Thursday, May 8, 2025

Exercise 1:

Jaspergold Warm-Up

a) I put 20250101

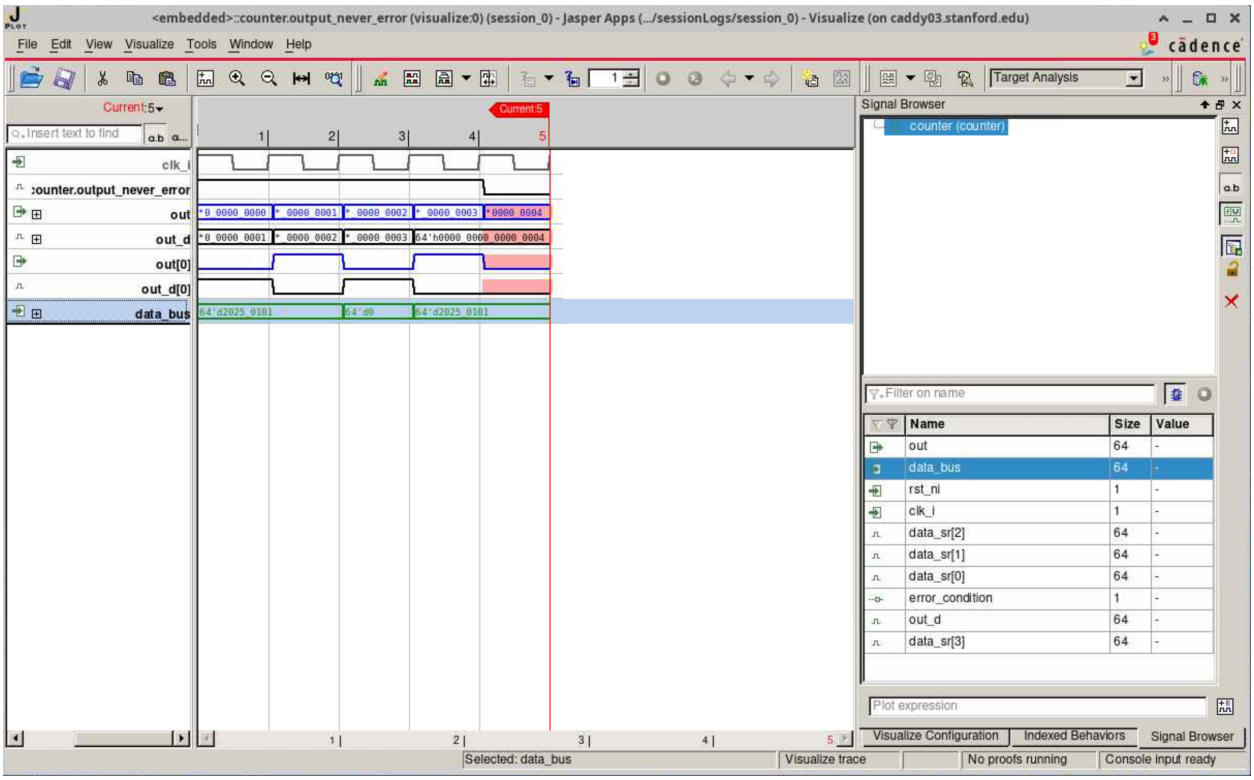
b)

“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.

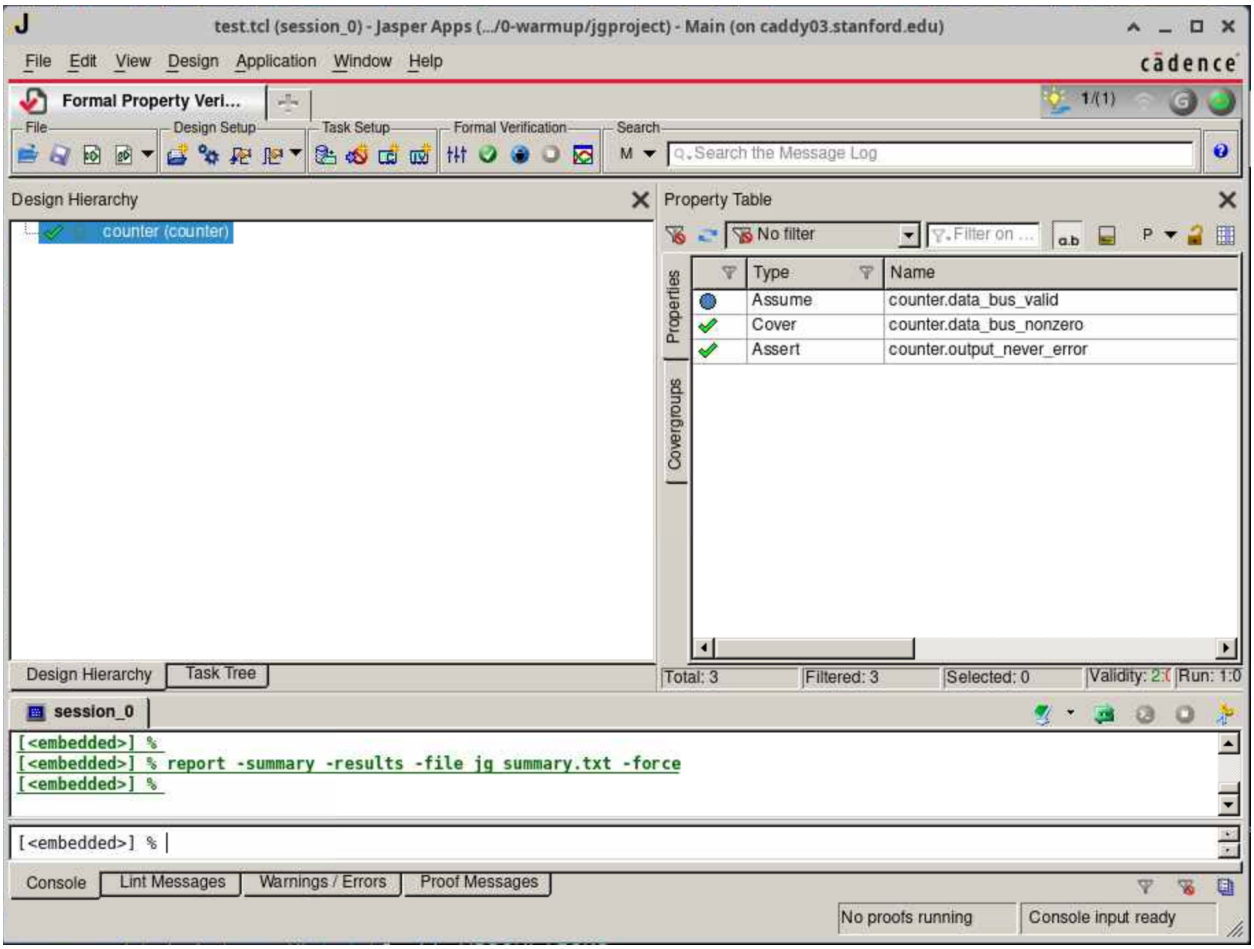


d)

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

e)

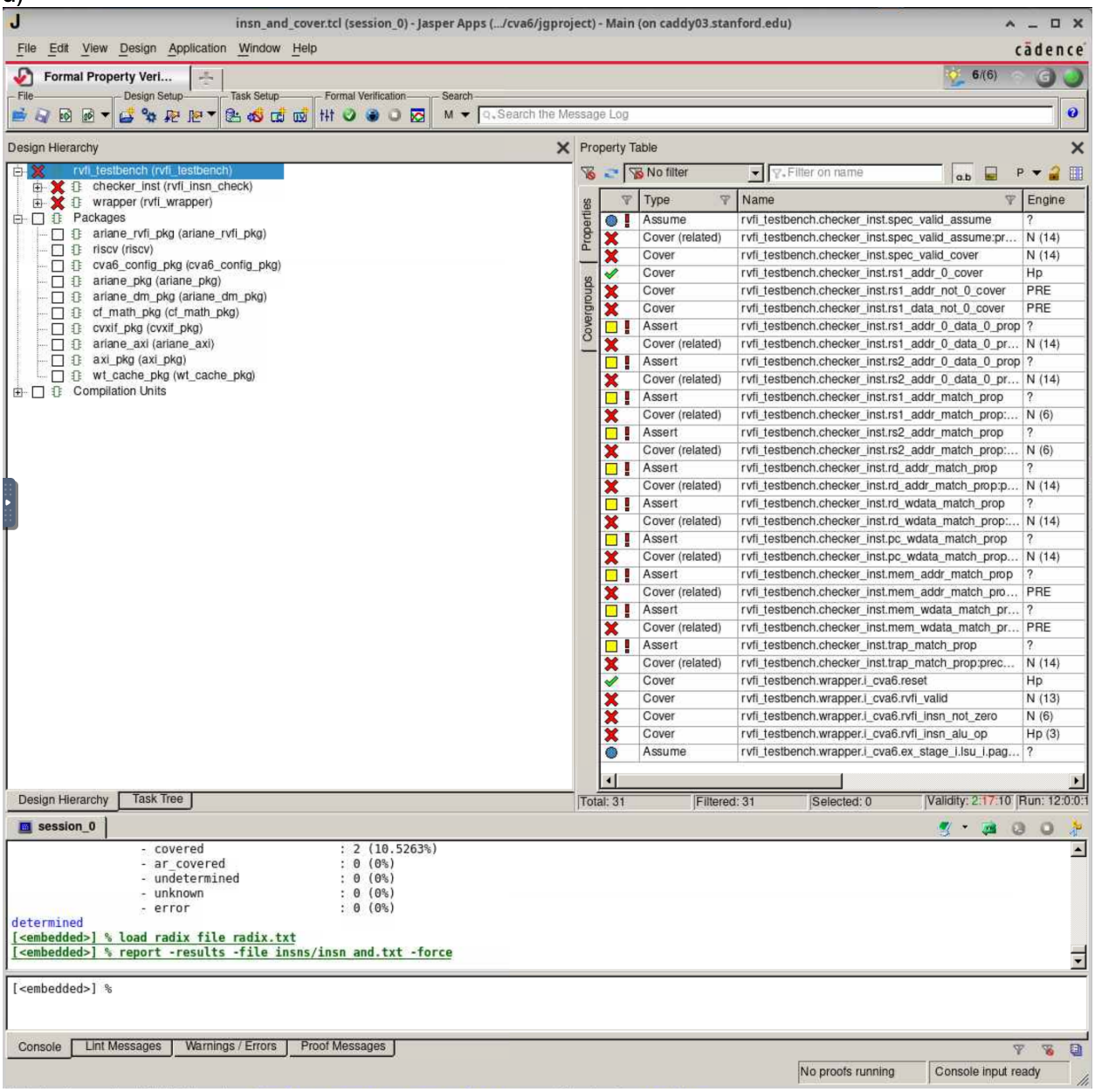
Added assume: `data_bus_valid: assume(data_sr[3] ≠ `STUDENT_ID);` such that assert only executes when the data_sr[3] is not equal to `STUDENT_ID, which means we always have out_d = out + 1.



Exercise 2:

Adding Formal Verification Signals

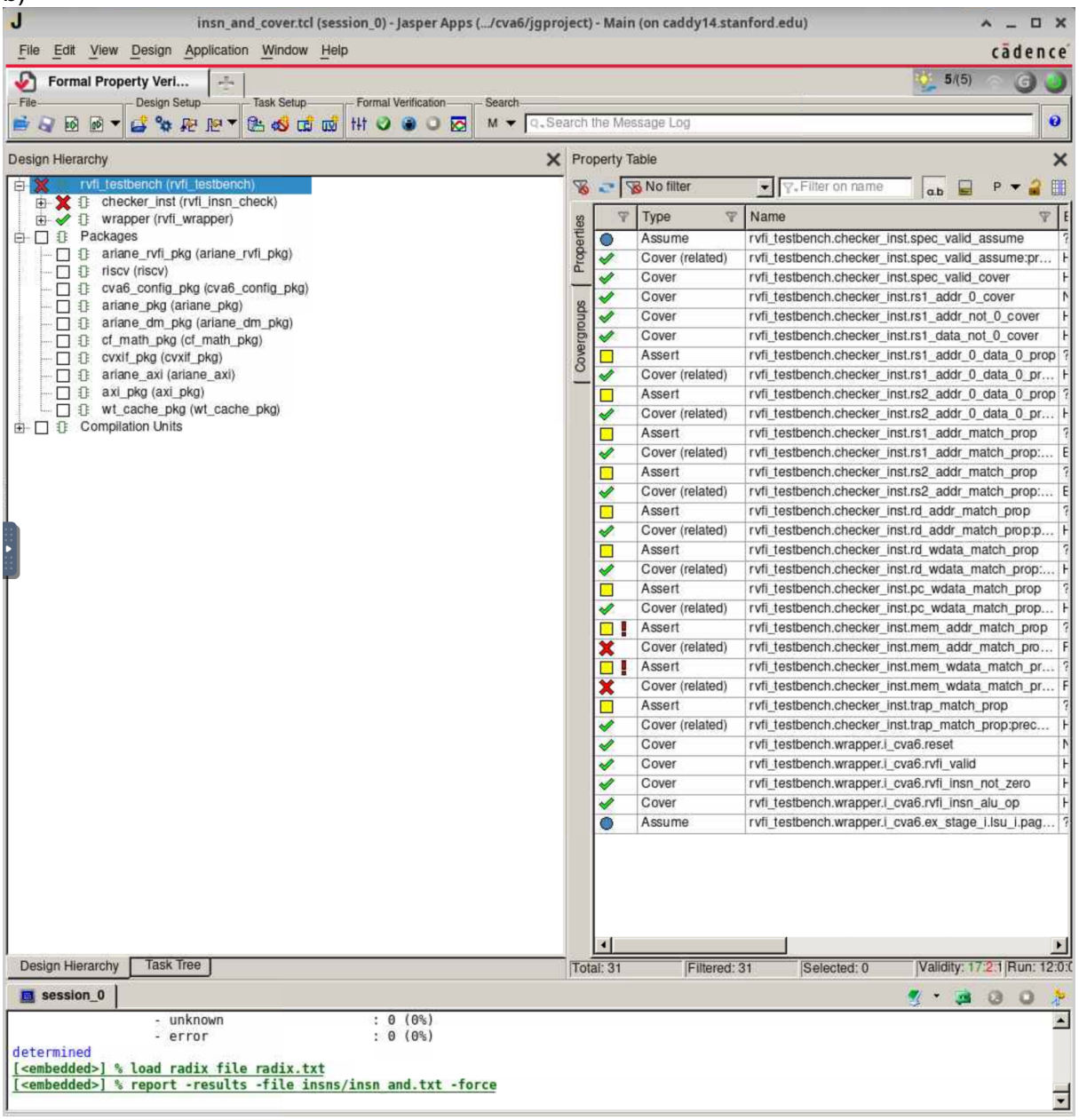
a)




```
// 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)



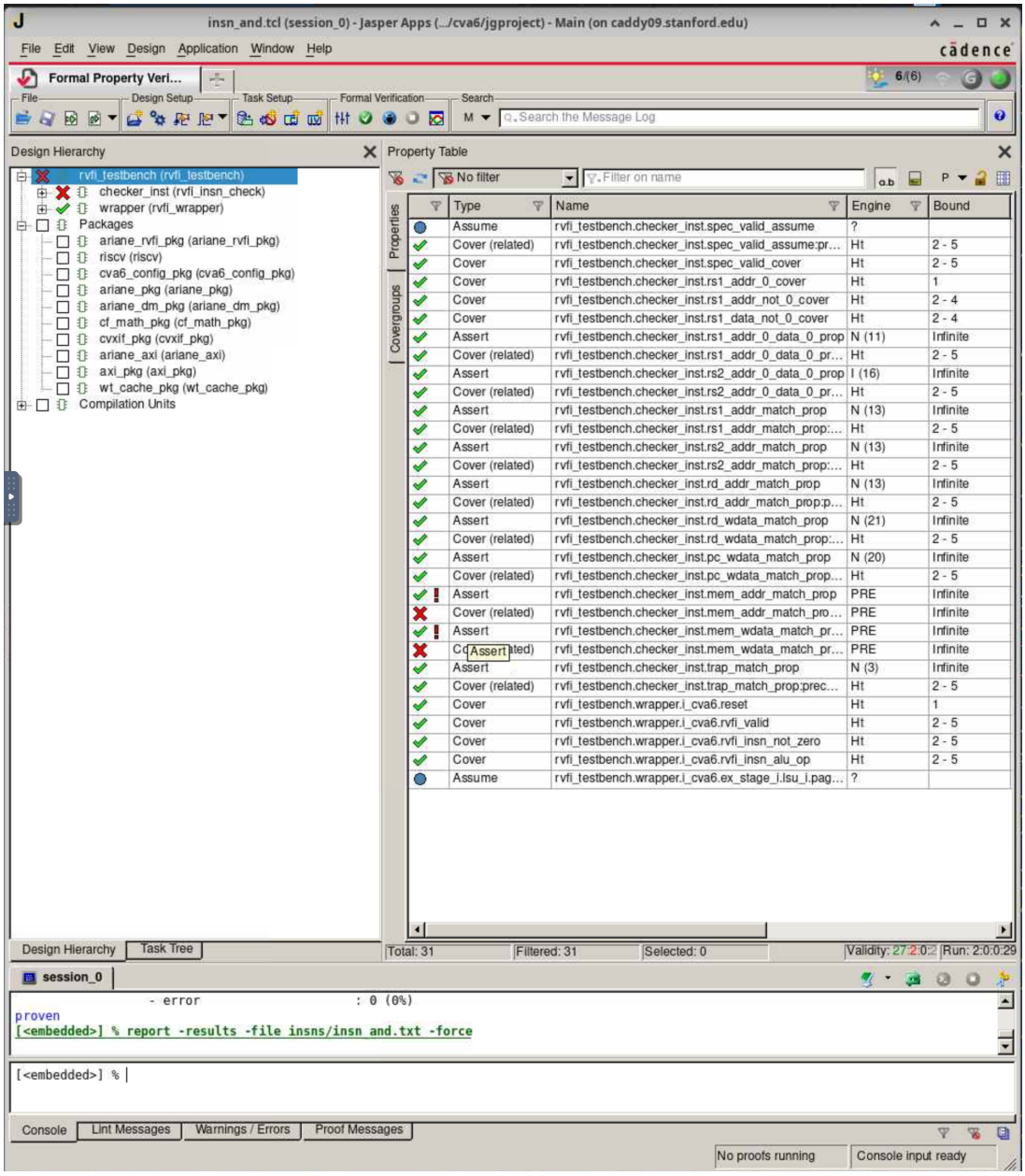
Most cover checks passed after passing jaspergold instruction to pipeline. However, there are still two covers that failed.

- `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

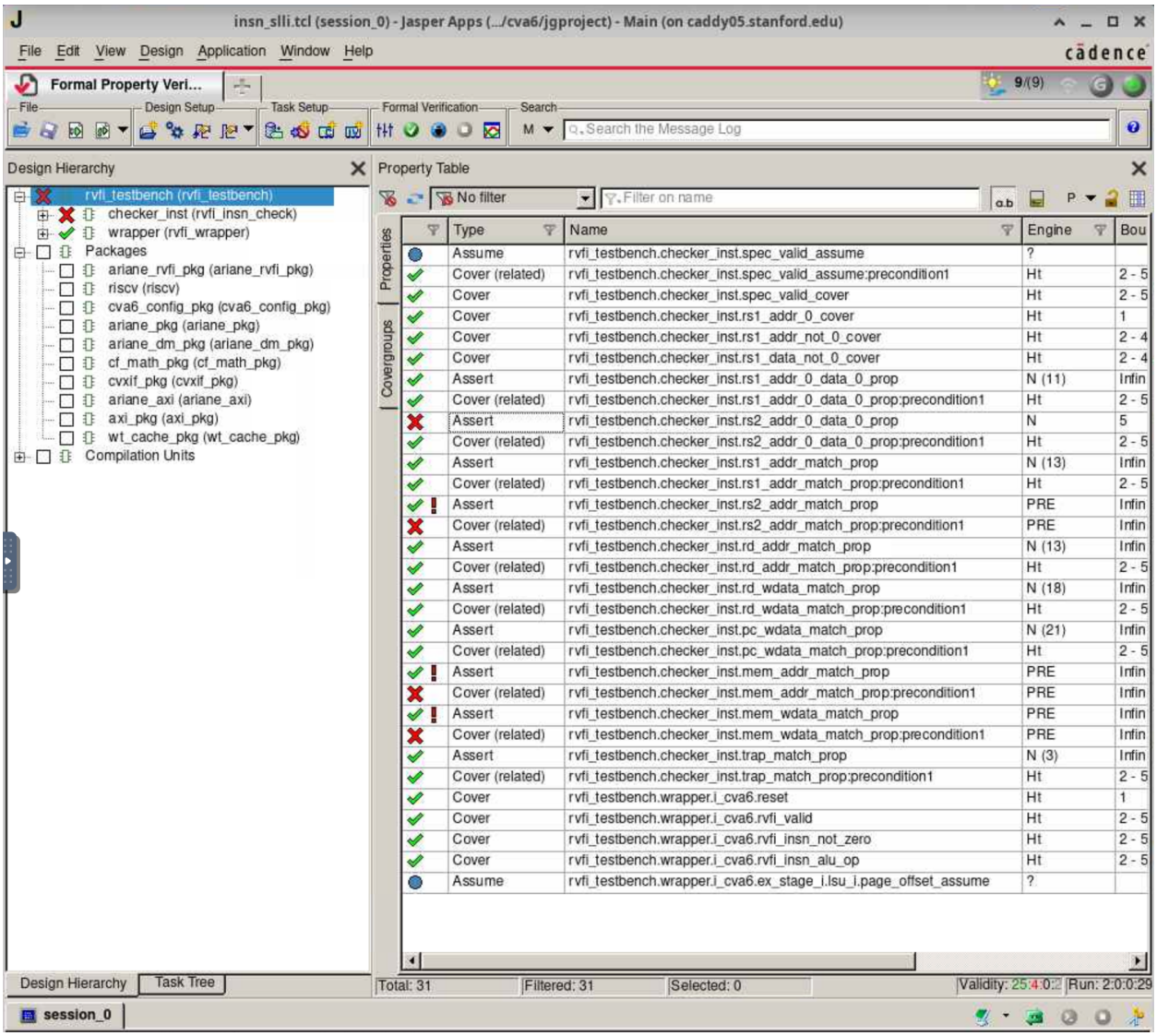


Most cover checks passed except two covers that failed, both are related covers created automatically by jaspergold.

- `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



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`

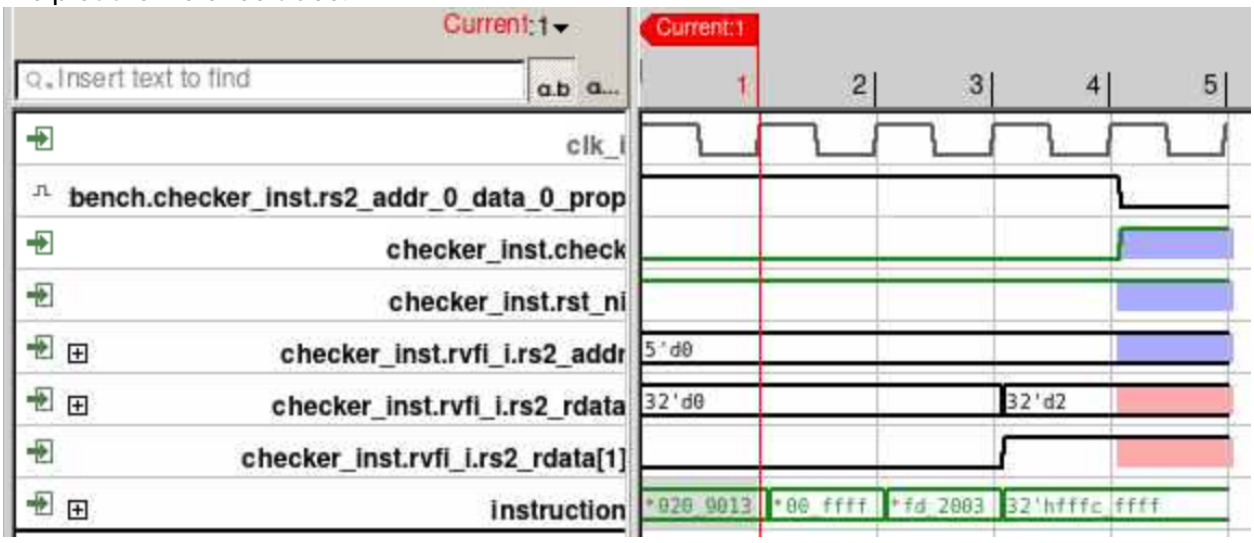
This failure is caused by the fact that SLLI is an I-type instruction, whose 2nd operand is an immediate, not a register value.
The assertion fails because it requires rs2_data = 0 if rs2_addr = 0. Here SLLI set rs2_addr=0 by default. However, if we compare the ISA for R-type instruction and I-type instruction, 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];
```

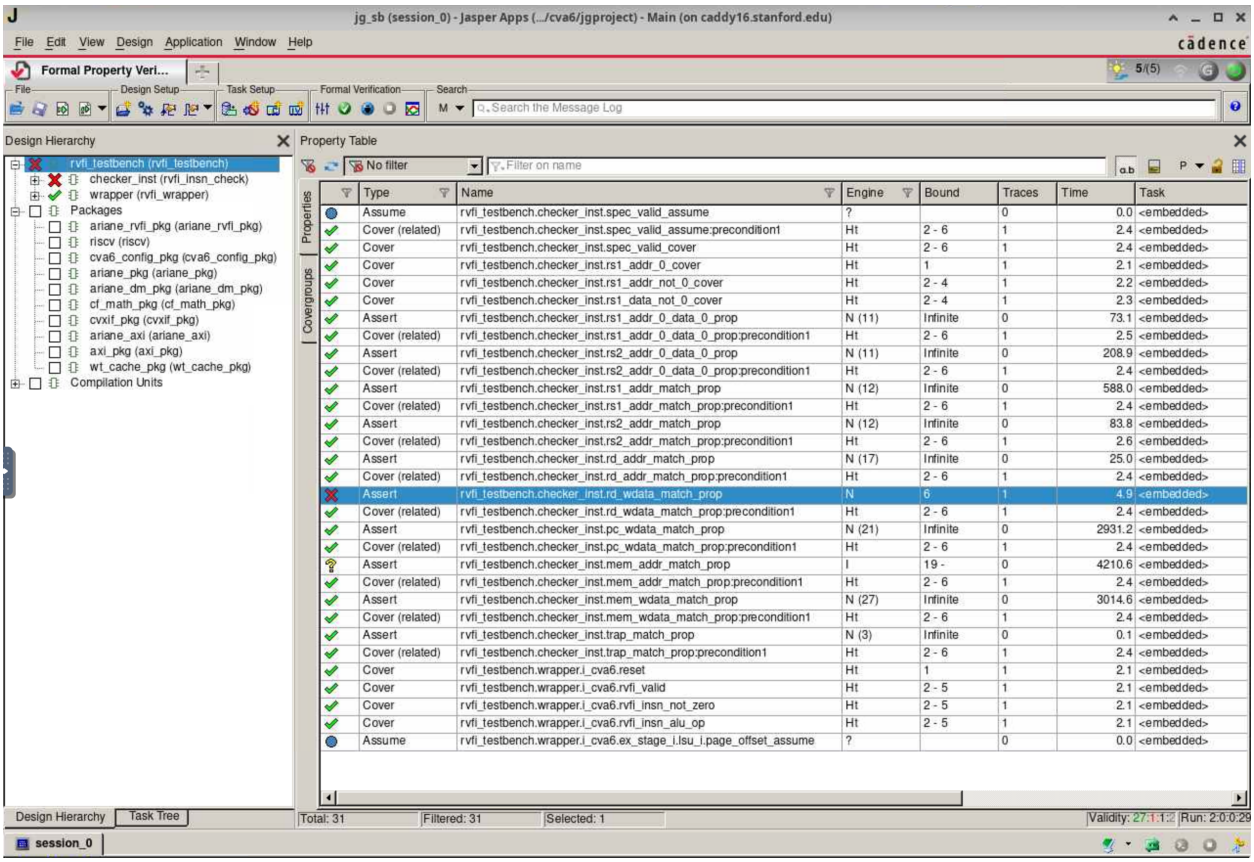
Above screenshots show the ISA for I-type and R-type instructions.

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.

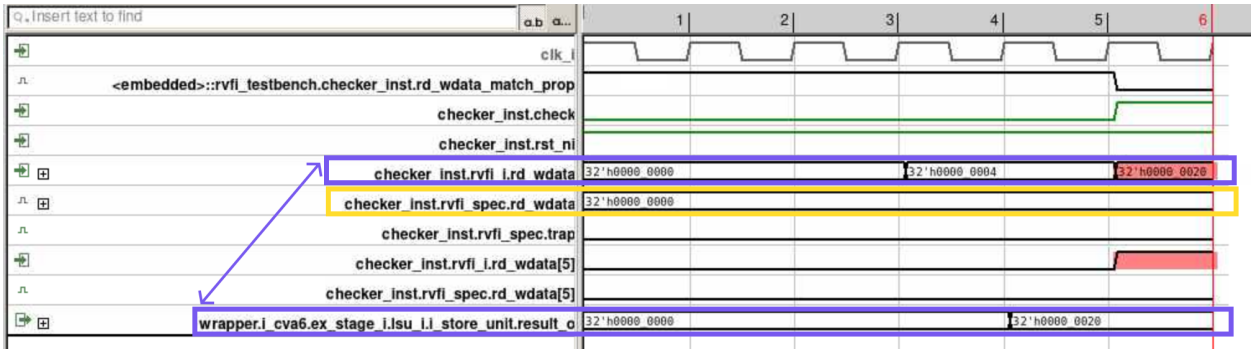
- SB



Type	Name	Engine	Bound	Traces	Time	Task
Assume	rvfi_testbench.checker_inst.spec_valid_assume	?		0	0.0	<embedded>
Cover (related)	rvfi_testbench.checker_inst.spec_valid_assume:precondition1	Ht	2 - 6	1	2.4	<embedded>
Cover	rvfi_testbench.checker_inst.spec_valid_cover	Ht	2 - 6	1	2.4	<embedded>
Cover	rvfi_testbench.checker_inst.rs1_addr_0_cover	Ht	1	1	2.1	<embedded>
Cover	rvfi_testbench.checker_inst.rs1_addr_not_0_cover	Ht	2 - 4	1	2.2	<embedded>
Cover	rvfi_testbench.checker_inst.rs1_data_not_0_cover	Ht	2 - 4	1	2.3	<embedded>
Assert	rvfi_testbench.checker_inst.rs1_addr_0_data_0_prop	N (11)	Infinite	0	73.1	<embedded>
Cover (related)	rvfi_testbench.checker_inst.rs1_addr_0_data_0_prop:precondition1	Ht	2 - 6	1	2.5	<embedded>
Assert	rvfi_testbench.checker_inst.rs2_addr_0_data_0_prop	N (11)	Infinite	0	208.9	<embedded>
Cover (related)	rvfi_testbench.checker_inst.rs2_addr_0_data_0_prop:precondition1	Ht	2 - 6	1	2.4	<embedded>
Assert	rvfi_testbench.checker_inst.rs1_addr_match_prop	N (12)	Infinite	0	588.0	<embedded>
Cover (related)	rvfi_testbench.checker_inst.rs1_addr_match_prop:precondition1	Ht	2 - 6	1	2.4	<embedded>
Assert	rvfi_testbench.checker_inst.rs2_addr_match_prop	N (12)	Infinite	0	83.8	<embedded>
Cover (related)	rvfi_testbench.checker_inst.rs2_addr_match_prop:precondition1	Ht	2 - 6	1	2.6	<embedded>
Assert	rvfi_testbench.checker_inst.rd_addr_match_prop	N (17)	Infinite	0	25.0	<embedded>
Cover (related)	rvfi_testbench.checker_inst.rd_addr_match_prop:precondition1	Ht	2 - 6	1	2.4	<embedded>
Assert	rvfi_testbench.checker_inst.rd_wdata_match_prop	N	6	1	4.9	<embedded>
Cover (related)	rvfi_testbench.checker_inst.rd_wdata_match_prop:precondition1	Ht	2 - 6	1	2.4	<embedded>
Assert	rvfi_testbench.checker_inst.pc_wdata_match_prop	N (21)	Infinite	0	2931.2	<embedded>
Cover (related)	rvfi_testbench.checker_inst.pc_wdata_match_prop:precondition1	Ht	2 - 6	1	2.4	<embedded>
Assert	rvfi_testbench.checker_inst.mem_addr_match_prop	I	19 -	0	4210.6	<embedded>
Cover (related)	rvfi_testbench.checker_inst.mem_addr_match_prop:precondition1	Ht	2 - 6	1	2.4	<embedded>
Assert	rvfi_testbench.checker_inst.mem_wdata_match_prop	N (27)	Infinite	0	3014.6	<embedded>
Cover (related)	rvfi_testbench.checker_inst.mem_wdata_match_prop:precondition1	Ht	2 - 6	1	2.4	<embedded>
Assert	rvfi_testbench.checker_inst.trap_match_prop	N (3)	Infinite	0	0.1	<embedded>
Cover (related)	rvfi_testbench.checker_inst.trap_match_prop:precondition1	Ht	2 - 6	1	2.4	<embedded>
Cover	rvfi_testbench.wrapper.i_cva6.reset	Ht	1	1	2.1	<embedded>
Cover	rvfi_testbench.wrapper.i_cva6.rvfi_valid	Ht	2 - 5	1	2.1	<embedded>
Cover	rvfi_testbench.wrapper.i_cva6.rvfi_insn_not_zero	Ht	2 - 5	1	2.1	<embedded>
Cover	rvfi_testbench.wrapper.i_cva6.rvfi_insn_alu_op	Ht	2 - 5	1	2.1	<embedded>
Assume	rvfi_testbench.wrapper.i_cva6.ex_stage_i.lsu_i.page_offset_assume	?		0	0.0	<embedded>

Most cover checks passed except
(i) **`checker_inst.mem_addr_match_prop`**: takes more than 10 hours but still not finished
(ii) **`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 insns/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.


Insert text to find
1
2
3
4
5
6
clk_i
<embedded>::rvfi_testbench.checker_inst.rd_wdata_match_prop
checker_inst.check
checker_inst.rst_ni
checker_inst.rvfi_i.rd_wdata
32'h0000_0000
32'h0000_0004
32'h0000_0020
checker_inst.rvfi_spec.rd_wdata
32'h0000_0000
checker_inst.rvfi_spec.trap
checker_inst.rvfi_i.rd_wdata[5]
checker_inst.rvfi_spec.rd_wdata[5]
wrapper.i_cva6.ex_stage_i.lsu_i.i_store_unit.result_o
32'h0000_0000
32'h0000_0020