Debugging loop (Example B: sequence_detector)

Context

Goal: Implement sequence_detector that passes the provided testbench sequence_detector_tb.v using:

In [ ]:
```
iverilog -g2012 -o sequence_detector.out sequence_detector.v sequence_detector
vvp sequence_detector.out
```

The testbench checks sequence_found at specific cycles and expects a one-cycle pulse.

Iteration 1 — API / notebook execution issue (function not defined)

(a) What failed

Tried to call Claude with fallback, but the function wasn't defined in the notebook runtime.

Error snippet:

In [ ]:
```
NameError: name 'call_with_fallback' is not defined
```

(b) What I changed

Added the missing helper cell defining:

- call_claude()
- call_with_fallback()
- candidate model list

(c) Why this should help

Colab notebooks require that function definitions are executed before use. Defining and running the helper cell makes call_with_fallback() available.

Iteration 2 — LLM-generated RTL did not compile (syntax error)

(a) What failed

Claude produced RTL that failed a standalone compile.

Command:

In [ ]:
```
iverilog -g2012 -o /tmp/seq_compile.out sequence_detector.v
```

Error snippet:

In [ ]:
```
sequence_detector.v:13: syntax error
sequence_detector.v:13: error: malformed statement
```

(b) What I changed

Replaced the Claude output with a cleaned/synthesizable RTL skeleton that compiles under iverilog -g2012 (proper always_ff, correct port list, no unsupported constructs).

(c) Why this should help

Even if the model understands the logic, it sometimes outputs invalid SystemVerilog for Icarus. A minimal synthesizable structure ensures compilation succeeds so we can move to functional verification.

Iteration 3 — Compiles, but fails test (mismatch at Cycle 8)

(a) What failed

After writing cleaned RTL, compilation succeeded but simulation failed.

Command:

In [ ]:
```
iverilog -g2012 -o sequence_detector.out sequence_detector.v sequence_detector
vvp sequence_detector.out
```

Error snippet (from TB):

```
In [ ]: Error: Cycle 8, Expected: 1, Got: 0
```

(b) What I changed

Adjusted the detection logic multiple times (pattern window using current data and history), but it still failed at cycle 8. This indicated the issue was likely timing alignment (what cycle the TB expects the pulse relative to stimulus application).

(c) Why I expected it to help

A classic sequence detector either:

detects including the current symbol (“window ends at this cycle”), or

detects based on previous symbols (“window ends at previous cycle”). The persistent failure suggested my assumption about which window the TB uses was wrong.

Iteration 4 — Add tracing to derive ground truth from TB timing

(a) What failed

Still failing “Cycle 8 expected 1 got 0” even after changing the assumed pattern and window.

(b) What I changed

Created a traced copy of the testbench to print cycle, data, sequence_found, and reset_n at each posedge clk.

Commands:

```
In [ ]: iverilog -g2012 -o seq_dbg.out sequence_detector.v sequence_detector_tb_tracec
        vvp seq_dbg.out
```

Trace snippet (critical cycles):

```
In [ ]: TRACE cycle=5 data=000 found=0 reset_n=1
```

```
TRACE cycle=6 data=110 found=0 reset_n=1
TRACE cycle=7 data=110 found=0 reset_n=1
TRACE cycle=8 data=011 found=0 reset_n=1
...
Error: Cycle 8, Expected: 1, Got: 0
```

This proved:

- At cycle 8, the current input is 011
- The TB expects sequence_found==1 at that time
- Therefore, the TB's expected detection does not match "pattern ends at current data 011" in the way we assumed earlier.

(c) Why I expected it to help

Instead of guessing the intended pattern, tracing makes the TB timing explicit. That allows deriving the exact detection window the TB expects.

Iteration 5 — Final fix: "detect-before-shift" using previous samples (PASS)

(a) What failed previously

Detection logic that included the current data did not match TB's expectation at cycle 8.

(b) What I changed

Implemented detection based on previous samples only and updated history after computing sequence_found (detect-before-shift). This aligned the design with the TB's cycle-based checking.

Then reran with traced TB:

Commands:

```
In [ ]: iverilog -g2012 -o seq_dbg.out sequence_detector.v sequence_detector_tb_tracec
        vvp seq_dbg.out
```

Passing output snippet:

```
In [ ]: TRACE cycle=8 data=011 found=0 reset_n=1
```

```
TRACE cycle=9 data=101 found=1 reset_n=1
...
All test cases passed.
```

(c) Why I expected it to help

The traced output showed the testbench checks sequence_found in a way that corresponds to a registered pulse that appears one cycle after the last relevant input history is fully observed. Computing detection from stored history (previous cycles) and then shifting in the current data matches that timing.

Final verification (final passing run)

Run the official testbench:

In [ ]:
```
iverilog -g2012 -o sequence_detector.out sequence_detector.v sequence_detector
vvp sequence_detector.out
```

Expected result:

In [ ]:
```
All test cases passed.
```