Introduction to Cadence Jasper Gold

Author: Lars Fischer

Prior to going through this tutorial, you should familiarize yourself with SystemVerilog Assertions and understand the meaning of Assertions, Covers, and Assumptions. Please Read the pages 49-54 in the book by Erik Seligman. You can find it under library reserves in courseworks.

This document helps you getting familiar with the very basic features of Cadence Jasper Gold. Jasper Gold is the Formal Verification tool by Cadence and it is widely used in the industry. This example guides you thorough a very simple design of a fifo (first-in-first-out memory) and explains some basic features of the tool.

The tool is installed on the teaching machines in Mudd 1214 as well as Mudd 1235. Remote access is available under https://www.ee.columbia.edu/content/ee-computing-lab-remote-access.

1. Unpack the compressed file to a folder of your choosing.

Edit: I added a fifo_hard.sv file, which has some more issues, if you are interested in practicing your debugging skills a bit more.

You will see the following files:

- `clean.sh` – run this script to delete log files and session information.
- `run.sh` – this shell script invokes jasper and passes the config tcl script
- `fifo_setup.tcl` – this is the script that configures the jasper session. This example is very basic but for more sophisticated designs these scripts become very important. If any of the commands in this file are not clear, use the jasper help (once you open the GUI under help you find the User Guide, the Command Reference Manual and the Tcl Command Help)
- `fifo.sv` – contains the simple Verilog design of the fifo as well as the assertions.

NOTE: In this simple design it is fine to put assertions and design in the same files. However, in more complicated designs, we usually use a so called “SVA Bind”, this allows us to create a separate testbench that can access all the signals from the actual design.

2. Run the command: `sh run.sh`

If it is the first time you run it, it can take a bit longer but eventually a window with the jasper tool should pop up (You can ignore the warning in the terminal about Ignoring `XDG_SESSION_TYPE`)
Explore the GUI!

3. On the bottom you see the jasper terminal, you can enter any jasper specific tcl command here.
4. On the left side you see the design hierarchy (in this case only consisting of a single file) – Double click the file! The file viewer is very good for debugging as it lets you trace signals to their drivers. For example, click on `rd_valid` in line 17 and then on the D in the top bar – you should see the line that drives the `rd_valid` signal. This function can be very helpful when it comes to debugging (the L shows you where a signal is responsible for driving another signal)
5. Close the Source Browser.
6. On the right side you see the Property table.

There are 4 properties and four covers. In the bar above the properties there are some symbols on the right side, if you click the symbol right next to the a.b. (Enable Property Detail Pane), another small bar appears. Now click at an assertion – you will see the assertion from the code in the window below. Click at the related Covers. What do they mean? Where are they in the Code?

7. If you right click one of the properties and hit "`prove property`" Japser will try to prove that property, with the green arrow in the top bar, you can prove all at the same time.
8. Click the green arrow in the top bar (Prove all)
9. You should see two properties fail, an assertion, and a related cover.

One of the assertions has a red exclamation point. That means that one of the related covers was never reached.
Note: let's assume we have a property `assert (a → b)`*. the tool automatically generates a cover to check if we ever reach a state where* `a` *is true. If this cover is unreachable, it is likely that either the assertion is not written correctly, or the design has a mistake.*

10. Double click the failing assertion
11. A waveform window should open.

This window is the visualize app in jasper – a very powerful tool that is crucial to debugging.
Some tips:

- Double click the red shaded area in the waveform – this is where the assertion fails. The code to the relevant signal opens and you can investigate further.

- Once you have the code open, you can add waveforms via drag and drop. E.g. add the read and write pointer signals to the waveform by dragging them from the code that opened at the bottom and dropping It in the waveform.

12. Carefully review all signals and the code. Can you spot the mistake?

NOTE: The verification of this fifo is **not** complete as there would be further properties to assert. However, it gets you familiarized with the tool and sets you up well for the project. If you have any questions, please ask a question on Ed or reach out to lf2793@columbia.edu.