NYDesign Workshop -- Design Example with OpenLane

Verilog Simulation

1) Go to the `designs/spm/hdl/dv` folder.

```
> cd hdl/dv
```

2) Simulate using iverilog

```
> iverilog ../src/spm.v ../src/mul32.v mul32_tb.v -o mul32_tb.vvp
> vvp mul32_tb.vvp
```

3) View the waveforms for debugging

```
> gtkwave mul32_tb.vcd
```

4) To automate the above steps, we created a Makefile with 3 targets: `sim`, `wave`, and `clean`.

```
> make sim          # simulate the design
> make wave         # load the vcd file into gtkwave
> make clean        # clean all generated files
```

OpenLane Hardening

1) Start OpenLane container by typing the following inside the root of the `spm` folder

```
> make pnr
```

This make target starts the container and mounts design folder inside the container as `/spm`

2) Create a template config.tcl for the design:

```
> ./flow.tcl -design /spm/openlane -init_design_config
```

This would create a `config.tcl` file under `/spm/openlane`

3) Update `config.tcl` file, found inside the folder `/spm/openlan`, to point to the design source files (`/spm/hdl/src`) and to update the design name to mul32

```
set ::env(DESIGN_NAME) mul32
set ::env(VERILOG_FILES) "/spm/hdl/src/spm.v /spm/hdl/src/mul32.v"
```

4) Run OpenLane flow:

```
> ./flow.tcl -design /spm/openlane -tag first_run
```

This command starts the flow to produce the layout in GDS2 format starting from the design HDL files. When the flow finishes, you should see the message:

```
[SUCCESS]: Flow complete.
```

As the flow runs, it produces lots of files, such as reports and log files, under

```
/spm/openlane/runs/first_run
```

5) To examine the resultant layout, navigate to the finishing folder under the run directory:

```
> cd  /spm/openlane/runs/first_run/results/finishing
```

then type:

```
> klayout mul32.gds
```

This would bring the produced layout.

6) To examine the layout after each step (for planning, placement, cts, ...), we can load the step DEF file produced by any step into Klayout. For example, to examine the layout after floor-planning, do the following:

- Navigate to `results/floorplan` folder under the results directory
  ```
  > cd /spm/openlane/runs/first_run/results/floorplan
  ```
- Copy the LEF file into the the current folder:
  ```
  > cp /spm/openlane/runs/first_run/tmp/merged.lef .
  ```
- Invoke Klayout:
  ```
  > klayout mul32.def
  ```
- Inside klayout: *File* ⇒ *Import* ⇒ *Other Files into Current*,
  this would open a dialog box. In the dialog click on the ... dots button to add the merged.lef file then select: *Extra Cells* option. Finally click on the *import* button.

Working with Caravel user's project example and wrapper

1) Navigate to caravel_user project folder

```
> cd caravel_user_project/
```

2) Harden the user's project example:

```
> make user_proj_example
```

3) Harden the user's project wrapper:

```
> make user_project_wrapper
```

4) Examine the generated layout:

```
> cd openlane/user_project_wrapper/runs/user_project_wrapper/results/finishing
> k user_project_wrapper.gds
```

Using Your own User Project

1) copy the files from `designs/spm/hdl/src` to `designs/caravel_user_project/verilog/rtl`

2) Navigate `designs/caravel_user_project/openlane` folder then make a copy of the forder `user_proj_example` into `user_proj_mul32`.

3) Navigate to the `user_proj_mul32 folder` then edit the file `config.tcl`

- Change the design name
  ```
  set ::env(DESIGN_NAME) user_proj_mul32
  ```
- Specify the design files
  ```
  set ::env(VERILOG_FILES) "\
        $::env(CARAVEL_ROOT)/verilog/rtl/defines.v \
        $script_dir/../../verilog/rtl/user_proj_mul32.v \
        $script_dir/../../verilog/rtl/mul32.v \
        $script_dir/../../verilog/rtl/spm.v"
  ```
- Disable basic placement
  ```
  # set ::env(PL_BASIC_PLACEMENT) 1
  ```
- Increase the target density:
  ```
  set ::env(PL_TARGET_DENSITY) 0.35
  ```

4) Inside the folder `designs/caravel_user_project`, run the following command

```
> make user_proj_mul32
```

5) Navigate to the folder `designs/caravel_user_project/openlane/user_project_wrapper`

6) Edit the config.tcl file as follows

```
### Black-box verilog and views
set ::env(VERILOG_FILES_BLACKBOX) "\
      $::env(CARAVEL_ROOT)/verilog/rtl/defines.v \
      $script_dir/../../verilog/rtl/user_proj_mul32.v"

set ::env(EXTRA_LEFS) "\
      $script_dir/../../lef/user_proj_mul32.lef"

set ::env(EXTRA_GDS_FILES) "\
      $script_dir/../../gds/user_proj_mul32.gds"
```

7) Go to the folder `designs/caravel_user_project/verilog/rtl`

8) Edit the file `user_project_wrapper.v` to change `user_proj_example` to `user_proj_mul32.`

9) Navigate `designs/caravel_user_project/openlane/user_project_wrapper`

10) Add the following line to the file `config.tcl`

```
set ::env(GLB_RT_ADJUSTMENT) 0.1
```

9) Inside the folder `designs/caravel_user_project`, run the following command

```
> make user_project_wrapper
```

Exploration

Sometimes we need to test different configurations to figure out which one best suits our design, and it wouldn't be practical to change the config.tcl file manually each time we want to test a new configuration. OpenLane provides run_designs.py, a script that can do multiple runs in parallel using different configurations on a single or multiple designs.
A run consists of a set of designs and a regression file that contains the configuration values. It is useful to explore the design implementation using different configurations to figure out the best one(s).
1) Create a regression configuration file (`regression.cfg`) with the following content

```
FP_CORE_UTIL=(45, 50, 55)
PL_TARGET_DENSITY=(FP_CORE_UTIL*0.01+0.05
)
CLOCK_PERIOD=(35, 40)
FP_ASPECT_RATIO=(0.8, 1.0)
FP_IO_MODE=(0, 1)
```

2) An exploration run that generates configuration files of all possible combinations of the passed regression file and runs them on the provided designs.

```
> python3 run_designs.py --regression ./scripts/config/regression.config
                         --threads 3 mul32 --tag regression
```

3) Each run will have its own output files under `./designs/<design>/runs/config_<tag>_<run_no>`.
In addition to these files, 3 files are produced:

- `regression_results/<tag>_<timestamp>/<tag>_<timestamp>.log`: A log file that describes the start and stopping time of each run.
- `regression_results/<tag>_<timestamp>/<tag>_<timestamp>.csv`: A report file that provides a summary of each run. The summary contains some metrics and the configuration of that run
- `regression_results/<tag>_<timestamp>/<tag>_<timestamp>_best.csv`: A report file that selects the best configuration per design based on number of violations

4) The synthesis strategy influences the design's area and speed. There are seven Synthesis strategies to choose from. To help the designer with selecting the right strategy, OpenLane provides a utility that performs synthesis strategies exploration and generates a summary report that compares the synthesis result of the different strategies. To do so, execute the following (assuming that you did not exit the docker container):

```
> ./flow.tcl -design OPC-7 -tag explore -synth_explore
```

This runs OpenLane flow in the synthesis exploration mode. OpenLane produces plenty of files while running and they are placed under `./designs/opc7cpu/runs/explore` folder. The name of the folder (explore) is used because we passed `-tag explore` to OpenLane. If you don't pass such a switch, OpenLane uses the current time-stamp as a run folder name. After some time, OpenLane finishes the exploration. Once finished, it prints the message "`[SUCCESS]: Done with synthesis exploration`" and points you to the location of the exploration report (HTML format). Loading the HTML file into your browser, gives you the following.