ChipChat Tutorial Assignment
Part II: Debugging and Iterative Improvement

Tejas Attarde
LLM4ChipDesign

1 Introduction

This section documents the debugging and iterative refinement process used to correct the `sequence_detector` design generated using the ChipChat workflow. The goal was to make the design pass the **provided tutorial testbench without any modification**. Throughout this process, the testbench was treated as a fixed verification reference, and all corrections were applied exclusively to the design under test (DUT).

The debugging process required careful reasoning about reset semantics, Verilog event scheduling, and finite state machine (FSM) robustness. Three iterations were required before the final design successfully passed all test cases.

2 Iteration 1: Initial FSM Implementation

2.1 Observed Issue

After implementing the initial FSM-based sequence detector and running the provided testbench, the simulation failed with the following error:

```
Error: Cycle 8, Expected: 1, Got: 0
```

This error indicates that the output signal `sequence_found` was not asserted at the expected simulation cycle, even though the applied input stimulus matched the intended detection pattern.

2.2 Root Cause Analysis

The initial design used a **synchronous active-low reset**. However, the testbench deasserts the reset signal on a rising clock edge, as shown below:

```
@(posedge clk);
reset_n <= 1;
```

Because the reset was synchronous, the FSM remained in reset for one additional cycle. This caused the first valid input symbol to be ignored, shifting the entire input sequence by one cycle and preventing correct detection.

2.3 Fix Applied

The reset logic was updated to use an **asynchronous active-low reset**. This ensured that the FSM exited reset immediately when `reset_n` was deasserted, allowing the first input symbol to be sampled correctly.

2.4 Outcome

After applying this fix, the simulation was rerun. Although reset behavior was corrected, the testbench still failed at cycle 8, indicating that additional issues were present.

3 Iteration 2: Output Timing and Verilog Scheduling

3.1 Observed Issue

Despite correcting the reset behavior, the same failure persisted:

```
Error: Cycle 8, Expected: 1, Got: 0
```

This suggested that the FSM was logically reaching the detection condition, but the output signal was not visible at the time the testbench checked it.

3.2 Root Cause Analysis

The testbench checks the output signal immediately after a rising clock edge. In the DUT, the `sequence_found` signal was assigned using a non-blocking assignment inside a clocked `always` block.

Due to Verilog event scheduling semantics, non-blocking assignments update in the **NBA (Non-Blocking Assignment) region**, which occurs after the testbench performs its check. As a result, the output was still low when the testbench evaluated it.

3.3 Fix Applied

To address this issue, the output signal `sequence_found` was removed from the sequential block and instead generated using **combinational logic**. The signal was derived directly from the FSM state, the current input, and the reset signal, ensuring that it was visible in the same simulation cycle as the triggering clock edge.

3.4 Outcome

After this modification, detection timing improved significantly. However, intermittent failures were observed when unexpected input values occurred during intermediate FSM states.

4 Iteration 3: FSM Robustness and Error Recovery

4.1 Observed Issue

The FSM occasionally failed to recover correctly when an unexpected input value was encountered mid-sequence. In such cases, the FSM could remain stuck in an incorrect state, preventing future valid detections.

4.2 Root Cause Analysis

The FSM did not explicitly define transitions for all possible input values in each state. When an invalid input occurred, the FSM behavior became undefined, leading to non-deterministic operation.

4.3 Fix Applied

Explicit fallback transitions were added to the FSM. For any unexpected input, the FSM transitions back to the initial state. This guarantees deterministic behavior and ensures that the detector can always recover and correctly detect future valid sequences.

4.4 Outcome

After applying this fix, the design was recompiled and simulated. The testbench completed successfully and printed the following message:

```
All test cases passed.
```

5 Iteration Summary

Iteration	Issue Observed	Fix Applied	Result
1	Reset mismatch with testbench	Changed to asynchronous reset	Fail
2	Output not visible at check time	Generated output combinationally	Fail
3	FSM not robust to invalid inputs	Added fallback transitions	Pass

Table 1: Summary of debugging iterations

6 Final Verification

The final design was verified using the following commands:

```
iverilog -g2012 sequence_detector.v sequence_detector_tb.v
vvp a.out
```

The simulation completed successfully without any errors.

7 Conclusion

This debugging process highlights the importance of understanding reset semantics, Verilog event scheduling, and FSM completeness when designing hardware that must interface with external verification environments. By treating the testbench as immutable and iteratively refining the DUT, a robust and correct design was achieved.