Top-Level Integration & Power Routing

By Mohamed Hosni & Mohamed Gaber

Integration Options

There are 3 options for implementing user project wrapper using OpenLane. These include:

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, 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.

Each option is accompanied by a set of configuration variables for the user_project_wrapper.

For the **first** option, the layout will look as in figure 1. There is one macro that is hardened and placed in the user's project wrapper with no logic on the top-level.

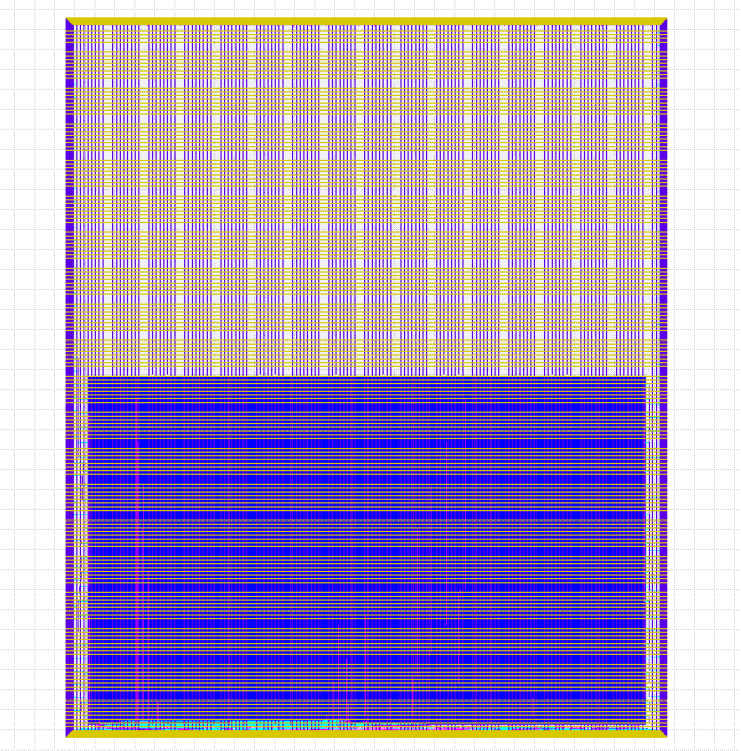

Figure 1. Integration option 1: Hardening the user macro(s) first, then inserting it in the user project wrapper with no standard cells on the top level.

The following variables should be set on the user's project wrapper configuration to achieve this implementation:

```
"SYNTH_ELABORATE_ONLY": 1,
"PL_RESIZER_DESIGN_OPTIMIZATIONS": 0,
"PL_RESIZER_TIMING_OPTIMIZATIONS": 0,
"GRT_RESIZER_DESIGN_OPTIMIZATIONS": 0,
"GRT_RESIZER_TIMING_OPTIMIZATIONS": 0,
"PL_RESIZER_BUFFER_INPUT_PORTS": 0,
"FP_PDN_ENABLE_RAILS": 0,
"GRT_REPAIR_ANTENNAS": 0,
"RUN_FILL_INSERTION": 0,
"RUN_TAP_DECAP_INSERTION": 0,
```

The following projects can be used as examples for this option:

- caravel_user_project
- Caravel_aes
- caravel_ips

For the **second** option, the layout will look as in figure 2. There are no macros and the top-level has all the logic flattened in it.

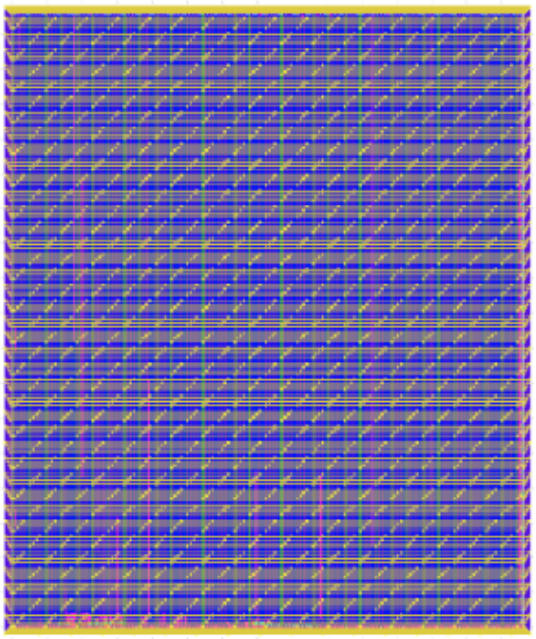

Figure 2. Integration option 2: Flattening the user macro(s) with the user_project_wrapper.

The following variables should be set on the user's project wrapper configuration to achieve this implementation:

```
    "SYNTH_ELABORATE_ONLY": 0,
    "PL_RESIZER_DESIGN_OPTIMIZATIONS": 1,
    "PL_RESIZER_TIMING_OPTIMIZATIONS": 1,
    "GRT_RESIZER_DESIGN_OPTIMIZATIONS": 1,
    "GRT_RESIZER_TIMING_OPTIMIZATIONS": 1,
    "PL_RESIZER_BUFFER_INPUT_PORTS": 1,
    "FP_PDN_ENABLE_RAILS": 1,
    "GRT_REPAIR_ANTENNAS": 1,
    "RUN_FILL_INSERTION": 1,
    "RUN_TAP_DECAP_INSERTION": 1,
```

For the **third** option, the layout will look as in figure 3. There are macros that are hardened and placed in the top-level and there is also logic on the top-level.

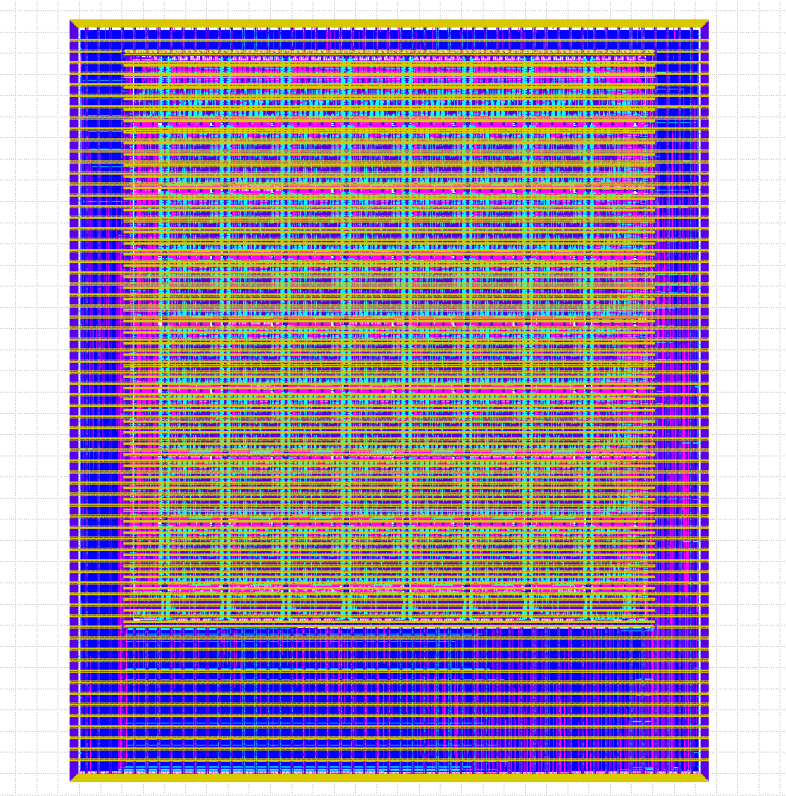

Figure 3. Integration option 3: Placing the user macro(s) in the wrapper along with standard cells on the top level.

The following variables should be set on the user's project wrapper configuration to achieve this implementation:

```
    "SYNTH_ELABORATE_ONLY": 0,
    "PL_RESIZER_DESIGN_OPTIMIZATIONS": 1,
    "PL_RESIZER_TIMING_OPTIMIZATIONS": 1,
    "GRT_RESIZER_DESIGN_OPTIMIZATIONS": 1,
    "GRT_RESIZER_TIMING_OPTIMIZATIONS": 1,
    "PL_RESIZER_BUFFER_INPUT_PORTS": 1,
    "FP_PDN_ENABLE_RAILS": 1,
    "GRT_REPAIR_ANTENNAS": 1,
    "RUN_FILL_INSERTION": 1,
    "RUN_TAP_DECAP_INSERTION": 1,
```

The following projects can be used as examples for this option:

- clear
- caravel_openframe_project

Power Routing

The method by which power routing for designs is performed varies depending on the integration method used. For option #2, (flattening the user macros with the user project wrapper,) there are no special considerations and the default hardening flow should automatically be able to power-route the design successfully.

Options #1 and #3 however are considerably more involved- the power routing techniques vary according to the characteristics of the macros being used and indeed- how they were hardened.

The two methods to power route for designs are:

- Hierarchical method: Saves space, but less routing layers are available for the macros (i.e. for a Macro nested *i*-deep, *N* - *i* metal layers are available for routing.) This is the default behavior and should be used in most cases.
- Ring method: Uses more space, allows any arbitrarily-nested macro to use the full routing layer stack. Useful if routing very complex macros.

Regardless of the power routing method being used, you need to ensure that your design is configured as follows:

- The Verilog headers for the submacros have the power pins declared as follows (with a preprocessor guard):

```
`ifdef USE_POWER_PINS
    inout vccd1,
    inout vssd1,
`endif
```

- The configuration includes hooks to show OpenLane how to connect the signals, as well as the name of the preprocessor guard for the power pins. Additionally, we ignore OpenROAD's PSM network checker as it lacks the ability to check hierarchical designs correctly.
    - In Tcl that would be:

```
set ::env(FP_PDN_MACRO_HOOKS) {mprj vccd1 vssd1 vccd1 vssd1}
set ::env(SYNTH_POWER_DEFINE) {USE_POWER_PINS}
set ::env(FP_PDN_CHECK_NODES) {0}
```

    - And in JSON:

```
"FP_PDN_MACRO_HOOKS": "mprj vccd1 vssd1 vccd1 vssd1",
"SYNTH_POWER_DEFINE": "USE_POWER_PINS",
"FP_PDN_CHECK_NODES": false
```

💡Note

Caravel has two pairs of power/ground pins: vccd1/vssd1 and vdda1/vssa1. The former are to be used for integrating digital macros and the latter for integrating analog macros.

To integrate macros that use both pairs (i.e. macros that themselves integrate both digital and analog components,) you will need to provide a custom OpenROAD PDN script. You can do this using the `FP_PDN_CFG` variable.

Hierarchical Method

The hierarchical method works as follows: the top level integration has access to all metal layers; and the deeper you go in the macro hierarchy, you lose the top-most metal layer as being available for routing.

So for example, a macro routed for top-level integration must not have any signals or power routed on `met5`. A macro routed *within* that must not have any signals or power routed on `met4`. And so on.

The power straps on the top-most metal layer of a submacro are then connected to the layer above using vias. (There is no continuity between power straps within each metal layer across the macro boundary.)

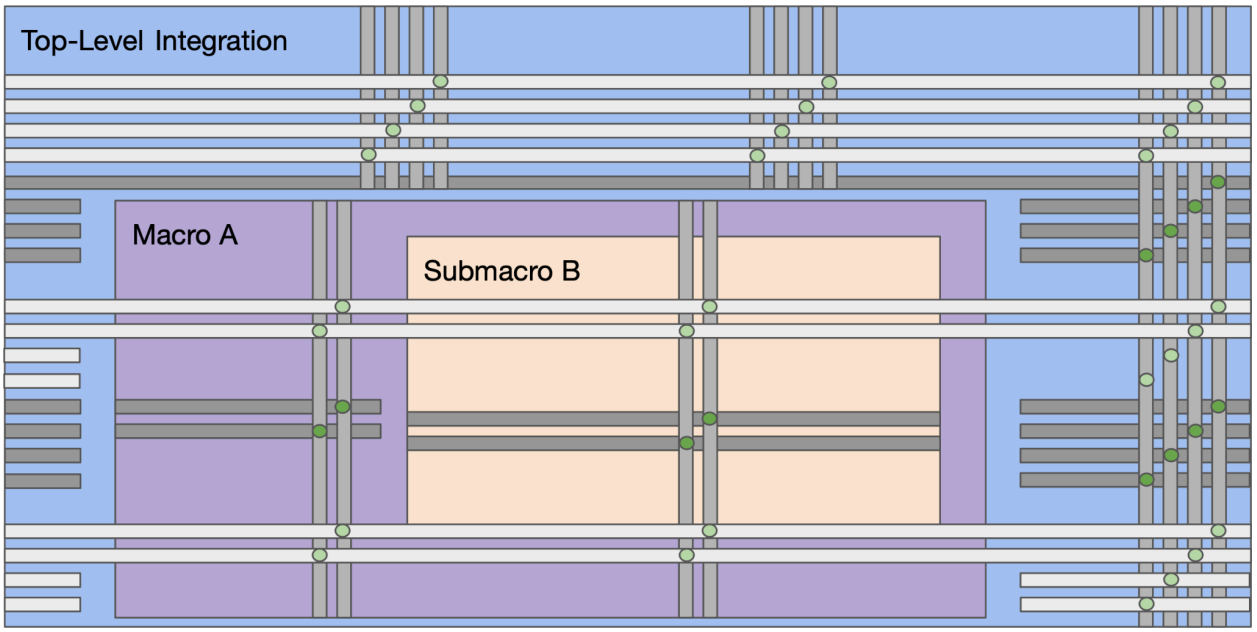

Figure 4: A top view of an example PDN, showing the power rails. Note how within each macro, straps may have different properties such as width, pitch or spacing.
The topmost metal layer, `met5`, is the lightest gray, while `met4` and `met3` are darker. Vias between metal layers are represented in green.

As you may be able to tell from Figure 4; sub-macros must be large enough for the straps in the layer above them to intersect with the straps in its topmost layer, otherwise the macro would not be connected to power.

Additionally, particularly in option 1, there are metal "stubs" generated as part of the power distribution network, which aren't connected to any macros. These are generally harmless but do cause higher routing congestion at the top level. If the configuration variable `FP_PDN_SKIPTRIM` is set to `0`[Tcl]/`false`[JSON], the PDN will attempt to remove those stubs.

The hierarchical mode is the default used by OpenLane and no configuration other than that which was shown above is required.

Ring Method

The ring method works as follows: each macro is hardened with a "power ring" around the **core area** of the macro, which unlike the hierarchical method, does interrupt the straps on the top level.

This allows the use of the full layers stack for routing, however, it takes more area, making it less space-efficient.

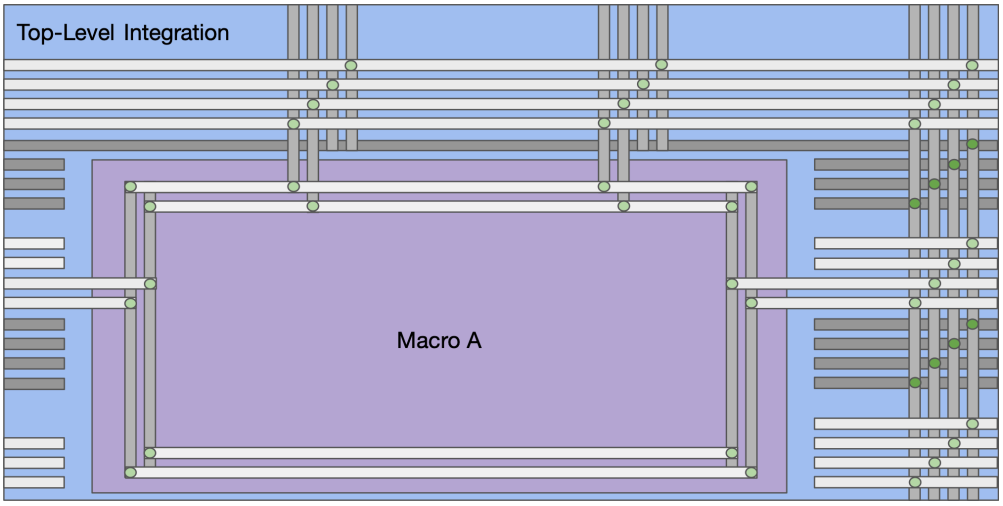

Figure 5: A top view showing the integration of a macro that uses power rings.

Akin to the hierarchical method; sub-macros must be large enough for the straps in the layer above them to connect to the rings, otherwise the macro would not be connected to power.

The core ring method does not actually require any special configuration for the top-level integration, but all macros need to be hardened with the following options:

- `FP_PDN_CORE_RING`: A boolean indicating whether to form a core ring around a macro or not. `0`[Tcl]/`false`[JSON] to disable, and `1`[Tcl]/`true`[JSON] to enable.
- `FP_PDN_HORIZONTAL_LAYER`: While the vertical layer may remain unchanged, the vertical layer should be different from the integrator (i.e.) met5 as .
- `FP_PDN_CORE_RING_HWIDTH`: The width of the horizontal straps forming the core ring.
- `FP_PDN_CORE_RING_VWIDTH`: The width of the vertical straps forming the core ring.
- `FP_PDN_CORE_RING_HOFFSET`: The distance between the horizontal boundaries of the die area and the beginning of the horizontal core ring straps.
- `FP_PDN_CORE_RING_VWIDTH`: The distance between the vertical boundaries of the die area and the beginning of the vertical core ring straps.
- `FP_PDN_CORE_RING_HSPACING`: The intra-strap distance within the two sets of horizontal straps forming the core ring.

- `FP_PDN_CORE_RING_VSPACING`: The intra-strap distance within the two sets of vertical straps forming the core ring.
    - All distances in μm.