CIIC Harness

efabless

May 18, 2021

CONTENTS

CHAPTER

ONE

TABLE OF CONTENTS

CHAPTER

TWO

OVERVIEW

This repo contains a sample user project that utilizes the caravel chip user space. The user project is a simple counter that showcases how to make use of caravel's user space utilities like IO pads, logic analyzer probes, and wishbone port. The repo also demonstrates the recommended structure for the open-mpw shuttle projects.

CHAPTER

THREE

PREREQUISITES

- Docker

CHAPTER

FOUR

INSTALL CARAVEL

To setup caravel, run the following:

```
# By default, CARAVEL_ROOT is set to $(pwd)/caravel
# If you want to install caravel at a different location, run "export CARAVEL_ROOT=
↪<caravel-path>"
# Disable submodule installation if needed by, run "export SUBMODULE=0"

git clone https://github.com/efabless/caravel_user_project.git
cd caravel_user_project
make install
```

To update the installed caravel to the latest, run:

```
make update_caravel
```

To remove caravel, run

```
make uninstall
```

By default caravel-lite is installed. To install the full version of caravel, run this prior to calling make install.

```
export CARAVEL_LITE=0
```

CHAPTER

FIVE

CARAVEL INTEGRATION

5.1 Repo Integration

Caravel files are kept separate from the user project by having caravel as submodule. The submodule commit should point to the latest of caravel/caravel-lite master. The following files should have a symbolic link to caravel's corresponding files:

- Openlane Makefile: This provides an easier way for running openlane to harden your macros. Refer to *Hardening the User Project Macro using Openlane*. Also, the makefile retains the openlane summary reports under the signoff directory.
- Pin order file for the user wrapper: The hardened user project wrapper macro must have the same pin order specified in caravel's repo. Failing to adhere to the same order will fail the gds integration of the macro with caravel's back-end.

The symbolic links are automatically set when you run `make install`.

5.2 Verilog Integration

You need to create a wrapper around your macro that adheres to the template at user_project_wrapper. The wrapper top module must be named `user_project_wrapper` and must have the same input and output ports. The wrapper gives access to the user space utilities provided by caravel like IO ports, logic analyzer probes, and wishbone bus connection to the management SoC.

For this sample project, the user macro makes use of:

- The IO ports for displaying the count register values on the IO pads.
- The LA probes for supplying an optional reset and clock signals and for setting an initial value for the count register.
- The wishbone port for reading/writing the count value through the management SoC.

Refer to user_project_wrapper for more information.

CHAPTER

SIX

BUILDING THE PDK

You have two options for building the pdk:

- Build the pdk natively.

Make sure you have Magic VLSI Layout Tool installed on your machine before building the pdk.

```
# set PDK_ROOT to the path you wish to use for the pdk
export PDK_ROOT=<pdk-installation-path>

# you can optionally specify skywater-pdk and open-pdks commit used
# by setting and exporting SKYWATER_COMMIT and OPEN_PDKS_COMMIT
# if you do not set them, they default to the last verfied commits tested for this␣
↪project

make pdk
```

- Build the pdk using openlane's docker image which has magic installed.

```
# set PDK_ROOT to the path you wish to use for the pdk
export PDK_ROOT=<pdk-installation-path>

# you can optionally specify skywater-pdk and open-pdks commit used
# by setting and exporting SKYWATER_COMMIT and OPEN_PDKS_COMMIT
# if you do not set them, they default to the last verfied commits tested for this␣
↪project

make pdk-nonnative
```

CHAPTER

SEVEN

RUNNING FULL CHIP SIMULATION

First, you will need to install the simulation environment, by

```
make simenv
```

This will pull a docker image with the needed tools installed.

Then, run the RTL and GL simulation by

```
export PDK_ROOT=<pdk-installation-path>
export CARAVEL_ROOT=$(pwd)/caravel
# specify simulation mode: RTL/GL
export SIM=RTL
# Run IO ports testbench, make verify-io_ports
make verify-<dv-pattern>
```

The verilog test-benches are under this directory verilog/dv. For more information on setting up the simulation environment and the available testbenches for this sample project, refer to README.

CHAPTER

EIGHT

HARDENING THE USER PROJECT MACRO USING OPENLANE

You will need to install openlane by running the following

```
export OPENLANE_ROOT=<openlane-installation-path>
export OPENLANE_TAG=<latest-openlane-tag>
make openlane
```

For detailed instructions on how to install openlane and the pdk refer to README.

There are two options for hardening the user project macro using openlane:

1. Hardening the user macro, then embedding it in the wrapper
2. Flattening the user macro with the wrapper.

For more details on this, refer to this README.

For this sample project, we went for the first option where the user macro is hardened first, then it is inserted in the user project wrapper.

To reproduce hardening this project, run the following:

```
# Run openlane to harden user_proj_example
make user_proj_example
# Run openlane to harden user_project_wrapper
make user_project_wrapper
```

CHAPTER

NINE

RUNNING OPEN-MPW PRECHECK LOCALLY

You can install the precheck by running

```
# By default, this install the precheck in your home directory
# To change the installtion path, run "export PRECHECK_ROOT=<precheck installation path>"
make precheck
```

This will clone the precheck repo and pull the latest precheck docker image.

Then, you can run the precheck by running Specify CARAVEL_ROOT before running any of the following,

```
# export CARAVEL_ROOT=$(pwd)/caravel
export CARAVEL_ROOT=<path-to-caravel>
make run-precheck
```

This will run all the precheck checks on your project and will produce the logs under the `checks` directory.

CHAPTER

TEN

OTHER MISCELLANEOUS TARGETS

The makefile provides a number of useful that targets that can run LVS, DRC, and XOR checks on your hardened design outside of openlane's flow.

Run `make help to display available targets.

Specify CARAVEL_ROOT before running any of the following,

```
# export CARAVEL_ROOT=$(pwd)/caravel
export CARAVEL_ROOT=<path-to-caravel>
```

Run lvs on spice,

```
make lvs-<macro_name>
```

Run lvs on the gds,

```
make lvs-gds-<macro_name>
```

Run lvs on the maglef,

```
make lvs-maglef-<macro_name>
```

Run drc using magic,

```
make drc-<macro_name>
```

Run antenna check using magic,

```
make antenna-<macro_name>
```

Run XOR check,

```
make xor-wrapper
```

CHAPTER

ELEVEN

CHECKLIST FOR OPEN-MPW SUBMISSION

- [x] The project repo adheres to the same directory structure in this repo.
- [x] The project repo contain info.yaml at the project root.
- [x] Top level macro is named `user_project_wrapper`.
- [x] Full Chip Simulation passes for RTL and GL (gate-level)
- [x] The hardened Macros are LVS and DRC clean
- [x] The hardened `user_project_wrapper` adheres to the same pin order specified at pin_order
- [x] XOR check passes with zero total difference.
- [x] Openlane summary reports are retained under ./signoff/