IC Lab Formal Verification
Lab11 Quick Test
2023 Fall

Name: 葉舜良 **Student ID:** 312591037 **Account:** iclab126

(a) What is Formal verification?

A: Formal verification is to use mathematical logic and reasoning to prove or disprove the correctness of a system.

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

A: Formal exhaust all possible outcomes, while pattern simulation only visits limited outcomes.

And list the pros and cons for each.

A:

Formal verification is resource intensive if the search space is large for large block design yet gives a good confidence for the correctness of the design.

Pattern simulation is less resource intensive can quickly cover general case for testing however, might not be able to reach every possible states which might leads to system failure.

(b) What is glue logic?

A: Using auxiliary logic to observe and track events

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

A: Since it greatly simplify the SVA properties needed for assertion also it takes no extra price for insertion of glue logic.

(c) What is the difference between **Functional coverage** and **Code coverage**?

A:

Functional coverage: Determine if the user specified states conditions or sequences are covered.

Code coverage: Enumerates all possible statements for covering using brute force automatically.

What's the meaning of 100% code coverage, could we claim that our assertion is well enough

for verification? Why?

A: Means that the target you want to be measured are all covered, which gives enough confidence of the correctness of your circuit under these targets. No, if the assertions are incomplete or incorrect, the assertion might still not be well enough.

(d) 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.

A:

COI coverage: Does not need formal engines, does not require proof.

Proof coverage: Slower measurement and uses full and bounded proofs using formal engine, the subset of COI coverage.

(e) What are the roles of **ABVIP** and **scoreboard** separately?

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

A:

ABVIP: Verification IP with written assertions within to check the correctness of your circuit.

Scoreboard: Monitor IP which observe the input and output data of DUV.

(f) List four **bugs** in Lab Exercise

What is the answer of the Lab Exercise?

A:

Line 90: AR_VALID is not triggered by the handshake transaction, replace with correct handshake condition.

```
85  always_ff@(posedge clk or negedge inf.rst_n) begin
86      if(!inf.rst_n)begin
87          inf.AR_VALID <= 'b0;
88      end
89      else begin
90          if(n_state == AXI_AR)  inf.AR_VALID <=  1'b1;
91          else                   inf.AR_VALID <=  1'b0;
92      end
93  end
```

Line 123: AW_VALID is not triggered by the handshake transaction, replace condition with correct handshake condition.

```
119  always_ff@(posedge clk or negedge inf.rst_n) begin
120      if(!inf.rst_n)begin
121          inf.AW_VALID <= 'b0;
122      end
123      else begin
124          if(n_state == AXI_AW)    inf.AW_VALID <=  1'b1;
125          else                     inf.AW_VALID <=  1'b0;
126      end
127  end
```

Line 123: Data width of addr to AW_ADDR register is incorrect.

```
129  always_ff@(posedge clk or negedge inf.rst_n) begin
130      if(!inf.rst_n)begin
131          inf.AW_ADDR <= 'b0;
132      end
133      else begin
134          if(n_state == AXI_AW && c_state != AXI_AW)   inf.AW_ADDR <= {1'b1, 7'b0, inf.C_addr, 2'b0};
135          else                                        inf.AW_ADDR <= inf.AW_ADDR ;
136      end
137  end
```

Line 145: Should be write mode, not read mode for inf.C_r_wb.

```
140  always_ff@(posedge clk or negedge inf.rst_n) begin
141      if(!inf.rst_n)begin
142          inf.W_DATA  <= 'b0;
143      end
144      else begin
145          if(inf.C_in_valid && !inf.C_r_wb)      inf.W_DATA  <= inf.C_data_w;
146          else                                   inf.W_DATA  <= inf.W_DATA  ;
147      end
148  end
```

(g) Among the JasperGold tools (Formal Verification, SuperLint, Jasper CDC, IMC Coverage), which one have you found to be the most effective in your verification process? Please describe a specific scenario where you applied this tool, detailing how it benefited your workflow and any challenge you encountered while using it.

A:

I found SuperLint and IMC Coverage to be helpful, SuperLint helps spotting some coding error for my design, while IMC Coverage helps me know whether my pattern has good quality or not under some assertions without me actually tracing it, those are pretty powerful tools. I might try to use formal verification for my small project or design before writing out complex pattern for tracing to speedup development time.