**Laboratory 01.- “Introduction to Formal Methods”**

Authors: Diego Hernandez Ramirez  
**Date: 01/12/19**

# Objective

Familiarise with the Symbiotic EDA Suite Tool for Model Checking, and get a first understanding on how Formal Tools works, and how they differ from a Simulation process.

# Material

1. PC
2. SEDA tool, FPV App

# Practical development

* **Description:** SEDA, as most tools, needs a configuration file on which the user needs to define what the tool should do, with the source code (SystemVerilog file). That configuration file is known as **SBY** file, or **SymbiYosys** file.

For the first example, you will get that configuration file. In the next ones, you are expected to define your own (the task is rather easy since you can reuse this one and make some small changes if needed).

**

Figure 1. Create new VIVADO project.

A brief explanation of the file:

* There are two main tasks under [tasks] option: precondition and check. Precondition will cover the antecedent of any suffix implication (you will know what that means latter in this course), and check, which will actually run BMC and K-induction for an specific property.
* The options are cover (for the precondition) and prove (for the assertion checks).
* Engine for both tasks is smtbmc. There are different engines, they perform better or worse depending on the design. This topic will be covered too, later in the course. For now, this engine will be used.
* The [script] option shows actual synthesis commands to elaborate the design into cells that the tool can formally check. You can ask me for details what these commands does. It also will be covered in the future. For now, this will be the “recipe” for the tests.
* Lastly, [files] section is where you define where your SV files are and what the name of these is.

The design to verify is shown below. It is a simple counter whose main property to verify is that, **if the counter register goes full, it should remain stable forever.** There is an obvious bug here. And we will fix it using the model checking tool.

***Grade yourself:*** Write a testbench to check if the design complains with that statement.

* How many cycles of clock you need to get a pass or fail status?
* How many lines of SystemVerilog you wrote for your testbench?

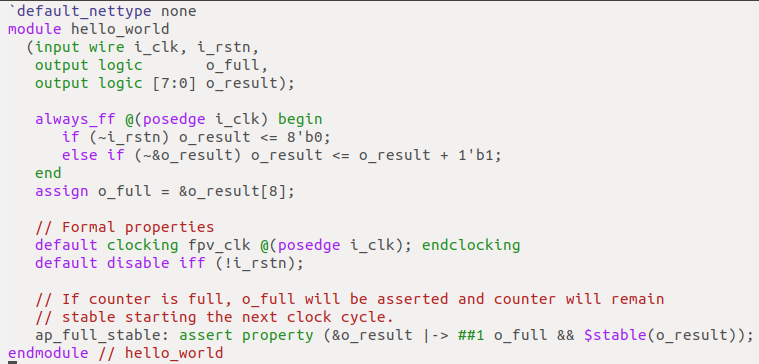
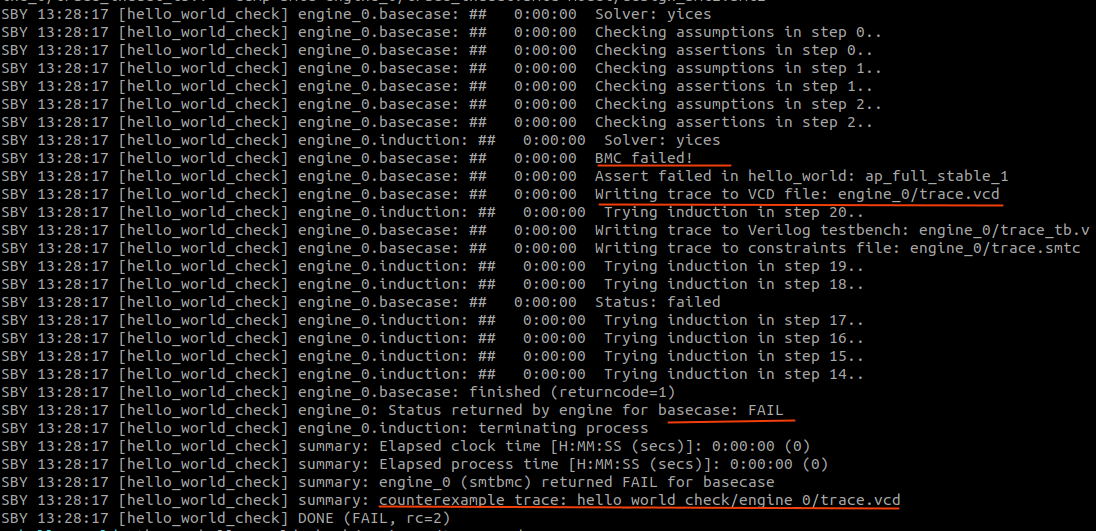


Figure 2. Write name of project

To run the tool, type this command in the terminal:

* ➜ hello\_world# **sby -f hello\_world.sby**

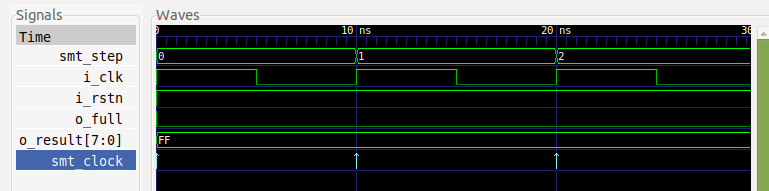
After running it, you will get a result in your terminal. In this case, I have highlighted the status so you can check it quickly. As you can see, there is a FAILED tasks. That means, our property is violated. to check the counterexample (CEX) you need to open the VCD file and look the trace.

  
Figure 3. Choose board.

To open the trace with GTKWave:

* ➜ hello\_world **gtkwave hello\_world\_check/engine\_0/trace.vcd**

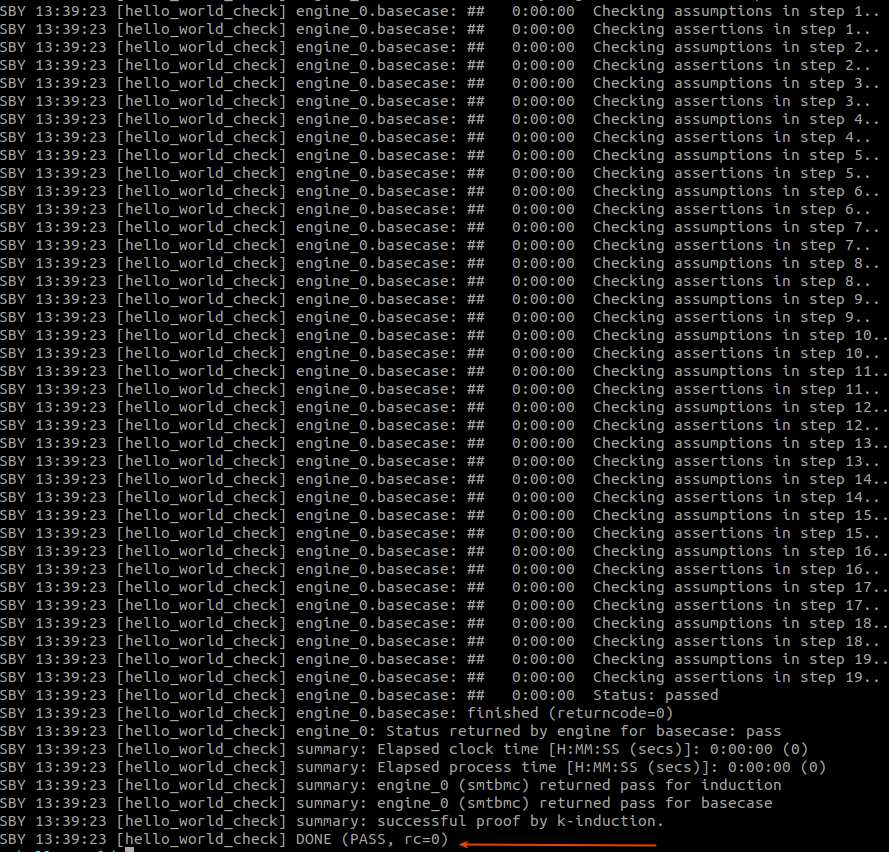
After pulling the signals into the waveform viewer, you will see something like this.



***Grade yourself:*** Describe what the problem is. Remember what was the property we wanted to check, what are the signals involved?.

* Can you fix the issue?

After fixing the problem, you will get a PASS status. This means, no matter what the input sequence of your design is, the tool guarantees you that your assertion will not be violated.

  
Figure 4. Create new block design.

**PART II - Booth Multiplier**

***Grade yourself:*** Now that you are already familiar with the tool, here is another interesting case. Can you analyze if there is a problem?, can you solve it?

* Copy the SBY file from **hello\_world** and modify it so it can read the new source file.
* Try to understand the design requirement.
* Run the Formal tool, and see if there is any error or not.
* Can you explain what is happening?
* Try to fix any possible problem.

The Booth multiplier is [in this link.](https://raw.githubusercontent.com/dh73/Model_Checking_Sandbox/master/session_I/booth_multiplier/booth_multiplier.sv) There is also a SBY file there. Try to first built it by yourself. If you end with problems, use the files in the [repository](https://github.com/dh73/Model_Checking_Sandbox/tree/master/session_I/booth_multiplier).

# Conclusions

After completing this lab, you will be able to understand how to interface with Symbiotic EDA tool for Formal Property Verification, and had a glance of how those flows looks like. Hopefully you can understand some of it usefulness, not only when verifying circuits, but also when modelling them.