NCTU-EE IC LAB – Fall 2019

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

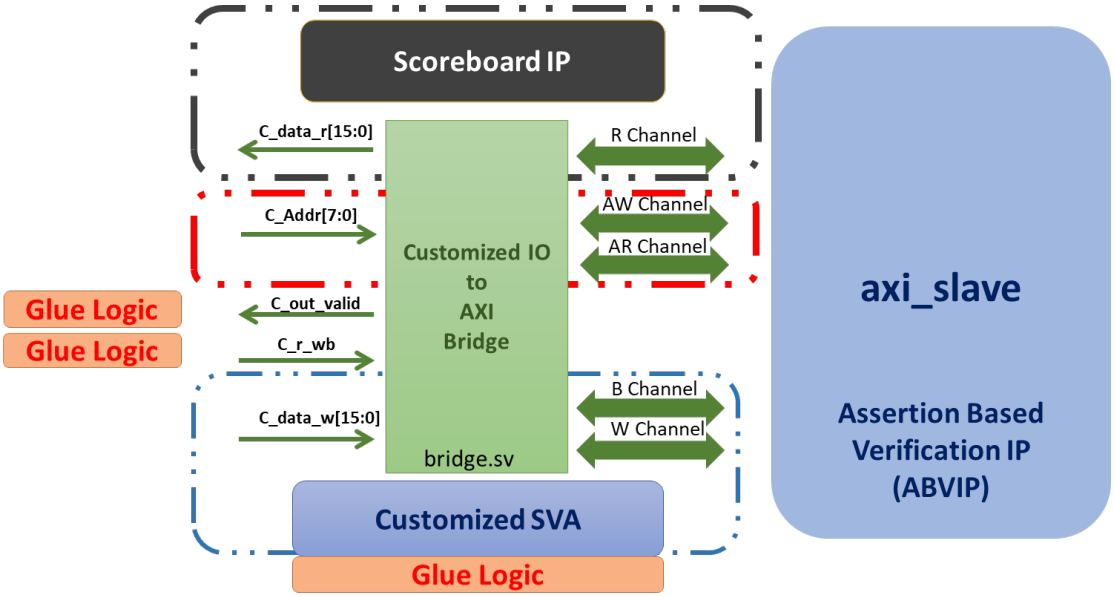
1. Data Extraction: `tar xvf ~iclabta01/ICLAB19F_Bonus_std_ver.tar`
2. The extracted LAB directory contains
   - 00_TESTBED/
     - Contain TA's **Usertype_PKG.sv** & **INF.sv**
   - 01_RTL/
     - Contain **bridge.sv (bug inserted)**
   - 02_JG/
     - Contain **run_jg, run.tcl, run_jg, top.sv**

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 payment module, it required 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 use 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 (Same as Lab07's Exercise)

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	17-bit	bridge	[16] =1'b0, password ; [16] =1'b1, balance [15:0] = The data to be sent into axi_slave
C_in_valid	1-bit	bridge	High when payment system is ready to communicate with bridge.
C_r_wb	1-bit	bridge	1'b1: Read axi_slave. 1'b0: Write axi_slave.

Output:

name	type	Send to	note
C_out_valid	1-bit	payment	High when returned data from is ready.
C_data_r	16-bit	payment	the returned data from axi_slave

Grading Policy (No 2nd Demo)

I. There are 4 hardware bugs TA has insert in the design. (60%)

A. Each Bug rewards 15%.

(We would use our top.sv to verify the bugs remained in your submitted bridge.sv.)

II. You should finish the blank in the top.sv to check **data integrity**. (40%)

A. Scoreboard Port Connection (inf.AW_ADDR) ….. (20%)

B. Customized Assertion (inf.C_data_w & W_DATA) ….. (20%)

Method1. Glue Logic + Assertion

Method2. Undetermined Constant

(We would use our bridge.sv to make sure your top.sv could find the bugs inserted by TA)

Note of Submission

1. Please upload the following files on new e3 platform before **12:00 noon** on **Dec 16th**.
2. Submission Format:

RTL design: **bridge_iclab??.sv** (?? is your account number)

Verification Environment: **top_iclab??.sv**

(Hints are left in the comment inside top.sv)

Enjoy Bug Hunting!!!

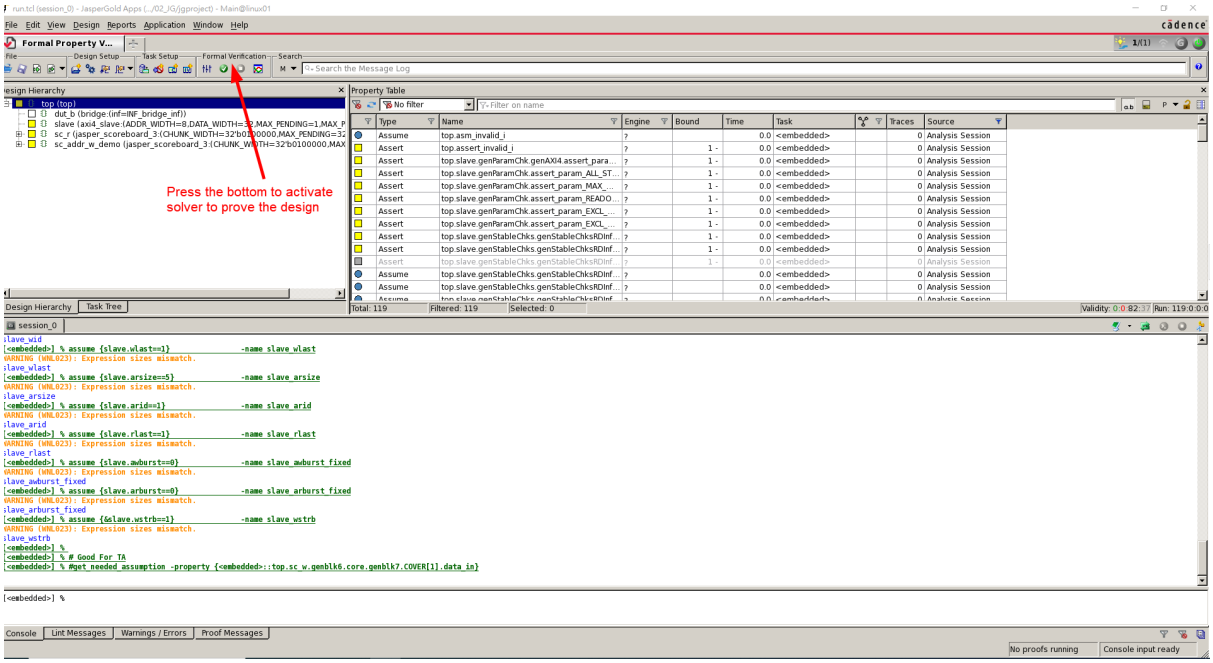
Step to run

1. Go to Directory

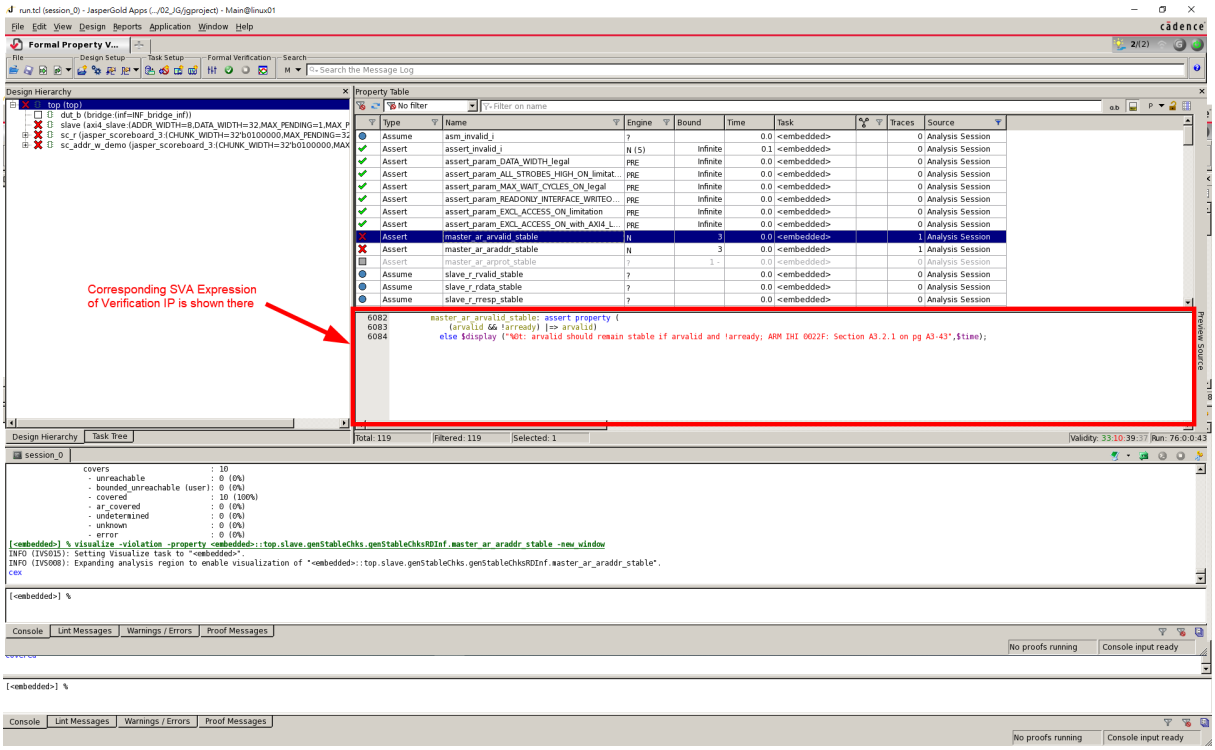
   ~/ICLAB19F_Bonus_Dev/ICLAB19F_Bonus_std_ver/Exercise/02_JG

2. execute the command: ./run_jg

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



After running proof

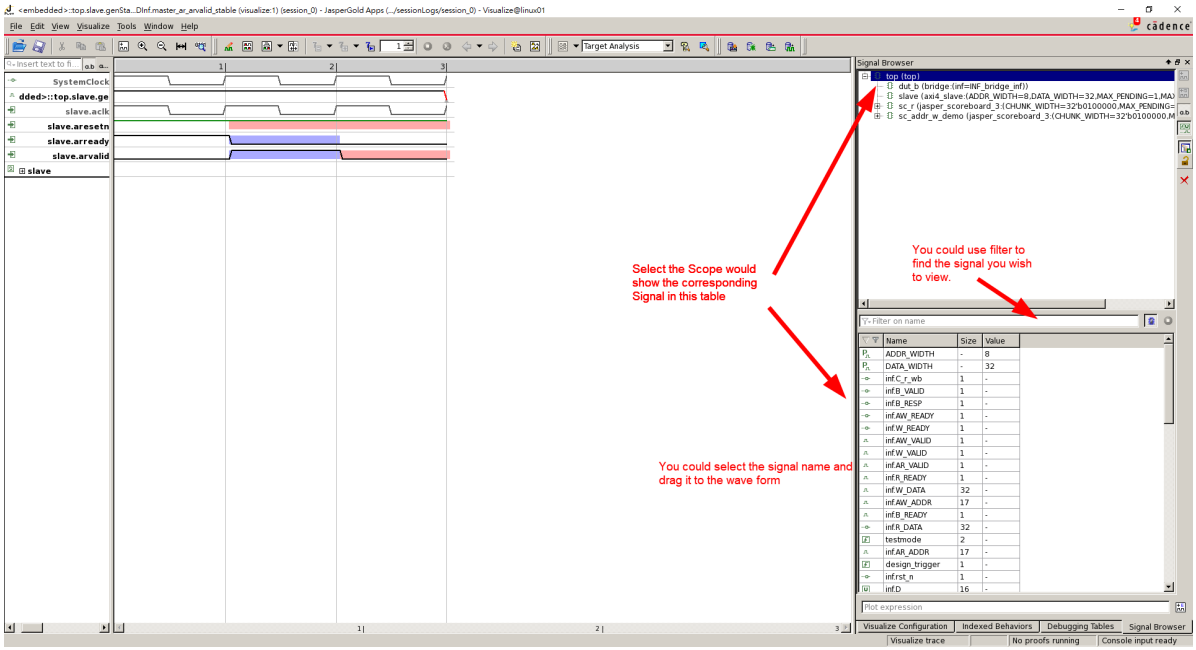


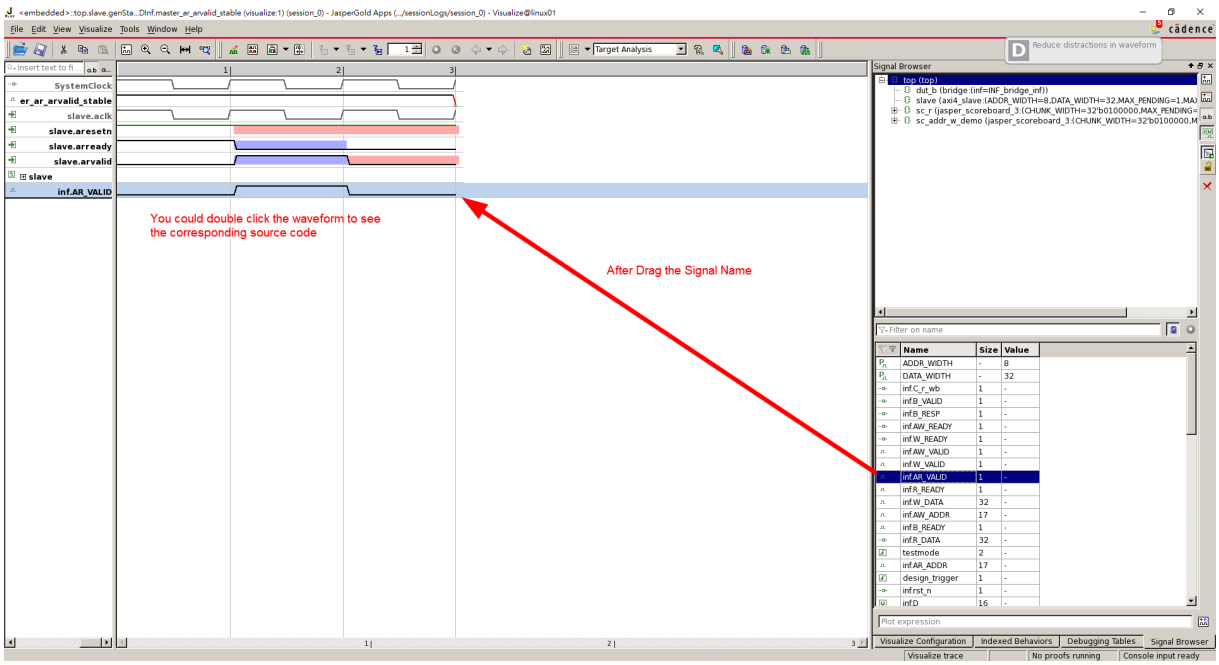
You could double click the read check to see the waveform which follow the property

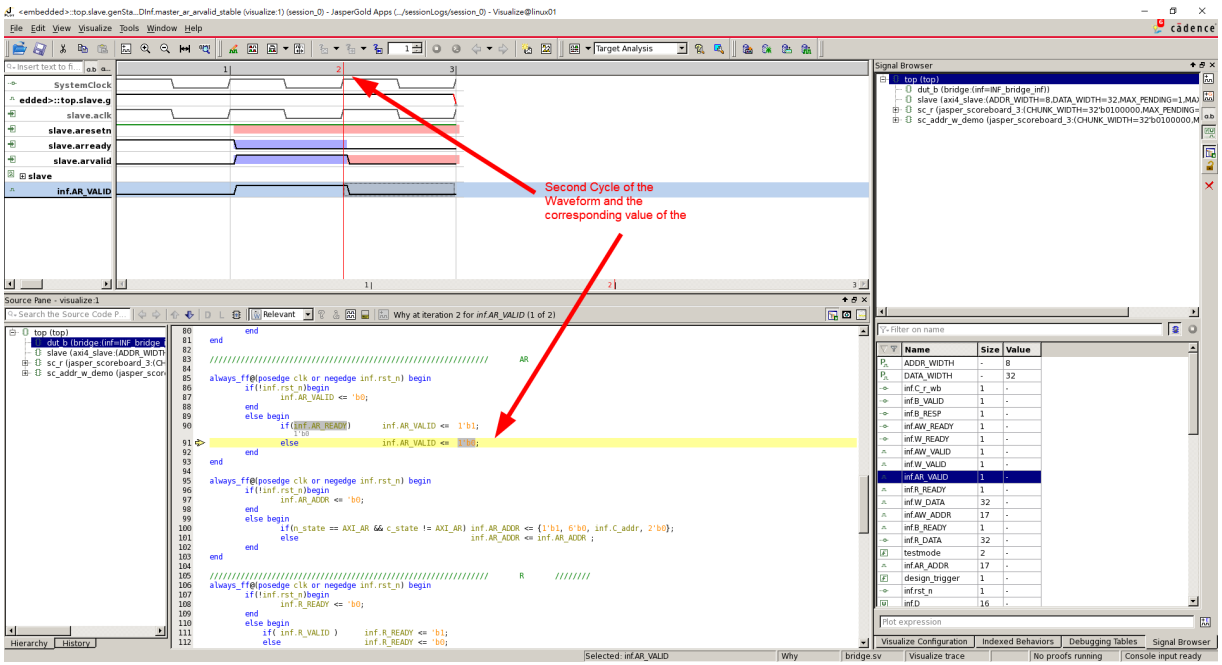
If there is assertion violation in your design, open the waveform and try your best to

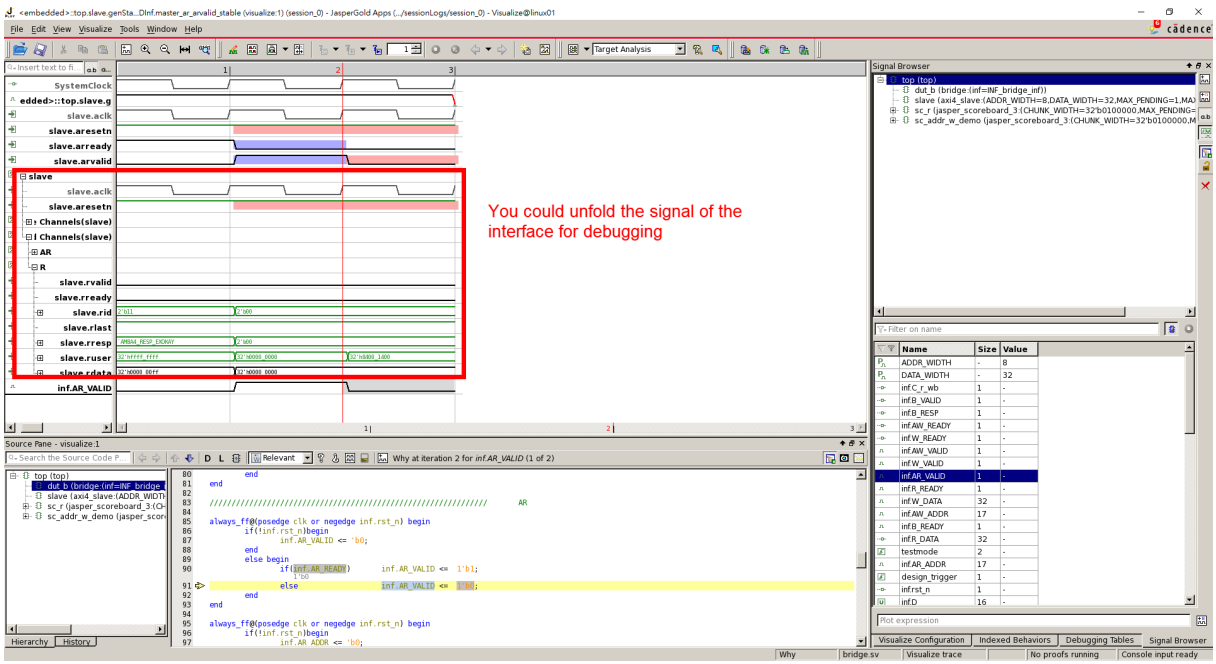
analyze the root cause of this violation and debug it.

Basic Tutorial to view waveform

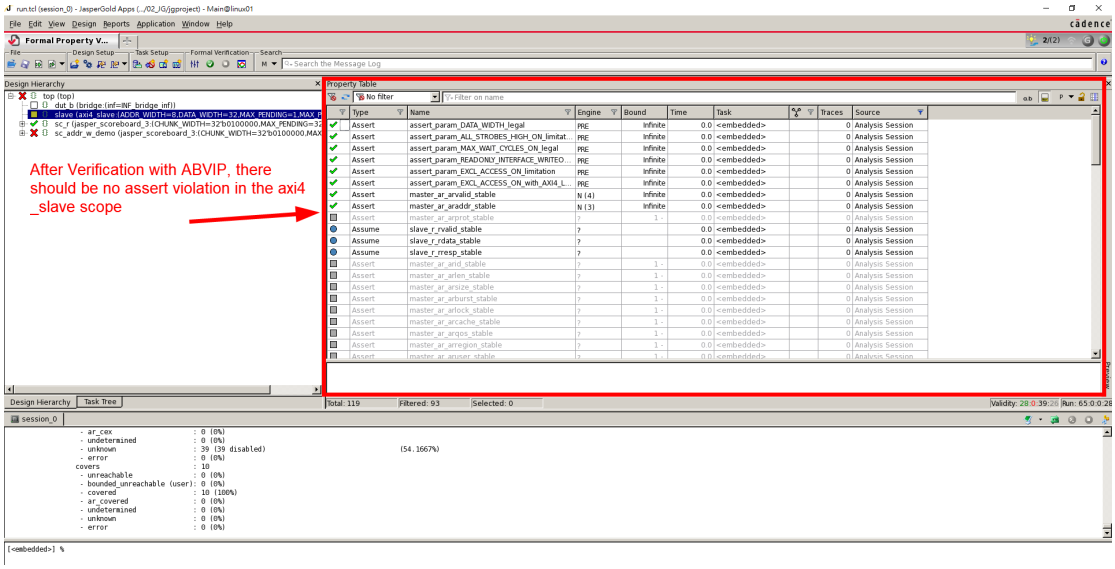








The figure of full proof of your assertion property



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