1. Part II: Debugging Loop Documentation

This section documents the debugging and refinement process for the `sequence_detector` finite state machine (FSM) implemented in `chipchat_exampleB.ipynb`. The objective of this exercise was to demonstrate how simulation feedback was systematically used to identify design issues and iteratively improve the RTL implementation until the design passed all verification tests.

1.1 Iteration 1: Initial Implementation and Simulation Failure

Observed Issue

The initial FSM implementation compiled successfully, indicating that the syntax and overall structure of the design were valid. However, when the design was simulated using the provided testbench, the following failure message was observed:

```
FAIL: sequence not detected
```

This indicated that the FSM failed to assert the `detected` output after the input sequence `1011` was applied.

Commands Used

```
iverilog -g2012 -o simv sequence_detector.v tb_sequence_detector.v
vvp simv
```

Analysis

Upon inspection, the state transition logic appeared to be correct, and the FSM did reach the final state corresponding to the full input sequence. However, the `detected` signal was asserted combinationally inside an `always_comb` block. As a result, the output pulse was extremely short and not aligned with a clock edge.

Because the testbench samples outputs synchronously, this brief pulse was missed, leading to the observed failure despite partially correct FSM behavior.

1.2 Iteration 2: Registering the Output Signal

Fix Applied

To resolve this issue, the `detected` output was moved into a clocked `always_ff` block. The signal was registered and asserted when the FSM reached the final detection state and the appropriate input condition was met.

```
always_ff @(posedge clk) begin
   if (reset)
```

```
        detected <= 1'b0;
    else
        detected <= (state == S101 && in_bit);
end
```

Rationale

Registering the output ensures that the `detected` signal is asserted for one full clock cycle, making it reliably observable by the testbench and any downstream logic. This change also aligns the design with standard synchronous design practices and avoids glitches commonly associated with purely combinational outputs.

Commands Used

```
iverilog -g2012 -o simv sequence_detector.v tb_sequence_detector.v
vvp simv
```

1.3 Iteration 3: Final Verification

Verification Step

After applying the fix, the design was recompiled and simulated again to confirm that the issue had been fully resolved and that no new errors were introduced.

Final Output

```
All tests passed!
```

Outcome

The corrected FSM successfully detects the overlapping input sequence `1011`. The `detected` output is asserted for exactly one clock cycle, the design compiles with `iverilog -g2012`, and all verification tests pass.

1.4 Summary and Lessons Learned

This debugging process highlights a common pitfall in FSM design: asserting event-based outputs combinationally rather than synchronously. While the initial implementation captured the intended state transitions, simulation results revealed a subtle timing issue that required correction. By registering the output signal, the FSM behavior became robust, verifiable, and consistent with good synchronous digital design practices.