# NYCU-EE IC LAB – FALL 2023

## Formal Verification – Bonus Exercise

1. **Using Assertion Based Verification IP (ABVIP) to verify AXI-4 Lite Master**
2. **Using Scoreboard & Customized SVA for Verification**

### Data Preparation

1. Data Extraction: tar xvf ~iclabTA01/Bonus_formal_verification.tar
2. The extracted LAB directory contains
   - 00_TESTBED/
     - Contain TA's **Usertype_PKG.sv** & **INF.sv**
   - 01_RTL/
     - Contain **bridge.sv (bug inserted)**
   - 02_JG/
     - Contain **run_jg, run.tcl, top.sv, jg.f, license.sh**
3. The answer of this lab: tar xvf ~iclabTA01/Bonus_formal_verification_ans.tar

### Description

In this bonus exercise, you are going to use ABVIP to verify the offered **Customized I/O to AXI4-Lite bridge**. Main focus of this exercise is to check the protocol, and find some bug inside the bridge with build-in scoreboard and customized SVA by Formal Verification tool - **JasperGold**.

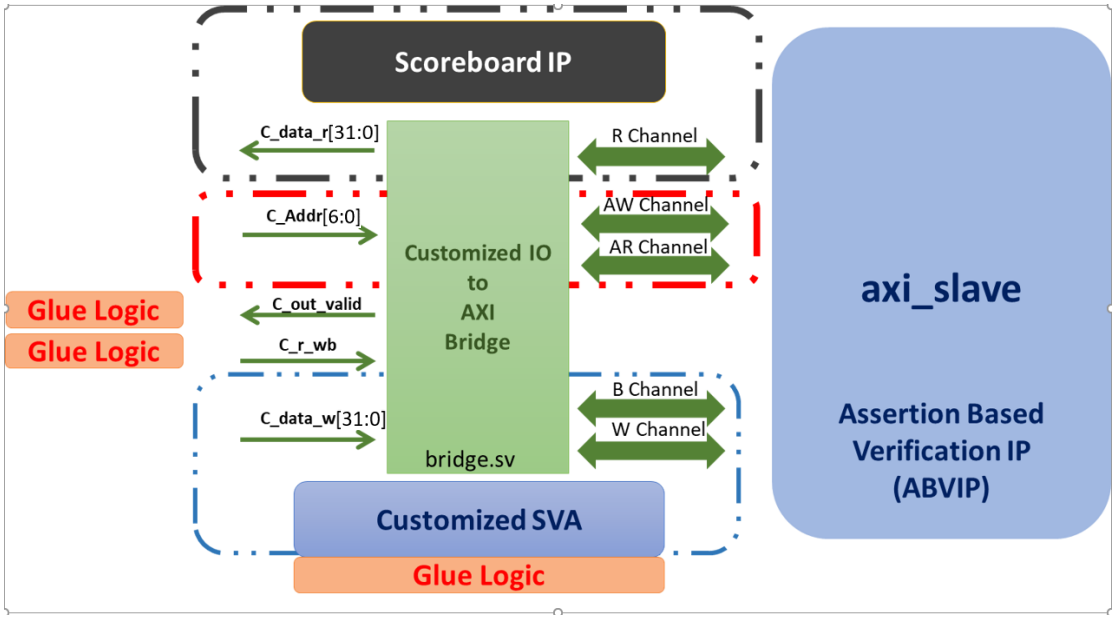
With ABVIP, we could use the build-in SVA property to check our design's AXI4-Lite interface part to make sure our design is reliable to handle any situation on the bus.

As for the interface connected to the design module, it requires customized SVA properties to check the remain of our design.

Different from previous lab (Simulation Based Verification with SVA), this lab demonstrates another perspective of verification – Formal Verification.

Formal Engine would transform our verification environment (including our design, SVA property and auxiliary logic) into another equivalent simplified mathematical problem, such that it uses different algorithms in the backend to finite number of traces to prove whether our design is verified or not.

## Block Diagram of the Verification Environment




## Customized Signal Behavior

### Input:

| name | width | Send to | |
|---|---|---|---|
| C_addr | 7-bit | bridge | C_addr Indicates which address we want to access.**(Bridge would do Address Conversion)** |
| C_data_w | 32-bit | bridge | The data to over right DRAM. |
| C_in_valid | 1-bit | bridge | High when the system is ready to communicate with bridge. |
| C_r_wb | 1-bit | bridge | 1'b1: Read axi_slave.<br>1'b0: Write axi_slave. |

### Output:

| name | type | Send to | note |
|---|---|---|---|
| C_out_valid | 1-bit | design | High when returned data from is ready. |
| C_data_r | 32-bit | design | The returned data from DRAM. |

## Bugs

I. There are 4 hardware bugs TA has insert in the design.

II. You should finish the blank in the top.sv to check **data integrity**

   A. Scoreboard Port Connection (inf.AW_ADDR)

   B. Customized Assertion (inf.C_data_w & W_DATA)

      ✓ Method1. Glue Logic + Assertion

      ✓ Method2. Undetermined Constant

## Note of Submission

1. This lab does not have a second demo opportunity!!!
2. No need to submit Verilog files, but you need to submit the answer of Quick Test, which include the answer of this lab, before 2023/12/11 12:00
3. Please submit the PDF format on the E3 platform, and name the file as Quick_Test_iclabxxx.pdf, for example, Quick_Test_iclab180.pdf.
4. Please name the file according to the file naming convention. If not, it will be considered a naming error, and five points will be deducted from the score.

**Enjoy Bug Hunting!!!**

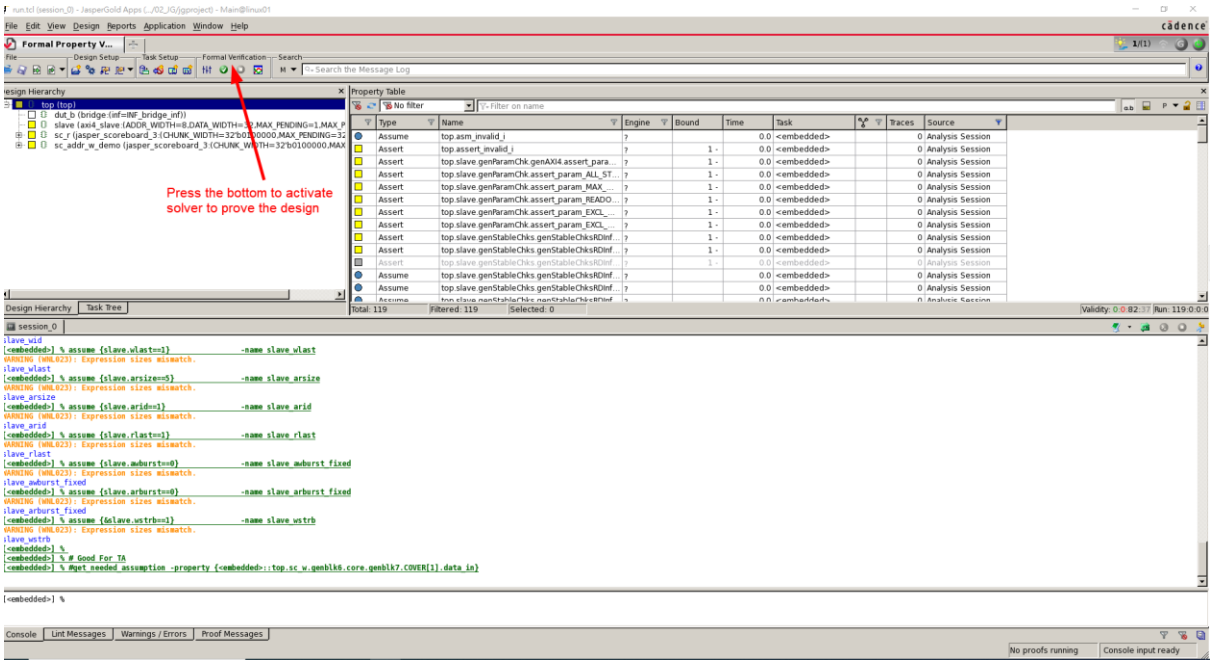
## Step to run

1. Go to Directory

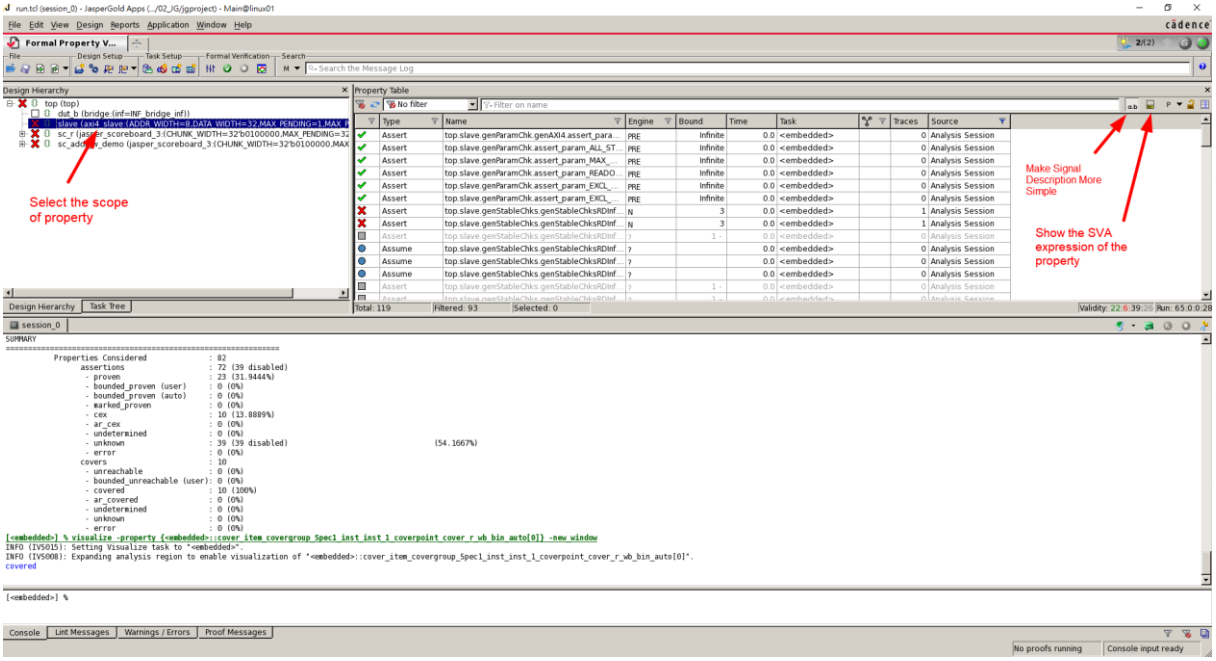
   ~/Bonus_formal_verification/Exercise/02_JG/

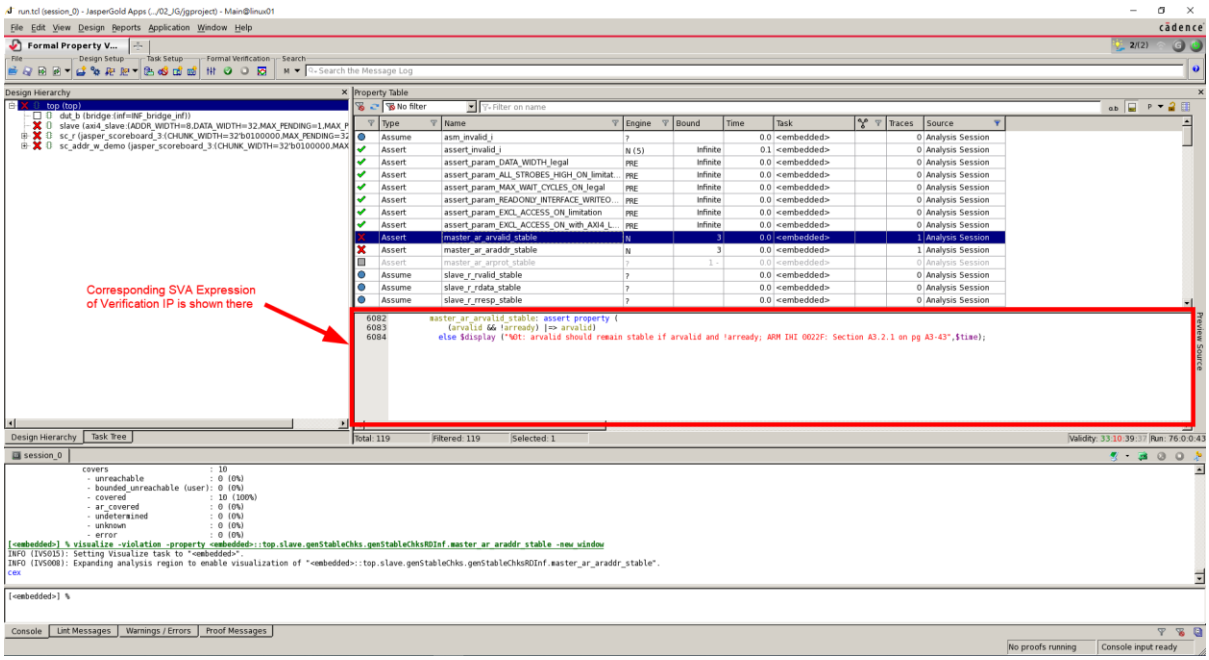
2. execute the command: source license.sh (**Important!!!**)
3. execute the command: ./run_jg

JasperGold would automatically start reading files (01_RTL/bridge.sv , 02_JG/top.sv)




After running proof

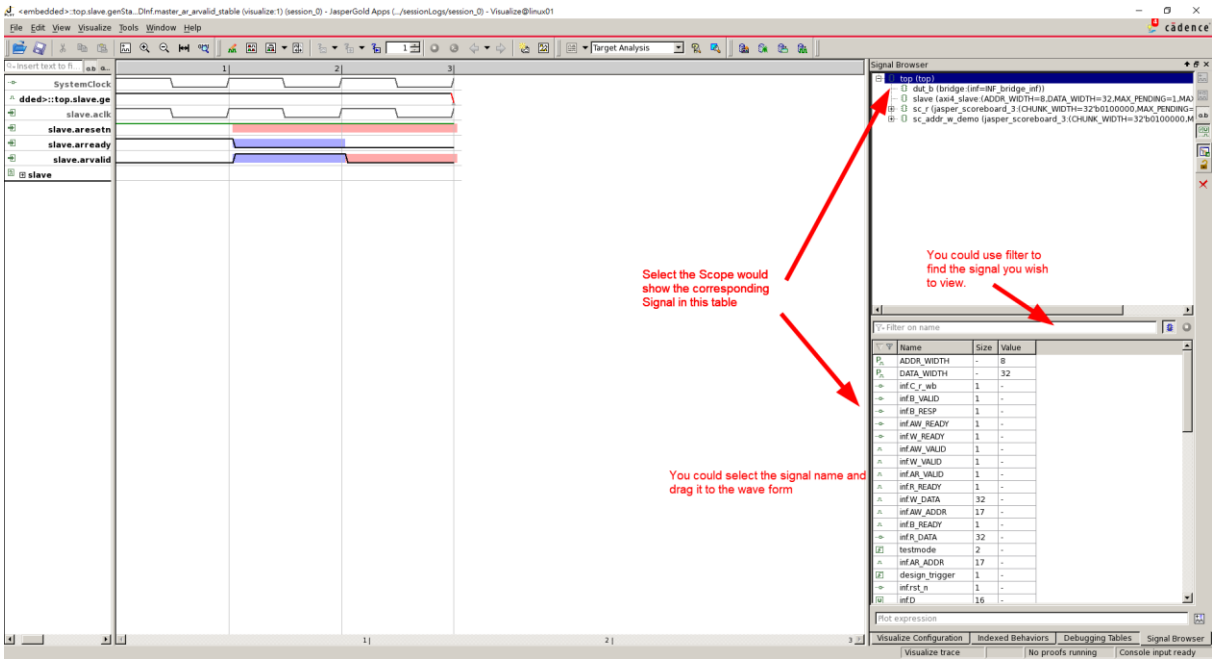


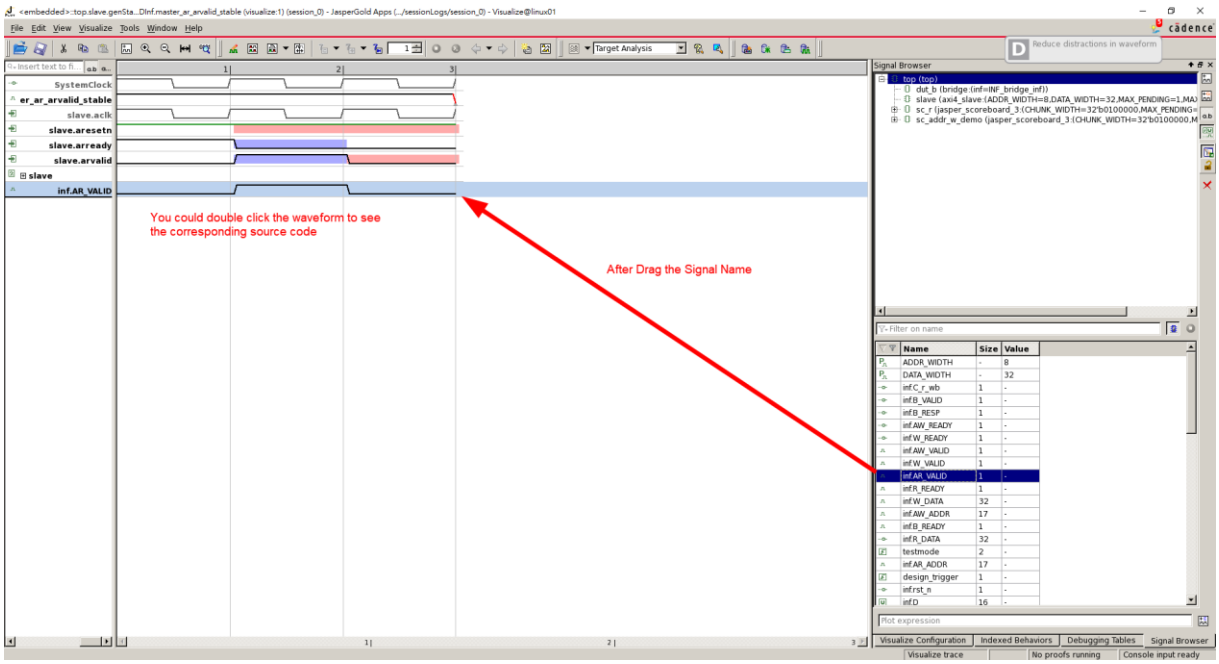


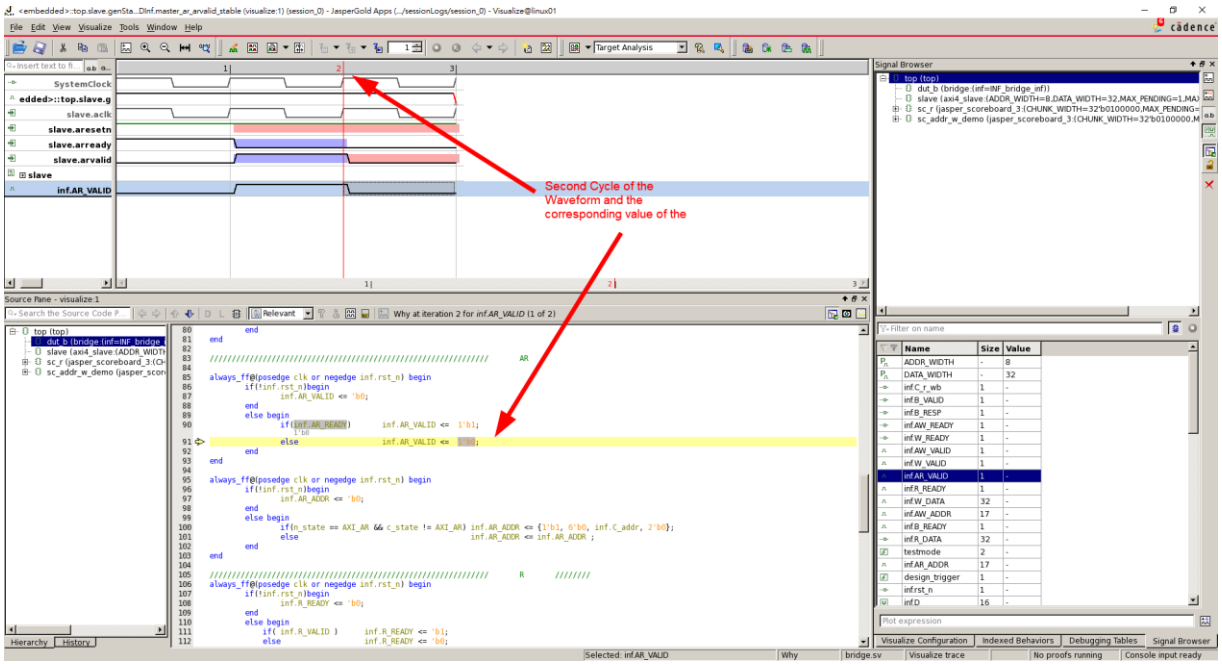

You could double click the read check to see the waveform which follow the property

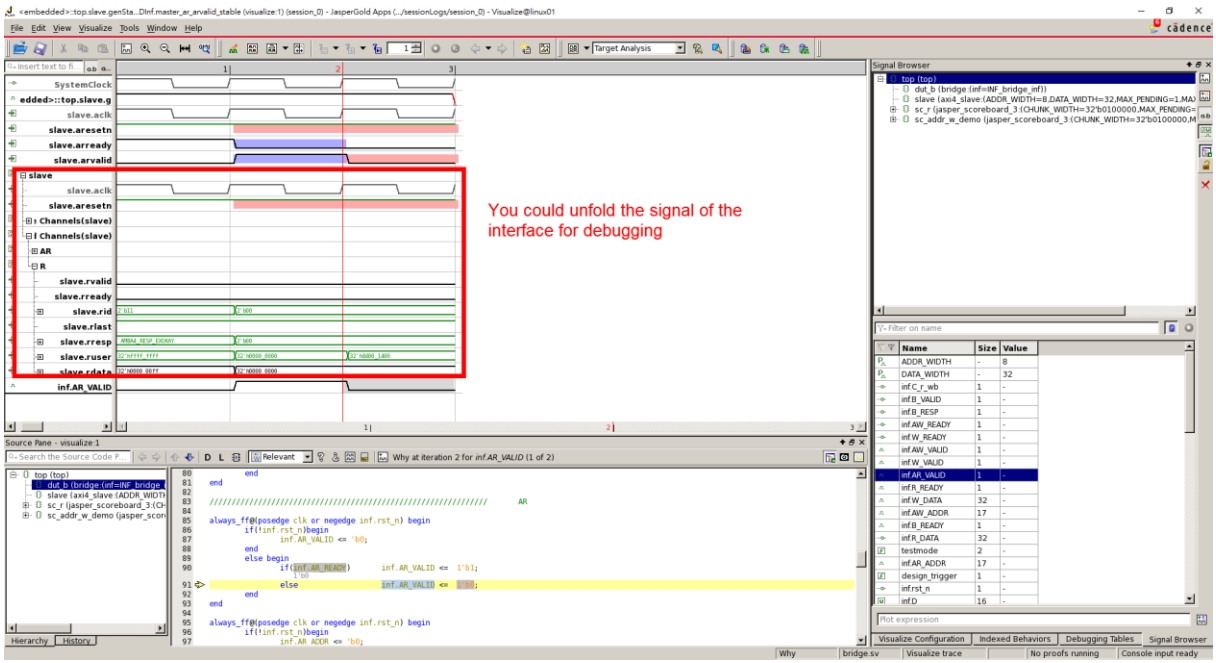
If there is assertion violation in your design, open the waveform and try your best to analyze the root cause of this violation and debug it.

## Basic Tutorial to view waveform

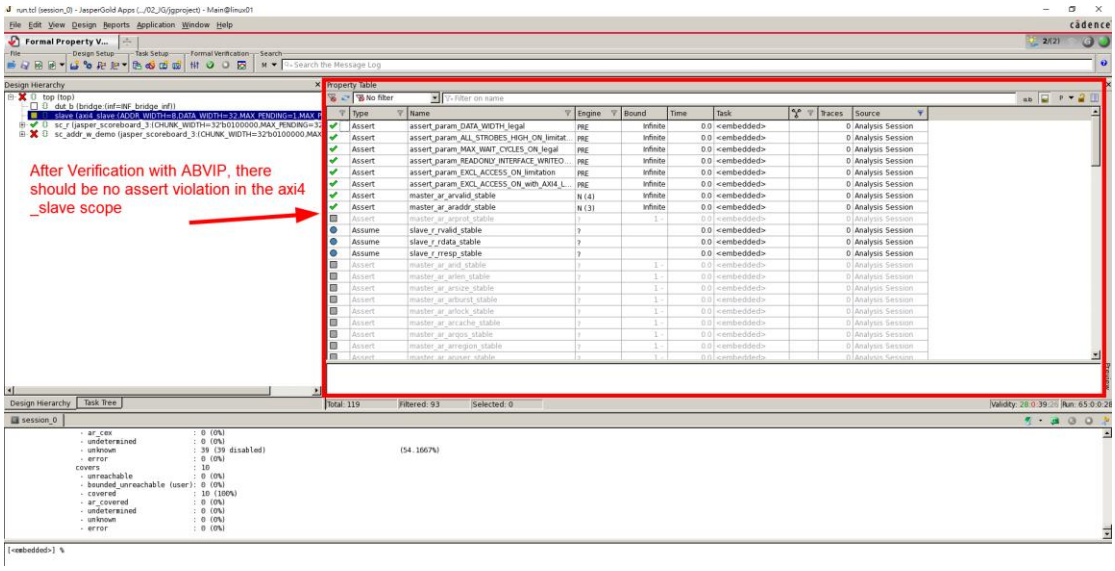



You could double click the waveform to see
the corresponding source code
After Drag the Signal Name


Second Cycle of the
Waveform and the
corresponding value of the




The figure of full proof of your assertion property




If the AXI-Lite Interface is already been proven, you could start debugging in the other scope such as scoreboards and or your customized SVA property in the top

The assertion property should not be violated, you could double click to check the waveform to see whether it match your property. You could scrutinize your **assumption, assertion or glue logic** to find the bugs with your SVA.

## Development Log:

Li-Wei, Liu @OASIS LAB ICLAB2019Fall

| | |
|---|---|
| Kai-Jyun, Hung @OASIS LAB | ICLAB2020Spring |
| Kai-Jyun, Hung @OASIS LAB | ICLAB2020FALL |
| Lin-Hung, Lai @Si2 LAB | ICLAB2021Spring |
| Wen-Yue, Lin @Si2 LAB. | ICLAB2021Fall |
| Po-Kang, Chang @Si2 LAB. | ICLAB2022Fall |
| Chan-Pin, Hung@Si2 LAB | ICLAB2023Spring |