IC Lab Formal Verification  
Bonus Quick Test

2023 Spring

**Name: 李宜紘 Student ID: 0812019 Account: iclab034y**

1. Bonus:
2. What is Formal verification?

Formal verification is the act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics. Usually, formal verification is done before using simulation (for software) and emulation (for hardware) to verify the functionality of the design.

What's the difference between Formal and Pattern based verification?

And list the pros and cons for each.

**Formal verification:**

It tests all possible stimulus, one cycle at the time. It is used to test and try to visit all the states in every cycle.

**Pro:** 1. Systematic method

-None or very little randomization

-More deterministic

2. Less testbench effort required

-Formal testbench tends to be much simpler than sim testbench

3. Leads to higher quality

-Find bugs from a different angle: find with breadth-first search(BFS)

-Often reveals bugs that simulation would never catch

4. Improve productivity and schedule

-Can replace some block-level testbenches

-Verification can begin prior to testbench creation and simulation

**Con:** If the design is too complicated or large, it may take a long time to test.

**Pattern based verification:**

Only input some random values into the system when the system is able to get data from outside. It always only test and visit a states every time. That behavior is similar to do a DFS in the testing system.

**Pro:** 1. Directly test the functionality of the design

2. The testing input may be more meaningful than formal verification.

**Con:** 1. Usually use randomization to create testing input, and that may cause some state

can not be visited and tested during verification. (Undetermined states)

2. Complicated Testbench

3. Need to implement the whole algorithm of the target function precisely. Not just describe some I/O behavior.

1. What is glue logic?

Why will we use glue logic to simplify our SVA expression?

While modeling complex behaviors, SVA may be complicated. Thus, we can use some “auxiliary logic” to observe and track events. That’s called glue logic. After using this trick, we can simplify codes, and expressing SVA properties may be trivial.

1. What is the difference between Functional coverage and Code coverage?

What’s the meaning of 100% code coverage, could we claim that our assertion is well enough for verification? Why?

**Functional coverage:**

It is used to test the behavior of the design. The targets are specific states, conditions, or sequences.

**Code coverage:**

It is used to test the code can be executed(visit) when the system is working. The targets are the branch, statement, and expression in the code.

No, we cannot claim that because it may not capture all meaningful design functionality. On the other words, the functionality may exist some mistakes.

1. What is the difference between COI coverage and proof coverage for realizing checker’s completeness? Try to explain from the meaning, relationship, and tool effort perspective.

**COI:**

**Meaning:** COI coverage is Cone-Of-Influence Coverage. With COI coverage, we can get a COI region, which indicate holes in the assertion set. Theses holes stand for some code that is not checked by any asserts.

**Relationship:** COI represents the maximum potential of proof coverage.

**Tool effort perspective:** It is a fast measurement because no formal engines are run. COI

doesn’t require a proof to take place.

**Proof Coverage:**

**Meaning:** It represents the portion of the design verified any formal engines. It will find the

region cannot truly influence assertion status

**Relationship:** It is a subset of the COI.

**Tool effort perspective:** It need greater tool effort and run slower than COI.

1. What are the roles of ABVIP and scoreboard separately?

Try to explain the definition, objective, and the benefit.

**ABVIP:**

**Definition:** The Assertion Based Verification Intellectual Properties are a comprehensive set of checkers and RTL that check for protocol compliance.

**Objective:** To verify a protocol and analyze its completeness.

**Benefit:** It can help the designer to verify new system within some protocol easier and more quickly.

**Scoreboard:**

**Definition:** Scoreboard behaves like a monitor to observe input data and output data of DUV.

**Objective:** To check (1) Data packet dropped (2) Duplicated Data Packets

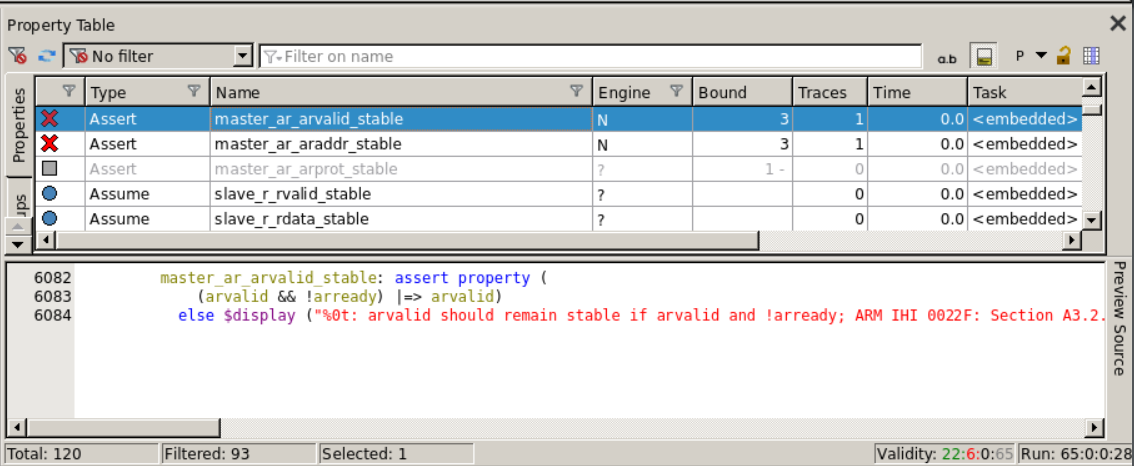
(3) Order of Data Packets (4) Corrupted Data Packets

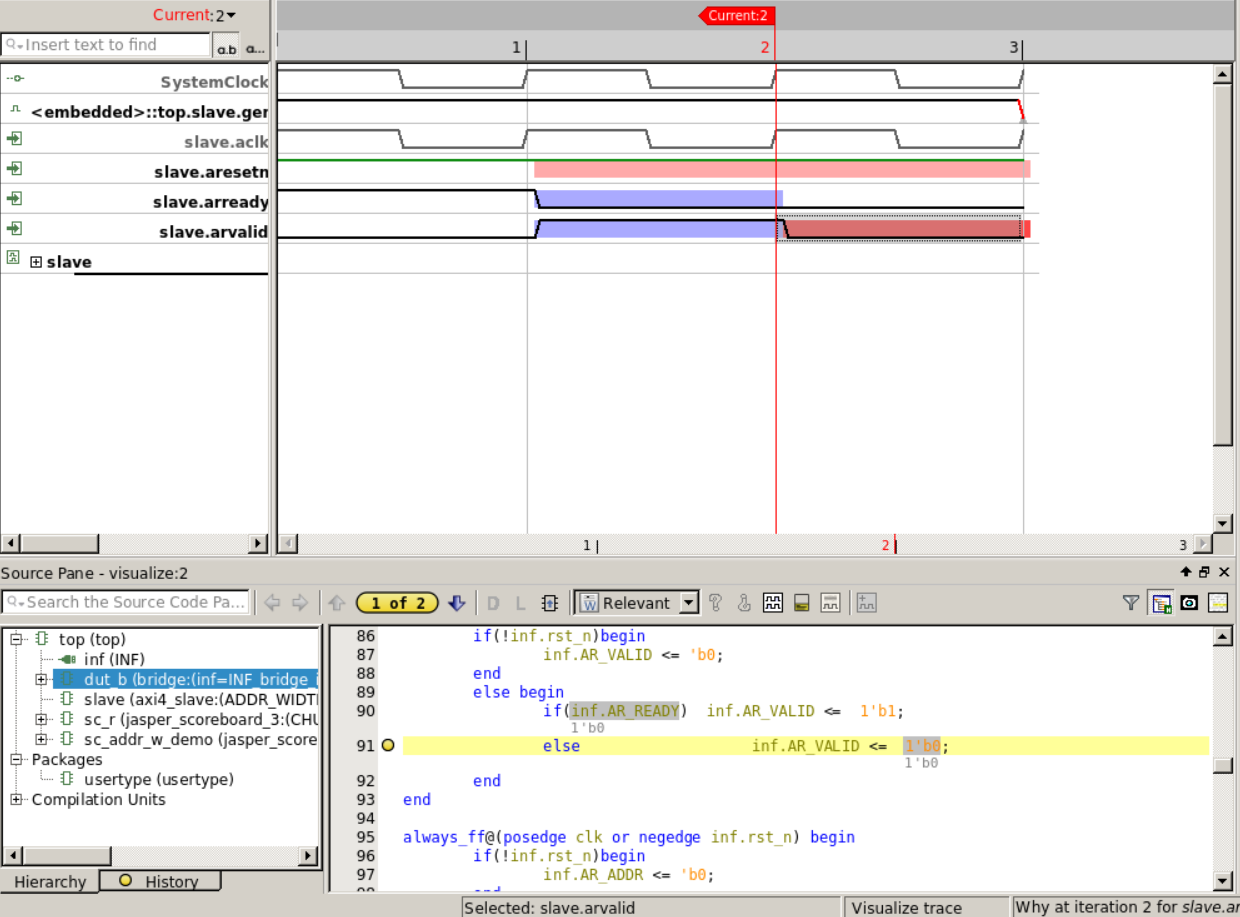
**Benefit:** It is formal optimized to reduce state-space complexity and reduce barrier for adoption.

1. List four bugs in Lab Exercise

What is the answer of the Lab Exercise?

**Error1:** AR\_VALID/AR\_ADDR should be stable if (AR\_VALID && !AR\_READY)

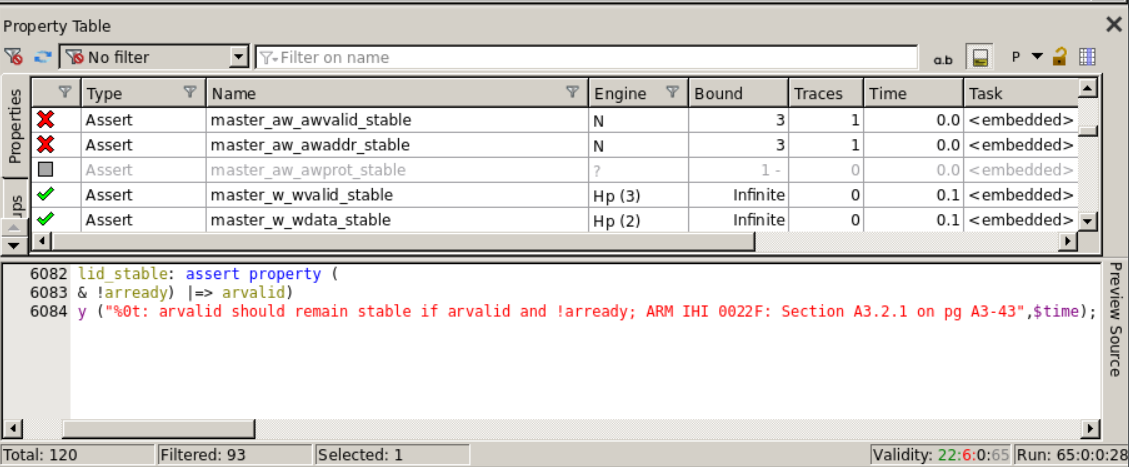


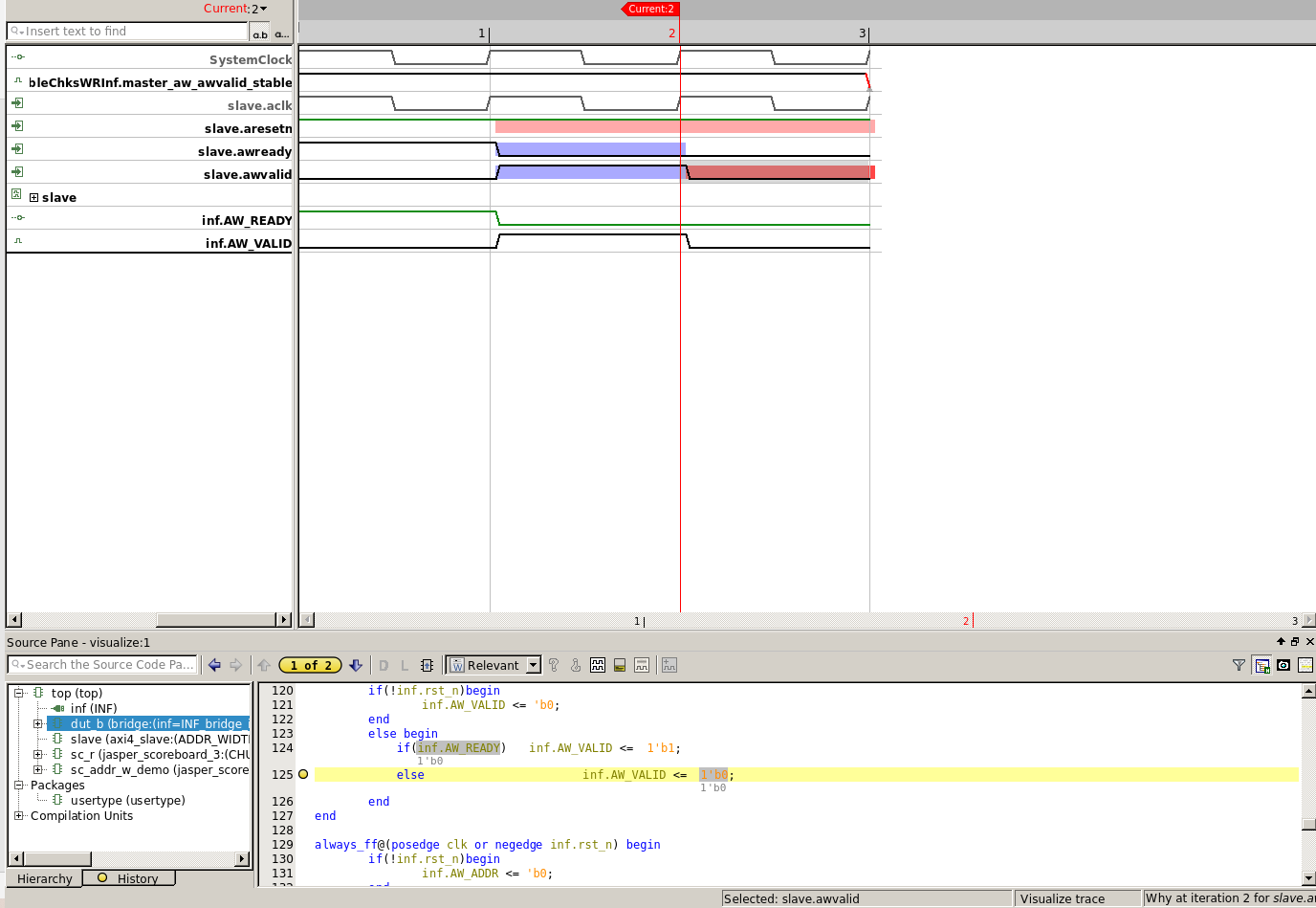


**Solution:**

Line 90: modify if statement-> if(**n\_state == AXI\_AR**)

**Error 2:** AW\_VALID/AW\_ADDR should be stable if (AW\_VALID && !AW\_READY)

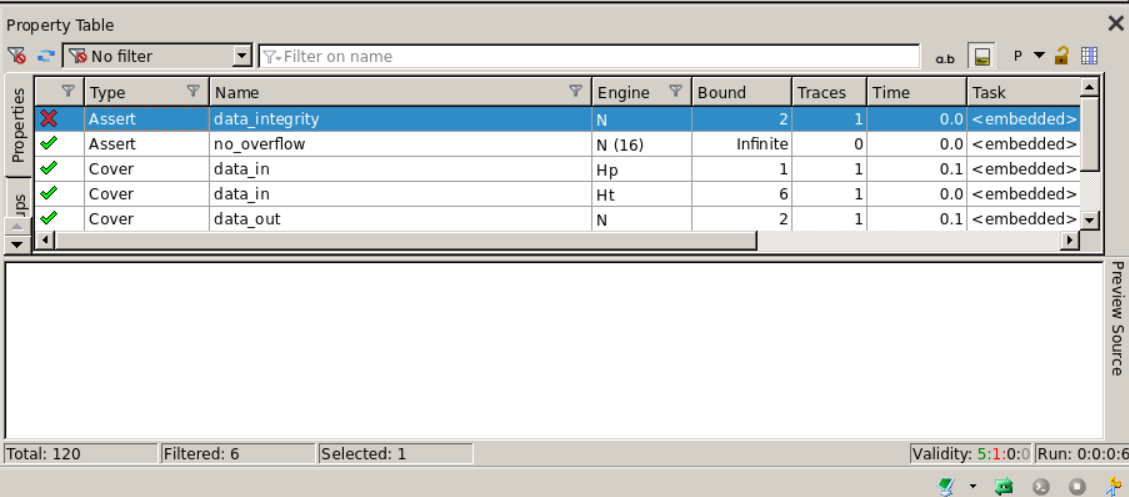


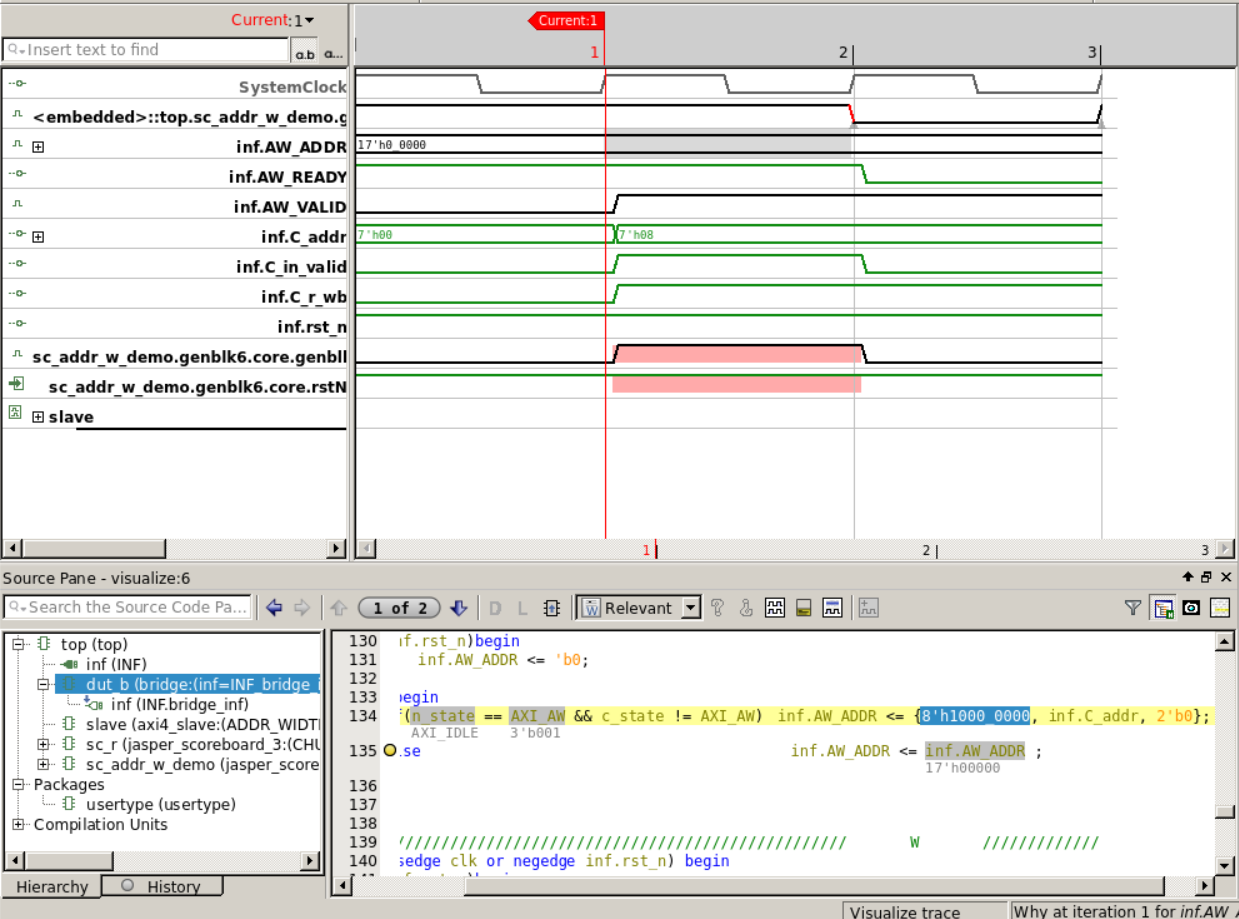


**Solution 2:**

Line 124: modify if statement-> if(**n\_state == AXI\_AW**)

**Error 3:** Address translation is fail.



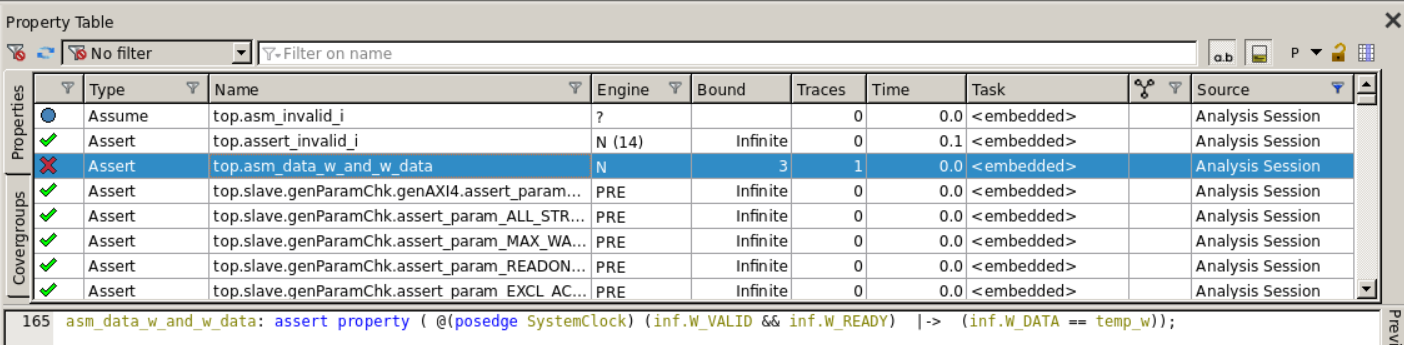


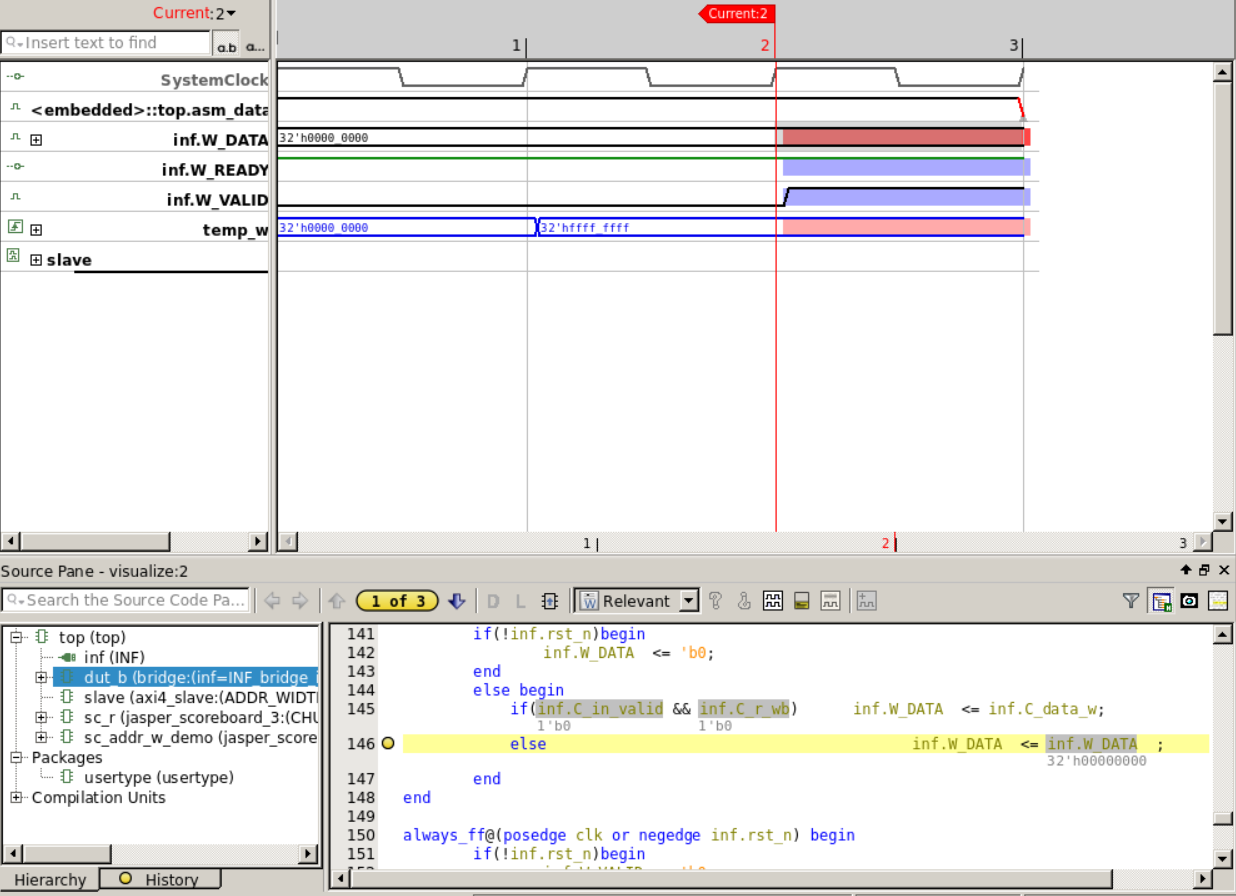
**Solution 3:**

Line 134:

if(n\_state == AXI\_AW && c\_state != AXI\_AW) inf.AW\_ADDR <= {**1'b1, 7'b0**, inf.C\_addr, 2'b0};

Error 4: inf.W\_DATA is not equal to temp\_w when (inf.W\_VALID && inf.W\_READY)



****

**Solution 4:**

Line 145: if(inf.C\_in\_valid && **!inf.C\_r\_wb**) inf.W\_DATA <= inf.C\_data\_w;