

Thursday, May 8, 2025

Exercise 1:

Jaspergold Warm-Up

a) I put 20250101

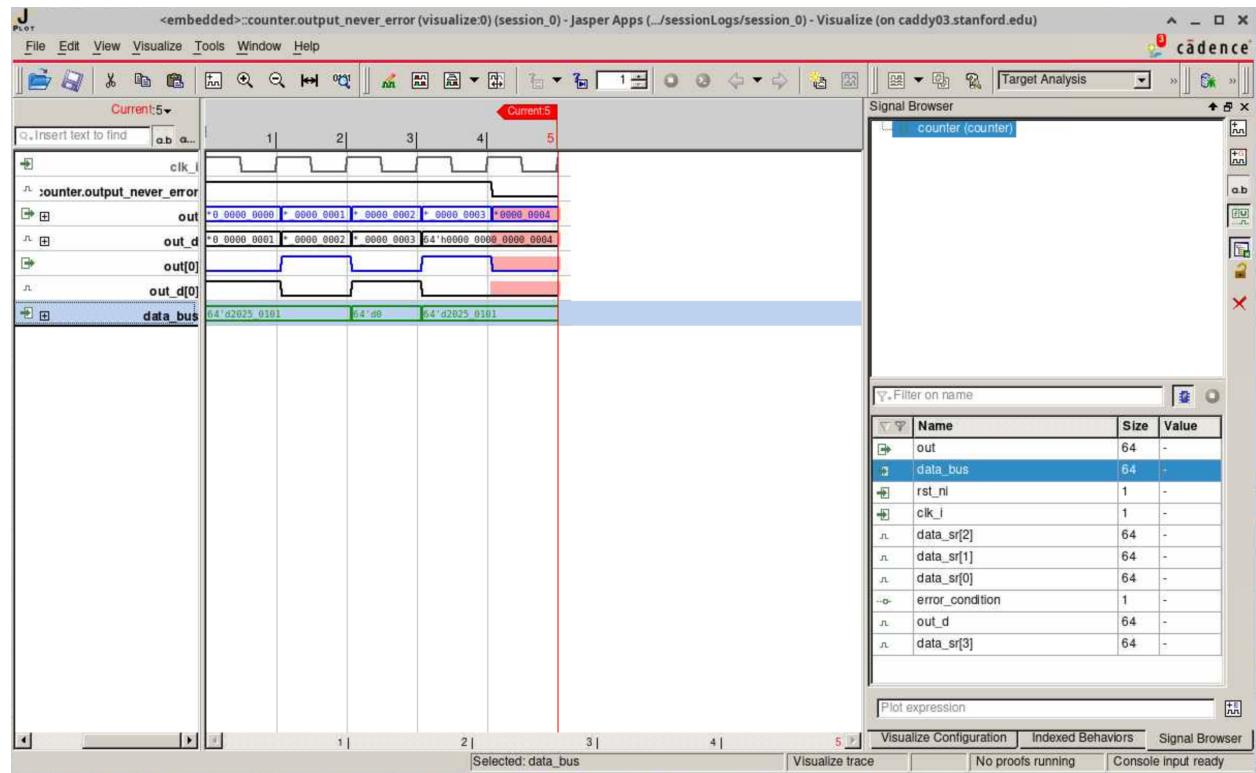
b)

"output_never_error" is an assertion, it will throw an error if condition "out_d == out + 1" fails. It checks whenever there is any signal changes.

"data_bus_nonzero" will trigger a coverage event when data_bus is not all zeros. It is meant to make sure the design handles meaningful data and not just idle states.

c)

Assert "output_never_error" fails.

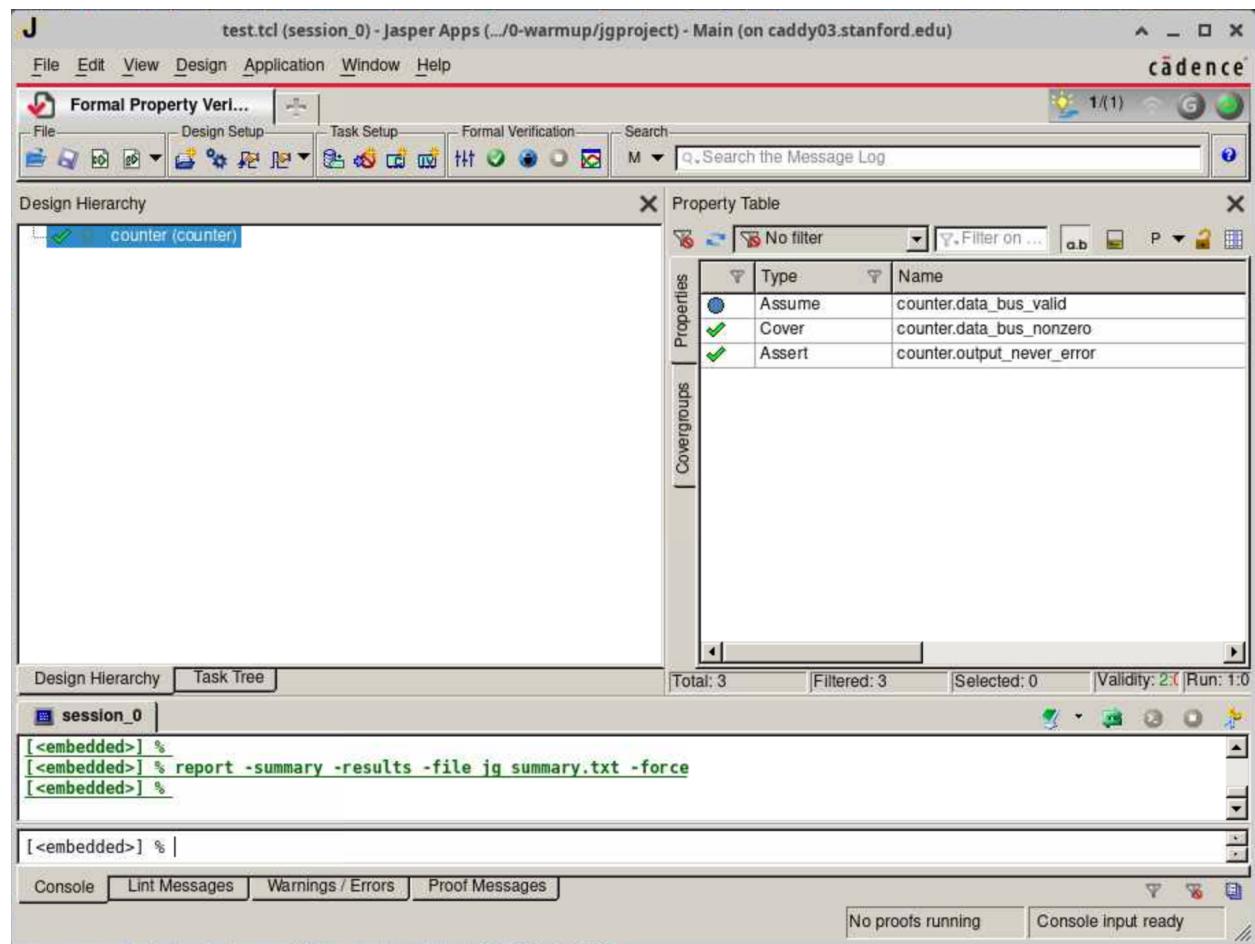


d)

"output_never_error" fails because out_d = out in 5th cycle, does not meet the condition that "out_d == out + 1".

e)

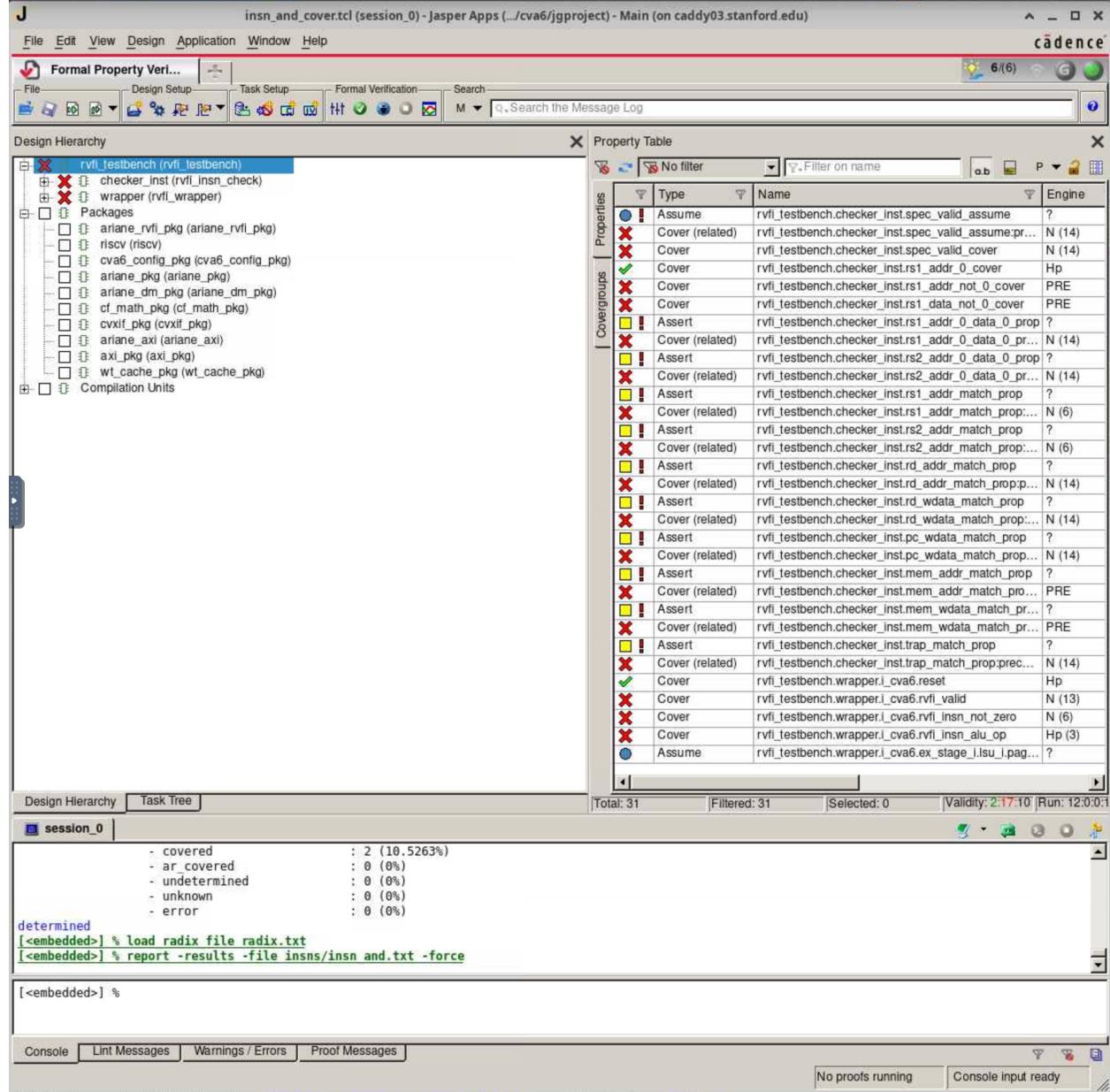
Added assume: data_bus_valid: assume(data_sr[3] ≠ `STUDENT_ID); such that assert only executes when the data_sr[3] is not equal to `STUDENT_ID, which means we always have out_d = out + 1.



Exercise 2:

Adding Formal Verification Signals

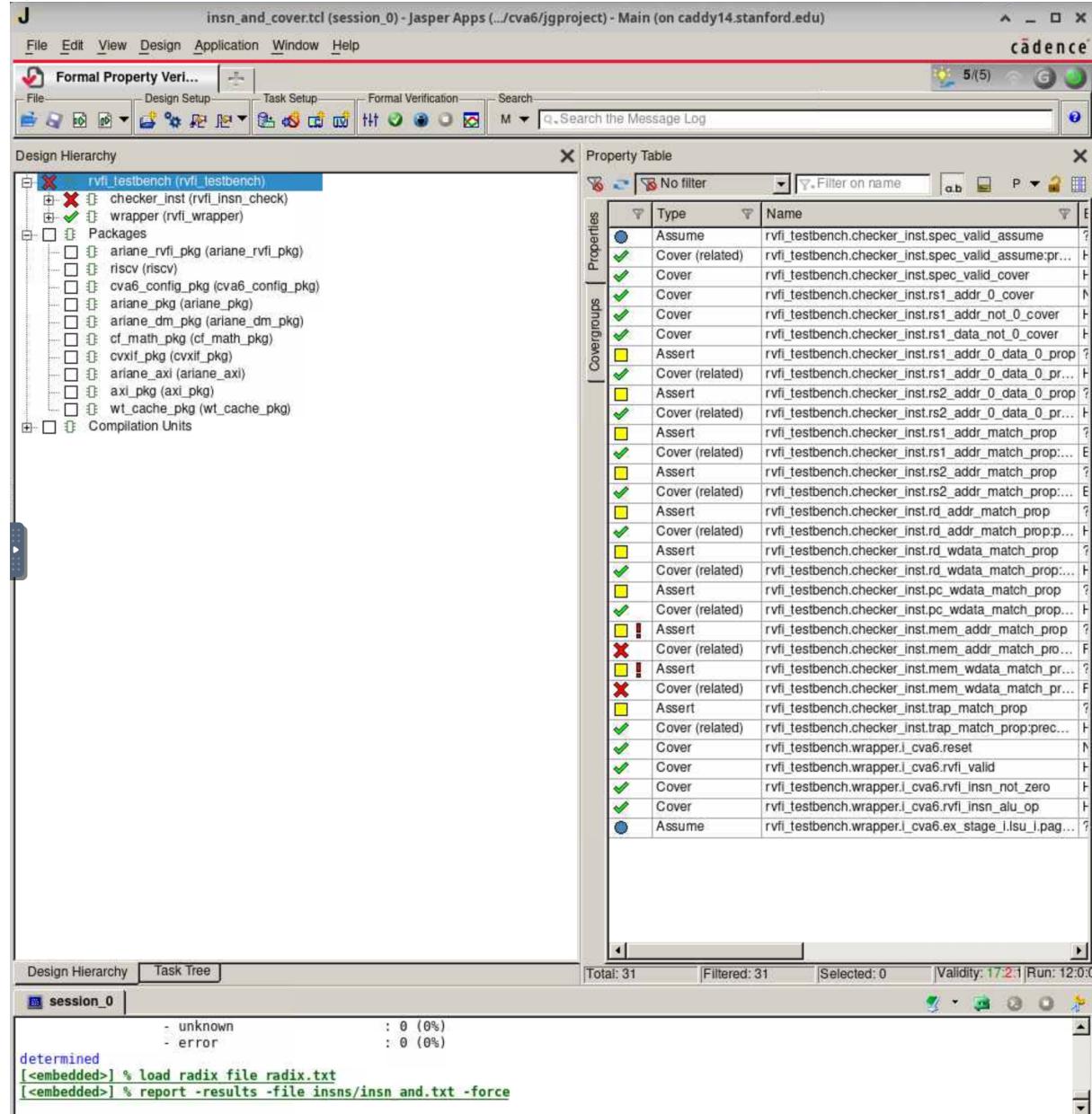
a)



```
// Cover Ensure that the processor is seeing a non-zero instruction
(JasperGold is sending nonzero instructions)
rvfi_insn_not_zero: cover((rvfi_o.insn != '0) & rvfi_o.valid);
rs1_addr_0_cover: cover(rvfi_i.rs1_addr == '0');
```

"rvfi_insn_not_zero" fails because JasperGold always send zero instructions. With zero instructions, rs1_addr is zero, therefore "rs1_addr_0_cover" passes.

b)



Most cover checks passed after passing jaspergold instruction to pipeline. However, there are still two covers that failed.

- checker_inst.mem_addr_match_prop
- checker_inst.mem_wdata_match_prop

Both are related covers created automatically by jaspergold. They failed because the condition associated with the assertion never reached. This is expected since we are checking insn_and.v which does not involve any memory read or write.

c)

- AND

The screenshot shows the Cadence Formal Property Verifier interface. The title bar reads "insn_and.tcl (session_0) - Jasper Apps (.../cva6/jgproject) - Main (on caddy09.stanford.edu)". The menu bar includes File, Edit, View, Design, Application, Window, Help, and a Cadence logo. The toolbar contains various icons for file operations, design setup, task setup, formal verification, and search. The main window has two panes: "Design Hierarchy" on the left and "Property Table" on the right.

Design Hierarchy:

- rvfi_testbench (rvfi_testbench)
 - checker_inst (rvfi_insn_check)
 - wrapper (rvfl_wrapper)
 - Packages
 - ariane_rvfi_pkg (ariane_rvfi_pkg)
 - riscv (riscv)
 - cva6_config_pkg (cva6_config_pkg)
 - ariane_pkg (ariane_pkg)
 - ariane_dm_pkg (ariane_dm_pkg)
 - cf_math_pkg (cf_math_pkg)
 - cvxif_pkg (cvxif_pkg)
 - ariane_axi (ariane_axi)
 - axi_pkg (axi_pkg)
 - wt_cache_pkg (wt_cache_pkg)
 - Compilation Units

Property Table:

Type	Name	Engine	Bound
Assume	rvfi_testbench.checker_inst.spec_valid_assume	?	
Cover (related)	rvfi_testbench.checker_inst.spec_valid_assume.pr...	Ht	2 - 5
Cover	rvfi_testbench.checker_inst.spec_valid_cover	Ht	2 - 5
Cover	rvfi_testbench.checker_inst.rs1_addr_0_cover	Ht	1
Cover	rvfi_testbench.checker_inst.rs1_addr_not_0_cover	Ht	2 - 4
Cover	rvfi_testbench.checker_inst.rs1_data_not_0_cover	Ht	2 - 4
Assert	rvfi_testbench.checker_inst.rs1_addr_0_data_0_prop	N (11)	Infinite
Cover (related)	rvfi_testbench.checker_inst.rs1_addr_0_data_0_pr...	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.rs2_addr_0_data_0_prop	I (16)	Infinite
Cover (related)	rvfi_testbench.checker_inst.rs2_addr_0_data_0_pr...	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.rs1_addr_match_prop	N (13)	Infinite
Cover (related)	rvfi_testbench.checker_inst.rs1_addr_match_prop...	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.rs2_addr_match_prop	N (13)	Infinite
Cover (related)	rvfi_testbench.checker_inst.rs2_addr_match_prop...	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.rd_addr_match_prop	N (13)	Infinite
Cover (related)	rvfi_testbench.checker_inst.rd_addr_match_prop...	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.rd_wdata_match_prop	N (21)	Infinite
Cover (related)	rvfi_testbench.checker_inst.rd_wdata_match_prop...	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.pc_wdata_match_prop	N (20)	Infinite
Cover (related)	rvfi_testbench.checker_inst.pc_wdata_match_prop...	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.mem_addr_match_prop	PRE	Infinite
Cover (related)	rvfi_testbench.checker_inst.mem_addr_match_pro...	PRE	Infinite
Assert	rvfi_testbench.checker_inst.mem_wdata_match_pr...	PRE	Infinite
CdAssert[ited]	rvfi_testbench.checker_inst.mem_wdata_match_pr...	PRE	Infinite
Assert	rvfi_testbench.checker_inst.trap_match_prop	N (3)	Infinite
Cover (related)	rvfi_testbench.checker_inst.trap_match_prop:prec...	Ht	2 - 5
Cover	rvfi_testbench.wrapper.i_cva6.reset	Ht	1
Cover	rvfi_testbench.wrapper.i_cva6.rvfl_valid	Ht	2 - 5
Cover	rvfi_testbench.wrapper.i_cva6.rvflInsn_not_zero	Ht	2 - 5
Cover	rvfi_testbench.wrapper.i_cva6.rvflInsnAluOp	Ht	2 - 5
Assume	rvfi_testbench.wrapper.i_cva6.ex_stage_i.lsu_i_pag...	?	

Task Tree:

- session_0
 - error : 0 (0%)

Message Log:

```
[<embedded>] % report -results -file insn_and.txt -force
```

Console:

```
[<embedded>] %
```

Bottom Status Bar:

- Console, Lint Messages, Warnings / Errors, Proof Messages
- No proofs running, Console input ready

Most cover checks passed except two covers that failed, both are related covers created automatically by jaspergold.

- checker_inst.mem_addr_match_prop
- checker_inst.mem_wdata_match_prop

They failed because the condition associated with the assertion never reached. This is expected since we are checking `insn_slli` which does not involve any memory read or write, and set `mem_rmask` and `mem_wmask = 0` by default. As a result, these checkers are never reached.

- SLLI

The screenshot shows the Jasper Apps interface with the title "insn_slli.tcl (session_0) - Jasper Apps (.../cva6/jgproject) - Main (on caddy05.stanford.edu)". The menu bar includes File, Edit, View, Design, Application, Window, Help, and Formal Verification. The toolbar has icons for Open, Save, New, Run, Stop, and others. The Design Hierarchy panel on the left shows the project structure with "rvfi_testbench (rvfi_testbench)" selected. The Property Table panel on the right displays a list of assertions and covers. The table has columns for Type, Name, Engine, and Bound. Most entries are green checkmarks, indicating success, while some are red X's or yellow exclamation marks, indicating failure. The table lists various assertions and covers related to memory access and trap conditions.

Type	Name	Engine	Bound
Assume	rvfi_testbench.checker_inst.spec_valid_assume	?	
Cover (related)	rvfi_testbench.checker_inst.spec_valid_assume:precondition1	Ht	2 - 5
Cover	rvfi_testbench.checker_inst.spec_valid_cover	Ht	2 - 5
Cover	rvfi_testbench.checker_inst.rs1_addr_0_cover	Ht	1
Cover	rvfi_testbench.checker_inst.rs1_addr_not_0_cover	Ht	2 - 4
Cover	rvfi_testbench.checker_inst.rs1_data_not_0_cover	Ht	2 - 4
Assert	rvfi_testbench.checker_inst.rs1_addr_0_data_0_prop	N (11)	Infin
Cover (related)	rvfi_testbench.checker_inst.rs1_addr_0_data_0_prop:precondition1	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.rs2_addr_0_data_0_prop	N	5
Cover (related)	rvfi_testbench.checker_inst.rs2_addr_0_data_0_prop:precondition1	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.rs1_addr_match_prop	N (13)	Infin
Cover (related)	rvfi_testbench.checker_inst.rs1_addr_match_prop:precondition1	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.rs2_addr_match_prop	PRE	Infin
Cover (related)	rvfi_testbench.checker_inst.rs2_addr_match_prop:precondition1	PRE	Infin
Assert	rvfi_testbench.checker_inst.rd_addr_match_prop	N (13)	Infin
Cover (related)	rvfi_testbench.checker_inst.rd_addr_match_prop:precondition1	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.rd_wdata_match_prop	N (18)	Infin
Cover (related)	rvfi_testbench.checker_inst.rd_wdata_match_prop:precondition1	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.pc_wdata_match_prop	N (21)	Infin
Cover (related)	rvfi_testbench.checker_inst.pc_wdata_match_prop:precondition1	Ht	2 - 5
Assert	rvfi_testbench.checker_inst.mem_addr_match_prop	PRE	Infin
Cover (related)	rvfi_testbench.checker_inst.mem_addr_match_prop:precondition1	PRE	Infin
Assert	rvfi_testbench.checker_inst.mem_wdata_match_prop	PRE	Infin
Cover (related)	rvfi_testbench.checker_inst.mem_wdata_match_prop:precondition1	PRE	Infin
Assert	rvfi_testbench.checker_inst.trap_match_prop	N (3)	Infin
Cover (related)	rvfi_testbench.checker_inst.trap_match_prop:precondition1	Ht	2 - 5
Cover	rvfi_testbench.wrapper.i_cva6.reset	Ht	1
Cover	rvfi_testbench.wrapper.i_cva6.rvfi_valid	Ht	2 - 5
Cover	rvfi_testbench.wrapper.i_cva6.rvfi_insn_not_zero	Ht	2 - 5
Cover	rvfi_testbench.wrapper.i_cva6Insn_alu_op	Ht	2 - 5
Assume	rvfi_testbench.wrapper.i_cva6_ex_stage_i_lsu_i.page_offset_assume	?	

Most cover checks passed except

(i) **Three related covers failed.** Again, they are created automatically by jaspergold and failed because the the condition associated with the assertion never reached.

- `checker_inst.mem_addr_match_prop`
- `checker_inst.mem_wdata_match_prop`

This is expected since we are checking `insn_slli` which does not involve any memory read or write, and set `mem_rmask` and `mem_wmask = 0` by default. As a result, these checkers are never reached.

- checker_inst.rs2_addr_match_prop

This is expected since insn_slli is an I-type instruction and set rs2_addr = 0 by default. As a result, this checker is never reached.

(ii) One assertion failed.

- checker_inst.rs2_addr_0_data_0_prop

This failure is caused by the fact that SLLI is an I-type instruction, whose 2nd operand is an immediate, not a register value.

The assertion fails because it requires rs2_data = 0 if rs2_addr = 0. Here SLLI set rs2_addr=0 by default. However, if we compare the ISA for R-type instruction and I-type instruction, rs2_data for R_type instruction is [24:20] whereas for I-type instruction, its immediate is located at [25:20]. Therefore, it is possible that rs2_data != 0 when SLLI executes.

```
// I-type instruction format (shift variation)
wire [`RISCV_FORMAL_ILEN-1:0] insn_padding = rvfi_i.insn >> 16 >> 16;
wire [6:0] insn_funct6 = rvfi_i.insn[31:26];
wire [5:0] insn_shamt = rvfi_i.insn[25:20];
wire [4:0] insn_rs1 = rvfi_i.insn[19:15];
wire [2:0] insn_funct3 = rvfi_i.insn[14:12];
wire [4:0] insn_rd = rvfi_i.insn[11: 7];
wire [6:0] insn_opcode = rvfi_i.insn[ 6: 0];

// R-type instruction format
wire [`RISCV_FORMAL_ILEN-1:0] insn_padding = rvfi_i.insn >> 16 >> 16;
wire [6:0] insn_funct7 = rvfi_i.insn[31:25];
wire [4:0] insn_rs2 = rvfi_i.insn[24:20];
wire [4:0] insn_rs1 = rvfi_i.insn[19:15];
wire [2:0] insn_funct3 = rvfi_i.insn[14:12];
wire [4:0] insn_rd = rvfi_i.insn[11: 7];
wire [6:0] insn_opcode = rvfi_i.insn[ 6: 0];
```

Above screenshots show the ISA for I-type and R-type instructions.

We plot the violence trace.



The first instruction (0x00209013) can be decoded as `slli x0, x1, 2`. It is clear that the immediate value is 2, which explains why we see 32'd2 in `rs2_data`, which triggers the assertional.

However, this is a false positive based on our explanation above.

- SB

The screenshot shows the Cadence Formal Property Verification interface. The main window displays a table titled "Property Table" under the "Coverage" tab. The table has columns for Type, Name, Engine, Bound, Traces, Time, and Task. The "Type" column includes entries like Assume, Cover (related), Cover, and Assert. The "Name" column lists various assertions and covers, such as `rvfi_testbench.checker_inst.spec_valid_assume`, `rvfi_testbench.checker_inst.spec_valid_cover`, and `rvfi_testbench.checker_inst.rs1_addr_0_cover`. The "Engine" column shows values like "?", "Ht", and "N". The "Bound" column shows ranges like "2 - 6" and "Infinite". The "Traces" column shows counts like "1" and "0". The "Time" column shows execution times like "0.0" and "2.4". The "Task" column shows "`<embedded>`". The "Properties" tab is selected on the left. The "Design Hierarchy" tab shows a tree view of the project structure, including packages like `ariane_rvfi_pkg`, `cav6_config_pkg`, `cav6_math_pkg`, `cavif_pkg`, `ct_math_pkg`, `ariane_axi`, `axi_pkg`, and `wt_cache_pkg`.

Type	Name	Engine	Bound	Traces	Time	Task
Assume	<code>rvfi_testbench.checker_inst.spec_valid_assume</code>	?		0	0.0	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst.spec_valid_assume:precondition1</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Cover	<code>rvfi_testbench.checker_inst.spec_valid_cover</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Cover	<code>rvfi_testbench.checker_inst.rs1_addr_0_cover</code>	Ht	1	1	2.1	<code><embedded></code>
Cover	<code>rvfi_testbench.checker_inst.rs1_addr_0_cover</code>	Ht	2 - 4	1	2.2	<code><embedded></code>
Cover	<code>rvfi_testbench.checker_inst.rs1_data_0_cover</code>	Ht	2 - 4	1	2.3	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst.rs1_addr_0_data_0_prop</code>	N (11)	Infinite	0	73.1	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst.rs1_addr_0_data_0_prop:precondition1</code>	Ht	2 - 6	1	2.5	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst.rs2_addr_0_data_0_prop</code>	N (11)	Infinite	0	208.9	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst.rs2_addr_0_data_0_prop:precondition1</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst.rs1_addr_match_prop</code>	N (12)	Infinite	0	588.0	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst.rs1_addr_match_prop:precondition1</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst.rs2_addr_match_prop</code>	N (12)	Infinite	0	83.8	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst.rs2_addr_match_prop:precondition1</code>	Ht	2 - 6	1	2.6	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst.rd_addr_match_prop</code>	N (17)	Infinite	0	25.0	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst_rd_addr_match_prop:precondition1</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst_rd_wdata_match_prop</code>	N (6)	1	4.9	<code><embedded></code>	
Cover (related)	<code>rvfi_testbench.checker_inst_rd_wdata_match_prop:precondition1</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst_pc_wdata_match_prop</code>	N (21)	Infinite	0	2931.2	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst_pc_wdata_match_prop:precondition1</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst.mem_addr_match_prop</code>	I	19 -	0	4210.6	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst.mem_addr_match_prop:precondition1</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst.mem_wdata_match_prop</code>	N (27)	Infinite	0	3014.6	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst.mem_wdata_match_prop:precondition1</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Assert	<code>rvfi_testbench.checker_inst.trap_match_prop</code>	N (3)	Infinite	0	0.1	<code><embedded></code>
Cover (related)	<code>rvfi_testbench.checker_inst.trap_match_prop:precondition1</code>	Ht	2 - 6	1	2.4	<code><embedded></code>
Cover	<code>rvfi_testbench.wrapper_i_cav6.reset</code>	Ht	1	1	2.1	<code><embedded></code>
Cover	<code>rvfi_testbench.wrapper_i_cav6.rvfi_valid</code>	Ht	2 - 5	1	2.1	<code><embedded></code>
Cover	<code>rvfi_testbench.wrapper_i_cav6.rvfi_insn_not_zero</code>	Ht	2 - 5	1	2.1	<code><embedded></code>
Cover	<code>rvfi_testbench.wrapper_i_cav6.rvfi_insn_alu_op</code>	Ht	2 - 5	1	2.1	<code><embedded></code>
Assume	<code>rvfi_testbench.wrapper_i_cav6.ex_stage_i_su_i_page_offset_assume</code>	?		0	0.0	<code><embedded></code>

Most cover checks passed except

- checker_inst.mem_addr_match_prop**: takes more than 10 hours but still not finished
- checker_inst.rd_wdata_match_prop** fails

This assertion checks whether the destination register (rd) receives expected results. However, SB does not have a destination register and by default it sets `rvfi_spec_o.rd_wdata = 0` (can be found in `insn/insn_sb.v`). That is why we found `rvfi_spec.rd_wdata = 0` in the check. On the other hand, in `core/store_unit.sv`, it sets `result_o = lsu_ctrl_i.data`. That is why we found `rvfi_i.rd_wdata` to be 0020 as shown in the trace.

This failure is expected since SB does not write to destination register anyway.

