VLSI Design Flow Tutorial: DRC and LVS

EE241B Tutorial
Written by Daniel Grubb (2020)

1 Overview

In this tutorial, we will focus mostly on Design Rule Checking (DRC) and Layout vs Schematic (LVS) checking. First, we will run through our VLSI flow again and produce a P&R'd GCD module. Then we will examine the design a bit more closely, run DRC and LVS, and examine the results.

2 Getting Started

2.1 Cadence VLSI Flow

First, we will need to set up the working directory for the VLSI design flow. If you still have your directory setup from lab 1, you may use it, but you need to pull from the source repo to get the lab 2 updates. Otherwise, you may clone the repo from scratch again. Once again, you will likely need to work in the `/scratch/` directory of the machine you are working on (use the eda or c125m machines). Run the commands below.

```
  % cd /scratch/userA
  % git clone ~ee241/spring20-labs/ee241bS20
  % cd ee241bS2020
  % git submodule update --init --recursive
or
  % cd /scratch/userA/ee241bS2020
  % git pull            // you will need to reconcile any local edits you've made
  % git submodule update

  % source sourceme.sh  // NEEDS TO BE RUN EVERY NEW SESSION
```

We will need to re-run the design through the tools using the Hammer flow again, as introduced in lab 1. Again, since Hammer generates a helper Make include, we can just run the below command to generate a similar design to lab 1.

```
  % make par
```

Note that if you edit an input yml/json file, the Hammer Make include file will detect this and re-run the flow. If you want to edit a setting without re-running the whole flow, you can run **make redo-STEP HAMMER_REDO_ARGS='-p new_input.yml'**. This will break the dependencies and just run that step, and additional arguments can be added to the call with the **HAMMER_REDO_ARGS** flag. New input files can add additional Hammer key flags, or it can override keys in the originally included files. Just make sure you are keeping track of what flags you are using when running the redo steps so you don't unintentionally change a setting.

3 Violation Inspection

The P&R tool, Innovus in this case, knows about the design rules for a technology and will do its best to create a design that is as clean as possible. However, in many cases Innovus is unable to fix all violations automatically, whether that is due to certain complexities or errors in input settings (like an incorrect floorplan). Innovus provides some utilities to check some of these issues. Open up the design in Innovus with:

```
% ./build/par-rundir/generated-scripts/open_chip
```

Run *Check Geometry > Ok* and *Check Connectivity > Ok*. Make sure you are in physical view and you should be able to see some white rectangles in the design (try turning some layers off if you are unable to see them). You can view the violations recorded by these checks if you open up the violation browser from the toolbar. Clicking on a violation will zoom you to it.

Q: 1. Take a screenshot of one of the violations that you find in the design, name it, and explain what is wrong. (hint: something that may help is looking at the design rules manual for ASAP7 at ~ee241/spring20-labs/asap7PDK_r1p5/docs/asap7_drm.pdf)

These checks are usually a good first pass look at some of the violations in the design. Some violations are less important to identify at this stage, but sanity checking that your power network is connected and there are no metal shorts is important. For more thorough checks, we need to actually run DRC and LVS.

4 Calibre Flow

You may have noticed a new submodule in the lab repo: **hammer-mentor-plugins**. This contains the Hammer plugins for the Mentor Graphics DRC and LVS tool `Calibre`. We will run through DRC and LVS on our design and examine some of the results.

4.1 DRC

You must run DRC on your design to check if it meets the rule requirements of the technology you are working in. Typically, a PDK will come with a design rule manual (ASAP7's DRM can be found at **~ee241/spring20-labs/asap7PDK_r1p5/docs/asap7_drm.pdf**). There are many rules, some of which we explored in lab 1, that have to do with metal spacing, metal density, antenna rules, and much more. A design typically must be `DRC clean` for it to be sent to the foundry, though certain rules may be waived depending on certain circumstances.

Run the following commands:

```
% make drc
% ./build/drc-rundir/generated-scripts/view_drc
```

Note, that similar to the **make redo-par** command, there is a **make redo-drc** command. This again will break all Makefile dependencies and just run DRC, in case you edited a Hammer input and don't want to run through the VLSI flow again.

This will open up the Calibre Design Review GUI and Calibre RVE. You can also look in the drc-rundir for text reports. You will not need to fix any DRC violations, but it is a good exercise

to look at a few of the errors and understand them. You can scroll through and click on some of the violations. It will zoom to and highlight the violation in the Design Review GUI. When first pushing through a design, there may be a very large number of violations, many of which can be eliminated in a systematic way by changing some flags or commands in your flow. When the number of violations is tractable, you may end up hand fixing some of them.

Q: 2. How many DRC violations are there total in your design?

Q: 3. Add a screenshot of Calibre RVE showing the DRC results to your lab report.

4.2 LVS

To signoff a design, you must also run LVS (Layout vs Schematic) to verify that your layout matches your source netlist. At a high level, this is done by giving the LVS tool your layout and a Spice netlist (can be generated from your post-P&R verilog netlist). The tool then extracts a netlist from your layout and attempts to match it to the source netlist. The PDK you use (ASAP7 in this case) will provide you with an LVS run-deck, which defines certain rules for the LVS tool (you can find these in the asap7PDK_r1p5/calibre/ directory. For a purely digital flow, LVS errors many times come from metal shorts from the P&R run, issues with the power network, or some other problems that you can try to catch earlier in the design process in Innovus. Some designers like to focus on getting their design `LVS clean` first (the design should never get `LVS dirty` after getting it clean), before fixing DRC violations.

Run the following commands:

```
% make lvs
% ./build/lvs-rundir/generated-scripts/view_lvs
```

You should get a green smiley-face in CalibreDRV to indicate that the design is clean! You can see the source netlist Spice files that are generated from your post-P&R verilog netlist in the lvs-rundir. You can also look there for text reports. A quick note on ASAP7 which is generally instructive. There is an error in some of the complex gates in ASAP7 which causes the design to come out LVS dirty, so these gates are excluded (you can look in the Hammer ASAP7 plugin). Sometimes certain cells must be excluded from your flow in a particular PDK, so this is an example of how this is done in Hammer.

Q: 4. Add a screenshot of Calibre RVE showing the LVS results to your lab report.

Q: 5. How many instances were matched between your layout and source netlist? (hint: look at the LVS report)

5 Conclusion

In this tutorial, we ran some checks in Innovus and examined some of the violations. We then ran DRC and LVS with Calibre. DRC and LVS are very important parts of chip signoff. It is typically important to get an LVS-clean and DRC-sane design early on in your design process. Doing this will help uncover systemic issues with your design regarding your floorplan, working with the technology, etc., and will save you stress as you get closer to tapeout.