Cadre Flow Guide

Cadre Group, University of Michigan

1 Introduction

The purpose of this guide is to provide information on the Cadre Flow and demonstrate running a simple design through the Cadre group's backend design flow ("Cadre Flow"). The Cadre flow is a set of scripts that quickly takes a design from RTL to GDS (plus simulation and verification). It is designed to work with arbitrary designs on different platforms (processes/kits) at arbitrary sites (universities/locations).

The scripts allow developers spend more time on the design and less time in the APR by providing a semi-automated and validated set of commands to push a design through Synthesis (using Synopsys Design Compiler), Implementation (Using Cadence Innovus) and verification (using Mentor Graphics Calibre). The flow can be used for any type of design across various processes and locations.

The sample design included in this version is a simple gcd (greatest common denominator) accelerator. The purpose of such a simple design is so that the turnaround time from RTL to signed-off GDS takes minutes and not days. It allows the user to view each step of implementation and understand how the flow works. We plan on creating other tutorials that feature more complex designs and exhibit more behaviors that would be found in making a state-of-the-art SoC.

Cadre Flow includes several components used for designing SoCs:

- A project directory structure for organizing source and generated files
- A Makefile-based flow for running EDA tools from synthesis to signoff
- Integration with EDA tool provider reference scripts
- Plugin scripts to allow easy user customization of flow steps

2. Design Directory Structure

By default, Cadre Flow uses a default directory structure for storing source files and generated files. These paths are customizable, but that is beyond the scope of this tutorial.

Source files/directories:

- blocks/ - This is an optional directory that is used for hierarchical designs. It is not used in this tutorial.

- include.mk - This is a Makefile fragment that is used for setting project parameters and paths. A majority of the required parameters are already set in the include default.mk file, which is sourced by this file. This file can also be used to override any defaults found in include default.mk.
- Makefile - This is the main Makefile for the project. All of the default rules are located in the Makefile default.mk file, which is sourced by this file. This file can be used to add additional rules or override the default rules.
- scripts/ - This directory contains user plugin scripts and design parameters that allow the user to customize steps of the flow.
- src/ - This is a directory that contains all of the source files for the project (Verilog, SystemVerilog, memory specs, etc.). The exact source directory structure inside of the src directory is quite flexible and user customizable.

Generated files/directories:

- *_dclib/ - This is a generated folder created by Design Compiler, where * is the design name. It contains generated files related to logical synthesis.
- *_mwlib/ - This is a generated folder created by Design Compiler, where * is the design name. It contains generated files related to physical synthesis.
- checkDesign/ - This is a generated folder created by Innovus. It contains several generated report files from Innovus.
- checkPoints/ - This is a generated folder created by Innovus. It contains checkpoints created by Innovus after each APR step so that they may be loaded by the next step or opened for debugging.
- export/ - This is a generated folder created by Cadre Flow by running make export. It contains copies of all the required files for exporting the design as a block to be used by other designs (.lib, .lef, .gds, etc.).
- logs/ - This is a generated folder created by Cadre Flow as a side effect to running most steps in the flow. It is the default location for any logs that are output by EDA tools.
- reports/ - This is a generated folder created by Cadre Flow as a side effect to running most steps in the flow. It is the default location for any reports or checks that are output by EDA tools.
- results/ - This is a generated folder created by Cadre Flow as a side effect to running most steps in the flow. It is the default location for final output files for most steps, such as netlists, layouts, timing libraries, etc.
- vpath/ - This is a generated folder created by Cadre Flow as a side effect to running any step in the flow. It is the default location for creating touch files, which are used by Make to look for rule dependencies and see which steps need to be run.

Getting the flow

1. You will need access to the git repositories hosted on bitbucket. Please contact cadre-admin@umich.edu to gain access.

2. Download the flow
   git clone git@bitbucket.org:DreslinskiLab/cad.git
3. Download the tutorials
   git clone git@bitbucket.org:DreslinskiLab/tutorials.git

3. Running the Cadre Flow

There are several rules included the the default makefile, which are generally sequentially dependent. The steps are:

1. `make synth` - runs synthesis synthesis using Design Compiler. The synthesized netlist ends up at results/dc/adder.mapped.v
2. `make init` - runs initialization of the netlist for APR using Innovus. The checkpoint ends up at checkPoints/init.enc.
3. `make place` - runs initial cell placement for APR using Innovus. The checkpoint ends up at checkPoints/place.enc.
4. `make cts` - runs clock tree synthesis (CTS) for APR using Innovus. The checkpoint ends up at checkPoints/cts.enc.
5. `make postcts hold` - runs post-CTS hold time fixing for APR using Innovus. The checkpoint ends up at checkPoints/postcts hold.enc.
6. `make route` - runs global route for APR using Innovus. The checkpoint ends up at checkPoints/route.enc.
7. `make postroute` - runs detailed route and optimization for APR using Innovus. The checkpoint ends up at checkPoints/postroute.enc.
8. `make signoff` - runs the QRC signoff-quality extraction engine to do timing analysis and backend (routing) DRCs. The checkpoint ends up at checkPoints/signoff.enc. It also creates the final unfilled GDS at results/innovus/adder.gds.gz
9. `make drc` - runs dummy fill and DRC with all DRC decks using Calibre. The filled design is merged with the seal ring GDS and the final GDS ends up at results/calibre/adder.top.gds.
10. `make lec signoff` - runs formal verification between the post-synthesis netlist and the post-APR netlist.
11. `make lvs` - runs Layout Vs. Schematic signoff using Calibre.
12. `make primetime` - runs timing and signal integrity signoff using PrimeTime SI.
13. `make voltus` - runs power signoff using Voltus.
14. `make export` - copies all files required to create a block design into the export folder. Because adder is a chip-level design, it will export the final GDS into the export folder as well (which is the only file you will need to submit to the fab).

Because of the Makefile dependencies, you can simply run make export to create the final GDS and copy it to the export/ directory. To make the design and run all signoff steps, you can run make all.

4. Examining Results

To examine the results for almost any step, you can simply run make debug_* where * is the name of the step as described in Section 3.

- debug synth elab and debug synth mapped will open up the elaborated and mapped designs in Design Vision, respectively. Any Innovus step will open up that step in the Innovus GUI.
- debug drc will open up the Calibre GDS viewer and the Calibre DRC database for the final GDS.
- debug lvs will open up the Calibre GDS viewer and the Calibre LVS database.
- debug voltus will open up the design with Voltus and load the simulation results data.

All reports (including for the steps not listed above) will be available in the reports/ directory under that tool's specific directory.

Supported Platforms

1. tscm16

Standard Cells: ARM
PDK: tsmc16
Supported Features: VCS, Synth, APR, DRC, LVS, PEX
Known issues:

2. asap7

Standard Cells: ARM
PDK: tsmc16
Supported Features: Synth, APR
Known issues:

- No support for physical synthesis (DC Topo)
-

3. freepdk45

Standard Cells: Nangate
PDK: freepdk45
Supported Features: Synth, APR
Known issues:

- No support for physical synthesis (DC Topo)

4. gf14

Standard Cells: ARM
PDK: gf14
Supported Features: Synth, APR
Known issues:

5. gpdk45

Standard Cells: Cadence
PDK: gpdk45 (Cadence Design Systems GPDK 45 nm)
Supported Features: Synth, APR
Known issues:

- No support for physical synthesis (DC Topo)

6. tsmc45

Standard Cells:
PDK:
Supported Features:
Known issues:

7. tsmc65lp

Standard Cells: ARM
PDK: tsmc65lp
Supported Features: Synth, APR
Known issues:

Adding a new Site/Platform

The scripts automatically select the platform based on the design's configuration file (include.mk) and the detected site. The detected site is the network domain derived from running "`hostname -f`". The flow will automatically attempt to include the following configuration files:

- `cad/{site}_{platform}.mk`
- `cad/synopsys_setup_{site}_{platform}.tcl`

In order to set up a new site, simply copy an existing site and update the tool locations and paths to point to the same documents.

As an example, the following instructions should be followed for setting up a new site to use tsmc65lp at a new site. This assumes you have successfully cloned the cad and tutorials repo. It will use the simple `tsmc65lp/gcd` desgin to validate the flow is successfully setup.

1. Run “hostname -f” to obtain the site name. This will be the domain name reported from the command. (e.g. A hostname of myserver.mycompany.com corresponds to the “mycompany” site)
2. Copy and rename existing files as a starter
   ```
   cp cad/umich_tsmc65lp.mk cad/mycompany_tsmc65lp.mk
   cp cad/synopsys_setup_umich_tsmc65lp.mk
   cad/synopsys_setup_mycompany_tsmc65lp.mk
   ```
3. First, edit `cad/mycompany_tsmc65lp.mk` and update all of the paths to match your local installation.
4. Next edit and inspect “`cad/synopsys_setup_mycompany_tsmc65lp.mk`“. This file will probably not need to be edited. If there are no TLUPLUS files available, set the following variable “DC_OPTIONS = -64bit” in “`cad/mycompany_tsmc65lp.mk`". This will disable topographical synthesis
5. Change directory to the “`tutorials/tsmc65lp/gcd`” directory and run “make synth” to validate the settings by attempting synthesis. Analyse any issues relating to missing files. Synthesis is successful it the “`tutorials/tsmc65lp/gcd/results/dc/gcd.mapped.v`” file exists
6. Test APR by running “`make init`”. Again, analyse any issues relating to missing files.
7. Complete the test by running “make”. APR is successful it the “`tutorials/tsmc65lp/gcd/results/innovus/gcd.gds.gz`” file exists

Supported sites

University of Michigan (cadre-admin@umich.edu)
University of California San Diego (bax002@eng.ucsd.edu)
San Diego Supercomputer Center (bax002@eng.ucsd.edu)
University of Virgina