.0502	0.0000	0
max_tt_025C_1…	0.2304	0.2304	0.0000	0
max_ss_100C_1…	0.3418	0.7198	0.0000	0
max_ff_n40C_1…	0.0537	0.0537	0.0000	0

Saving the views

To save the views, run the following script with the following arguments in order:

1. The directory of the project
2. The macro name
3. The successful run tag

```
[nix-shell:~/openlane2]$ bash ~/caravel_aes_accelerator/openlane/copy_views.sh
```

This will copy the physical views of the macro in the specified run to your project folder.

Congrats! Now you have an AES accelerator as a Caravel user project.

2. Full-Wrapper Flattening strategy

In this strategy, we will harden the `user_project_wrapper` with the aes as one large flattened macro.

Configuration

Since we will only harden the `user_project_wrapper`. Only the `openlane/user_project_wrapper/config.json` will be edited. The following edits are needed for the `user_project_wrapper`

1. Add the AES Verilog files

```
    "VERILOG_FILES": [
        "dir::../../../secworks_aes/src/rtl/aes.v",
        "dir::../../../secworks_aes/src/rtl/aes_core.v",
        "dir::../../../secworks_aes/src/rtl/aes_decipher_block.v",
        "dir::../../../secworks_aes/src/rtl/aes_encipher_block.v",
        "dir::../../../secworks_aes/src/rtl/aes_inv_sbox.v",
        "dir::../../../secworks_aes/src/rtl/aes_key_mem.v",
        "dir::../../../secworks_aes/src/rtl/aes_sbox.v",
        "dir::../../verilog/rtl/aes_wb_wrapper.v",
        "dir::../../verilog/rtl/defines.v",
        "dir::../../verilog/rtl/user_project_wrapper.v"
    ],
```

1. Remove the `Macros configurations` section as there will not be any macros
2. Update the hardening strategy part to the flattened version

```
    "//": "Hardening strategy variables (this is for 2-Full-Wrapper Flattening
    "SYNTH_ELABORATE_ONLY": false,
    "RUN_POST_GPL_DESIGN_REPAIR": true,
    "RUN_POST_CTS_RESIZER_TIMING": true,
    "DESIGN_REPAIR_BUFFER_INPUT_PORTS": true,
    "FP_PDN_ENABLE_RAILS": true,
    "RUN_ANTENNA_REPAIR": true,
    "RUN_FILL_INSERTION": true,
    "RUN_TAP_ENDCAP_INSERTION": true,
    "RUN_CTS": true,
    "RUN_IRDROP_REPORT": true,
    "VSRC_LOC_FILES": {
        "vccd1": "dir::vsrc/upw_vccd1_vsrc.loc",
        "vssd1": "dir::vsrc/upw_vssd1_vsrc.loc"
    },
```

Now the full configuration file will be:

config.json

Running the flow

To harden macros with OpenLane, we use the defaul

```
[nix-shell:~/openlane2]$ openlane ~/caravel_aes_accelerator/openlane/user_proje
```

The flow will finish successfully in ~2 hours and we will see:

```
Flow complete.
```

Viewing the layout

To open the final GDSII layout run this command:

```
[nix-shell:~/openlane2]$ openlane --last-run --flow openinklayout ~/caravel_aes
```

Now, we can see that there are STD cells all over the `user_project_wrapper` without any macros. Also, we can see that the logic is clustered in the bottom left corner close to the Wishbone bus.

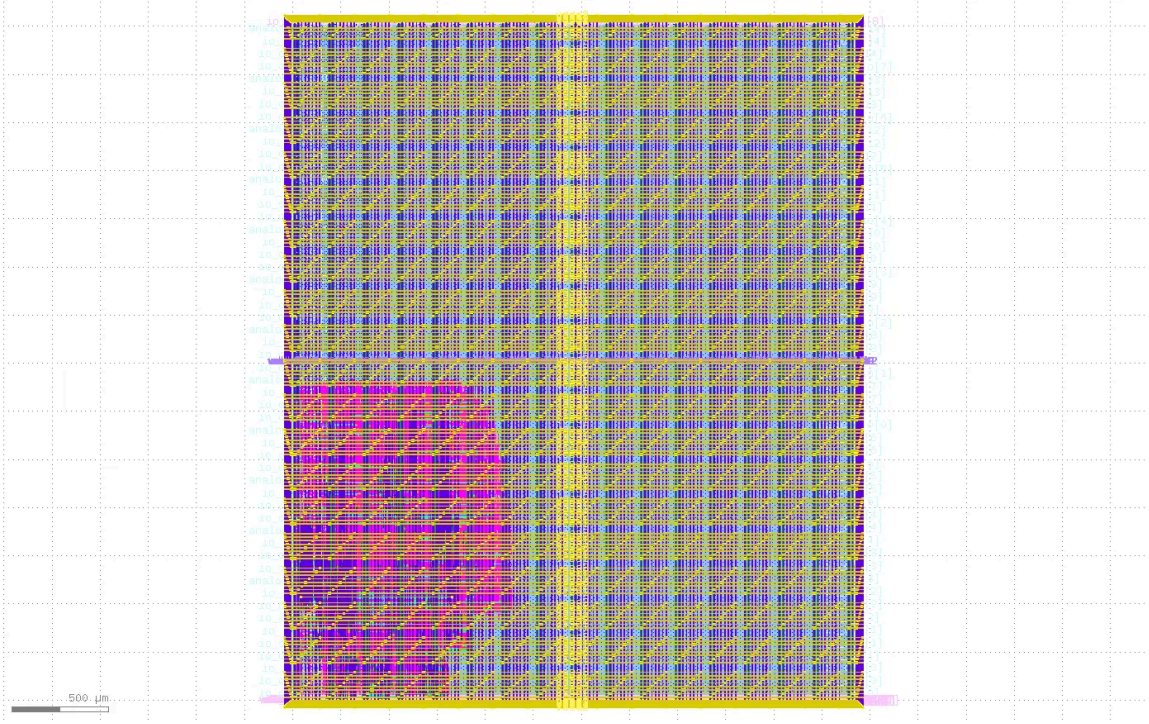

Fig. 31 Final layout of the user_project_wrapper after flattening

Checking the reports

`OpenROAD.CheckAntennas`

There are around 260 antenna violations with ratios up to 7.

Partial/Required	Required	Partial	Net
7.55	400.00	3018.79	_14743_
6.82	400.00	2727.89	net1493
4.68	400.00	1870.11	mprj.aes.core.enc_block.block_w0_reg
3.86	400.00	1544.22	_11561_
3.80	400.00	1519.33	_11868_
3.71	400.00	1485.23	_14658_
3.61	400.00	1443.72	net19
3.45	400.00	1380.11	mprj.aes.core.enc_block.block_w1_reg

We can fix those the same way we did in the AES here.

OpenROAD.STAPostPnR

Looking at `xx-openroad-stapostpnr/summary.rpt`, there are multiple max Slew/Cap violations and 1 hold violation which is not Reg to Reg.

Corner/Group	Hold Worst Slack	Reg to Reg Paths	Hold TNS	Hold Violations	of wh… to Reg
Overall	-0.0108	0.1068	-0.0108	1	0
nom_tt_025C…	0.1236	0.3212	0.0000	0	0
nom_ss_100C…	0.4500	0.8885	0.0000	0	0
nom_ff_n40C…	0.0365	0.1079	0.0000	0	0
min_tt_025C…	0.1726	0.3196	0.0000	0	0
min_ss_100C…	0.5527	0.8798	0.0000	0	0
min_ff_n40C…	0.0791	0.1068	0.0000	0	0
max_tt_025C…	0.0536	0.3234	0.0000	0	0
max_ss_100C…	0.3233	0.8972	0.0000	0	0
max_ff_n40C…	-0.0108	0.1095	-0.0108	1	0

The max Slew/Cap violations can be fixed the same way as in this section. For the hold violation, it is in the `max_ff_n40C_1v95` corner. To investigate the timing path, open the report `xx-openroad-stapostpnr/max_ff_n40C_1v95/min.rpt` and the violation will be in the first timing path.

> **Note**
>
> There might be more hold violations or no violations at all depending on the version of OpenLane being used as this is a violation with a very small negative slack and the results can slightly change with OpenLane or OpenROAD updates.

To fix hold violations, one or more of the following solutions can be applied:

1. Enable post-global routing timing optimizations using `RUN_POST_GRT_RESIZER_TIMING`:

```
    "RUN_POST_GRT_RESIZER_TIMING": true,
```

2. Increase the hold repair margins using `PL_RESIZER_HOLD_SLACK_MARGIN` and `GRT_RESIZER_HOLD_SLACK_MARGIN`. The default values can increase those as follows:

```
    "PL_RESIZER_HOLD_SLACK_MARGIN": 0.2,
    "GRT_RESIZER_HOLD_SLACK_MARGIN": 0.2,
```

3. Change the default timing corner using `DEFAULT_CORNER`:

```
    "DEFAULT_CORNER": "max_tt_025C_1v80",
```

4. Most importantly, it is recommended to use a specific constraint file for your design using `PNR_SDC_FILE` and `SIGNOFF_SDC_FILE`

```
    "PNR_SDC_FILE": "dir::cons.sdc",
    "SIGNOFF_SDC_FILE": "dir::cons.sdc",
```

`Magic.DRC`

Under the directory `xx-magic-drc`, you will find a file named `reports/drc.rpt` that summarizes the DRC violations reported by magic. The design is DRC clean so the report will look like this:

```
aes_wb_wrapper
----------------------------------------
[INFO] COUNT: 0
[INFO] Should be divided by 3 or 4
```

`KLayout.DRC`

Under the directory `xx-klayout-drc`, you will find a file named `violations.json` that summarizes the DRC violations reported by KLayout. The design is DRC clean so the report will look like this with `"total": 0` at the end:

```
{
  ⋮
  "total": 0
}
```

`Netgen.LVS`

Under the directory `xx-netgen-lvs`, you will find a file named `lvs.rpt` that summarizes the LVS violations reported by Netgen. The design is LVS clean so the last part of the report will look like this:

```
Cell pin lists are equivalent.
Device classes user_project_wrapper and user_project_wrapper are equivalent.

Final result: Circuits match uniquely.
```

Re-running the flow with a modified configuration

To fix the previous issues in the implementation, the following was added to the `user_project_wrapper` config file:

```
    "//": "New variables",
    "GRT_ANTENNA_ITERS": 10,
    "RUN_HEURISTIC_DIODE_INSERTION": true,
    "HEURISTIC_ANTENNA_THRESHOLD": 200,
    "DESIGN_REPAIR_MAX_WIRE_LENGTH": 800,
    "PL_WIRE_LENGTH_COEF": 0.05,
    "DEFAULT_CORNER": "max_tt_025C_1v80",
    "RUN_POST_GRT_DESIGN_REPAIR": true,
    "RUN_POST_GRT_RESIZER_TIMING": true,
```

and the following constraints file `pnr.sdc` was created at `~/caravel_aes_accelerator/openlane/user_project_wrapper/`. This file is originally copied from `~/caravel_aes_accelerator/openlane/user_project_wrapper/signoff.sdc` and edited to fix the transition and hold violations:

pnr.sdc

Then, the PnR SDC file path was edited in the JSON file.

```
    "PNR_SDC_FILE": "dir::pnr.sdc",
```

Now the final `~/caravel_aes_accelerator/openlane/user_project_wrapper/config.json` file will be:

config.json

Now let's try re-running the flow:

```
[nix-shell:~/openlane2]$ openlane ~/caravel_aes_accelerator/openlane/user_proj
```

The flow will finish successfully in ~2 hours and we will see:

```
Flow complete.
```

Re-checking the reports

Now, the antenna report under `xx-openroad-checkantennas-1/reports/antenna_summary.rpt` has much less violations:

```
┏━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━┳━━━━━━━━━┳━━━━━━━━━━━
┃ Partial/Required ┃ Required ┃ Partial ┃ Net
┡━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━╇━━━━━━━━━╇━━━━━━━━━━━
│ 2.03             │ 400.00   │ 810.79  │ _08755_
│ 2.00             │ 400.00   │ 798.00  │ _14836_
│ 1.75             │ 400.00   │ 701.33  │ _11832_
│ 1.70             │ 400.00   │ 678.78  │ _11884_
│ 1.62             │ 400.00   │ 647.67  │ net1481
│ 1.61             │ 3130.28  │ 5050.41 │ _14758_
│ 1.40             │ 3130.28  │ 4394.16 │ _14751_
│ 1.39             │ 3130.28  │ 4350.98 │ _14707_
⋮
```

Also, the STA report `xx-openroad-stapostpnr/summary.rpt` has no issues:

latest

Corner/Group	Hold Worst Slack	Reg to Reg Paths	Hold TNS	Hold Violations	of wh: to Reg
Overall	0.0486	0.0486	0.0000	0	0
nom_tt_025C…	0.2274	0.2274	0.0000	0	0
nom_ss_100C…	0.2414	0.7146	0.0000	0	0
nom_ff_n40C…	0.0497	0.0497	0.0000	0	0
min_tt_025C…	0.2253	0.2253	0.0000	0	0
min_ss_100C…	0.3307	0.7108	0.0000	0	0
min_ff_n40C…	0.0486	0.0486	0.0000	0	0
max_tt_025C…	0.2292	0.2292	0.0000	0	0
max_ss_100C…	0.1453	0.7174	0.0000	0	0
max_ff_n40C…	0.0509	0.0509	0.0000	0	0

Saving the views

To save the views, run the following script with the following arguments in order:

1. The directory of the project
2. The macro name
3. The successful run tag

```
[nix-shell:~/openlane2]$ bash ~/caravel_aes_accelerator/openlane/copy_views.sh
```

This will copy the physical views of the macro in the specified run to your project folder.

Congrats! Now you have another AES accelerator as a Caravel user project.

3. Top-Level Integration strategy

In the top-level integration methodology, we will need the AES with the wishbone wrapper as a macro, then integrate it in th

with optimizations and cell insertion enabled on the top level.

AES Wishbone Wrapper Hardening

For the AES, we can use the macro hardened in the Macro-first hardening strategy here.

User Project Wrapper Hardening

Configuration

The following edits are needed for this strategy:

1. Change the Hardening strategy variables:

```
    "//": "Hardening strategy variables (this is for 3-Top-Level Integration).
    "SYNTH_ELABORATE_ONLY": false,
    "RUN_POST_GPL_DESIGN_REPAIR": true,
    "RUN_POST_CTS_RESIZER_TIMING": true,
    "DESIGN_REPAIR_BUFFER_INPUT_PORTS": true,
    "FP_PDN_ENABLE_RAILS": true,
    "RUN_ANTENNA_REPAIR": true,
    "RUN_FILL_INSERTION": true,
    "RUN_TAP_ENDCAP_INSERTION": true,
    "RUN_CTS": true,
    "RUN_IRDROP_REPORT": false,
```

2. Since we will have the AES as macro, the Macro Configurations section should be reverted:

```
    "//": "Macros configurations",
    "MACROS": {
        "aes_wb_wrapper": {
            "gds": [
                "dir::../../gds/aes_wb_wrapper.gds"
            ],
            "lef": [
                "dir::../../lef/aes_wb_wrapper.lef"
            ],
            "instances": {
                "mprj": {
                    "location": [1500, 1500],
                    "orientation": "N"
                }
            },
            "nl": [
                "dir::../../verilog/gl/aes_wb_wrapper.v"
            ],
            "spef": {
                "min_*": [
                    "dir::../../spef/multicorner/aes_wb_wrapper.min.spef"
                ],
                "nom_*": [
                    "dir::../../spef/multicorner/aes_wb_wrapper.nom.spef"
                ],
                "max_*": [
                    "dir::../../spef/multicorner/aes_wb_wrapper.max.spef"
                ]
            },
            "lib": {
                "*": "dir::../../lib/aes_wb_wrapper.lib"
            }
        }
    },
    "PDN_MACRO_CONNECTIONS": ["mprj vccd2 vssd2 VPWR VGND"],
```

3. Change the new variables section to just have the antenna and maximum wire-length variables

```
    "//": "New variables",
    "GRT_ANTENNA_ITERS": 10,
    "RUN_HEURISTIC_DIODE_INSERTION": true,
    "HEURISTIC_ANTENNA_THRESHOLD": 200,
    "DESIGN_REPAIR_MAX_WIRE_LENGTH": 800,
    "CTS_CLK_MAX_WIRE_LENGTH": 800,
```

So, the final config.json for the User Project's Wrapper will be:

config.json

Running the flow

To harden macros with OpenLane, we use the default flow, `Classic`.

```
[nix-shell:~/openlane2]$ openlane ~/caravel_aes_accelerator/openlane/user_proj
```

The flow will finish successfully in ~1.5 hours and we will see:

```
Flow complete.
```

Viewing the layout

To open the final GDSII layout run this command:

```
[nix-shell:~/openlane2]$ openlane --last-run --flow openinklayout ~/caravel_ae
```

Now, we can see that there are STD cells all over the `user_project_wrapper` and there is our macro in the middle.

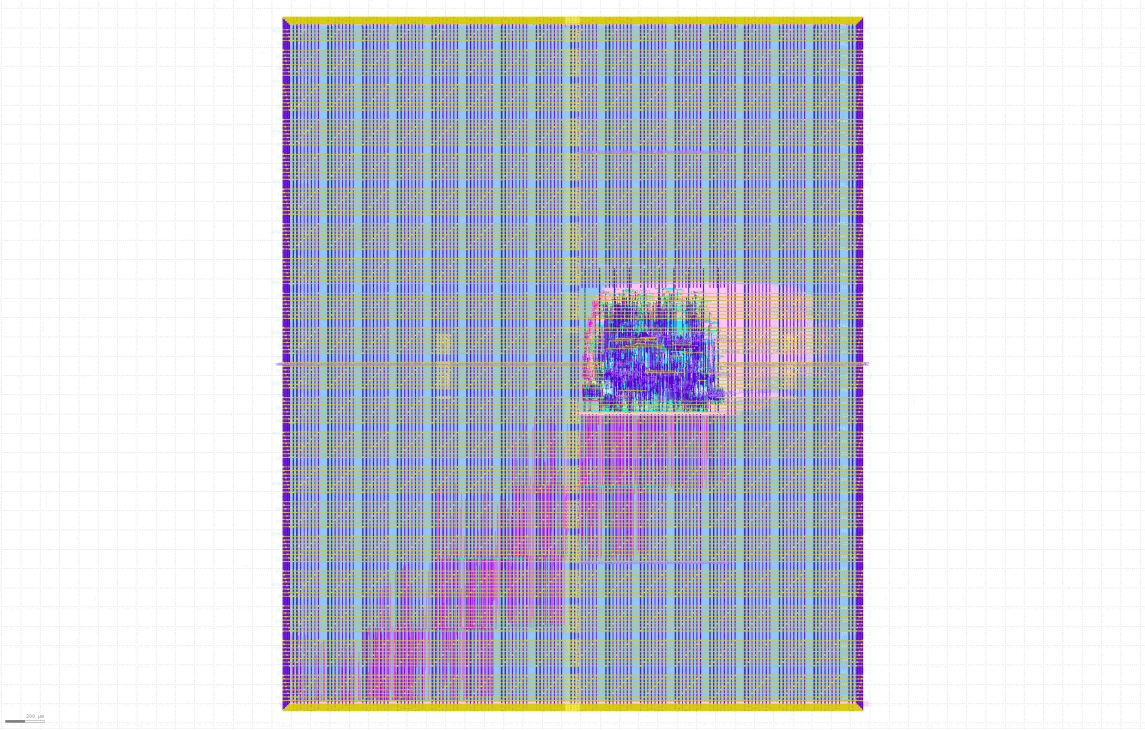

Fig. 32 Final layout of the user_project_wrapper with Top-level integration

Checking the reports

`OpenROAD.CheckAntennas`

There are no antenna violations at all since we already have the antenna variables in our configuration.

Partial/Required	Required	Partial	Net	Pin	Layer

`OpenROAD.STAPostPnR`

Looking at `xx-openroad-stapostpnr/summary.rpt`, there are no issues.

Corner/Group	Hold Worst Slack	Reg to Reg Paths	Hold TNS	Hold Violations	o t
Overall	0.0394	0.0394	0.0000	0	0
nom_tt_025C_…	0.2089	0.2089	0.0000	0	0
nom_ss_100C_…	0.6504	0.6504	0.0000	0	0
nom_ff_n40C_…	0.0444	0.0444	0.0000	0	0
min_tt_025C_…	0.2026	0.2026	0.0000	0	0
min_ss_100C_…	0.6390	0.6390	0.0000	0	0
min_ff_n40C_…	0.0394	0.0394	0.0000	0	0
max_tt_025C_…	0.2205	0.2205	0.0000	0	0
max_ss_100C_…	0.6725	0.6725	0.0000	0	0
max_ff_n40C_…	0.0515	0.0515	0.0000	0	0

Note

Despite the fact that the macro is placed very far from the top-level pins, there are no maximum slew violations because optimizations are enabled at the top-level and the long routes are being buffered.

Magic.DRC

Under the directory `xx-magic-drc`, you will find a file named `reports/drc.rpt` that summarizes the DRC violations reported by magic. The design is DRC clean so the report will look like this:

```
aes_wb_wrapper
----------------------------------------
[INFO] COUNT: 0
[INFO] Should be divided by 3 or 4
```

KLayout.DRC

Under the directory `xx-magic-drc`, you will find a file named `violations.json` that summarizes the DRC violations reported by KLayout
the report will look like this with `"total": 0` at the end:

```
{
  ⋮
  "total": 0
}
```

`Netgen.LVS`

Under the directory `xx-netgen-lvs`, you will find a file named `lvs.rpt` that summarizes the LVS violations reported by netgen. The design is LVS clean so the last part of the report will look like this:

```
Cell pin lists are equivalent.
Device classes user_project_wrapper and user_project_wrapper are equivalent.

Final result: Circuits match uniquely.
```

Saving the views

To save the views, run the following script with the following arguments in order:

1. The directory of the project
2. The macro name
3. The successful run tag

```
[nix-shell:~/openlane2]$ bash ~/caravel_aes_accelerator/openlane/copy_views.sh
```

This will copy the physical views of the macro in the specified run to your project folder.

Congrats! Now you have a third different AES accelerator as a Caravel user project.