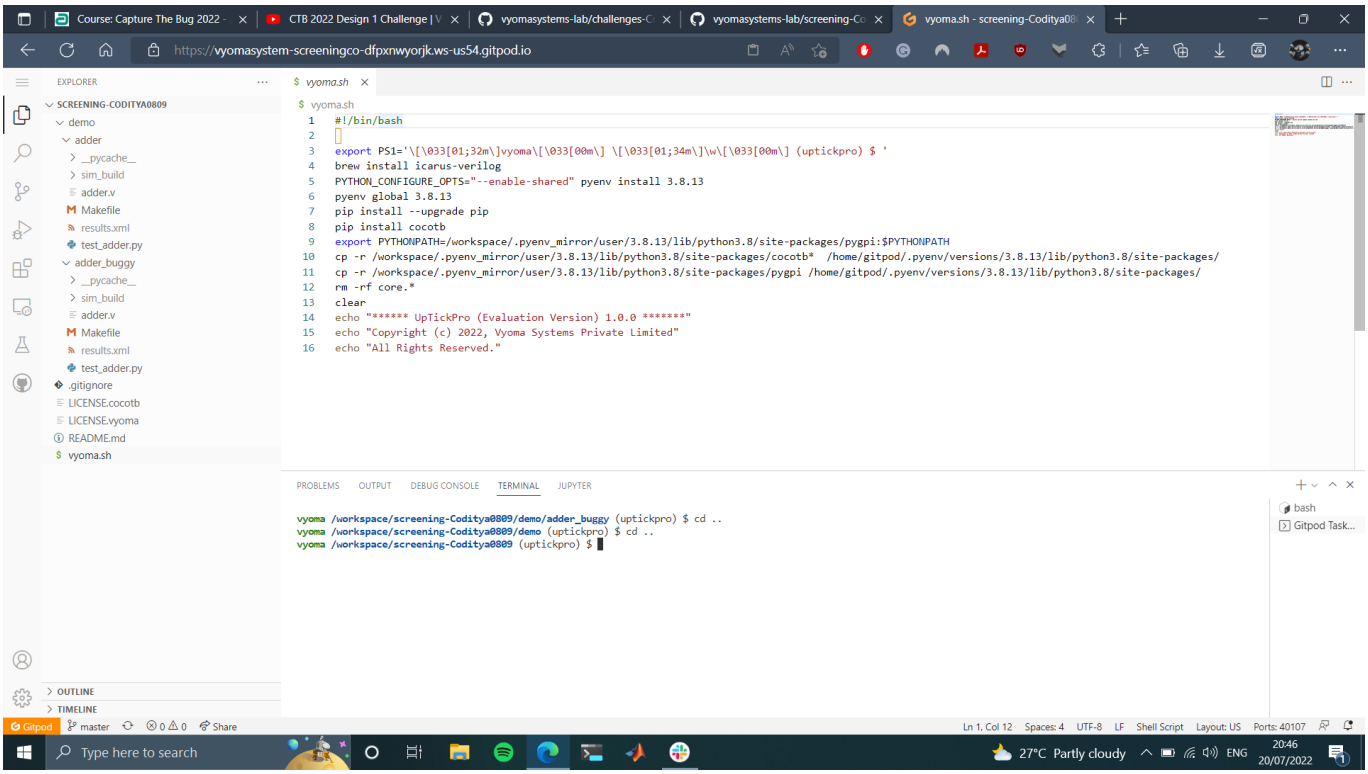
Level-1 Design-2 Verification

The verification environment was setup using Vyoma's UpTickPro provided for the hackathon.



Verification Environment

The CoCoTb based Python test was developed as explained. The test drives inputs to the Design Under Test (1011 Sequence detector module here) which takes in 1-bit input *inp_bit* and gives 1-bit output *seq_seen* which goes high when an input sequence of 1011 is detected in an overlapping sense (Overlapping a non-sequence).

Test 1 : Input 11011 not detected as a sequence

The values were assigned to the input port using

```
    test_seq = [1,1,0,1,1]
    dut.inp_bit.value = test_seq[0]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = test_seq[1]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = test_seq[2]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = test_seq[3]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = test_seq[4]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = 0               # driving the inp_bit to 0 to allow
another test case to execute, since this value is propagating to the next test and
```

```
changing the current_state value
    await Timer(1, units='ns')
```

The assert statement was used for comparing the mux's output to the expected value.

The following assert statement was used:

```
assert dut.seq_seen.value == 1, f'Sequence must be detected but is not detected.
Given sequence = {test_seq}. Model Output: {dut.seq_seen.value} Expected Ouput: 1'
```

Test Scenario 1 **(Important)**

- Test Inputs: An input test sequence 11011 was driven to the sequence detector
- Expected Output: seq_seen = 1
- Observed Output in the DUT dut.seq_seen.value = 0

Test 2 : Input 101011 not detected as a sequence

The values were assigned to the input port using

```
    test_seq = [1,0,1,0,1,1]
    dut.inp_bit.value = test_seq[0]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = test_seq[1]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = test_seq[2]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = test_seq[3]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = test_seq[4]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = test_seq[5]
    await FallingEdge(dut.clk)
    dut.inp_bit.value = 0             # driving the inp_bit to 0 to allow another
test case to execute, since this value is propagating to the next test and
changing the current_state value
    await Timer(1, units='ns')
```

The assert statement was used for comparing the sequence detector's output to the expected value.

The following assert statement was used:

```
assert dut.seq_seen.value == 1, f'Sequence must be detected but is not detected.
Given sequence = {test_seq}. Model Output: {dut.seq_seen.value} Expected Ouput: 1'
```

Test Scenario 2 **(Important)**

- Test Inputs: An input test sequence 101011 was driven to the sequence detector
- Expected Output: seq_seen = 1
- Observed Output in the DUT dut.seq_seen.value = 0

Failed Test Cases

```
vyoma /workspace/challenges-Coditya0809/level1_design2 (uptickpro) $ make
make -f Makefile results.xml
make[1]: Entering directory '/workspace/challenges-Coditya0809/level1_design2'
MODULE=test_seq_detect_1011      TESTCASE= TOPLEVEL=seq_detect_1011          TOPLEVEL_LANG=verilog \
         /home/linuxbrew/.linuxbrew/bin/vvp -M /workspace/.pyenv_mirror/user/3.8.13/lib/python3.8/site-packages/cocotb/libs -m libcocotbvpi_icarus   sim_build/sim.vvp
     -.--ns INFO     cocotb.gpi                         ..mbed/gpi_embed.cpp:76   in set_program_name_in_venv        Did not detect Python virtual environment. Using sy
stem-wide Python interpreter
     -.--ns INFO     cocotb.gpi                         ../gpi/GpiCommon.cpp:99   in gpi_print_registered_impl       VPI registered
     0.00ns INFO     Running on Icarus Verilog version 11.0 (stable)
     0.00ns INFO     Running tests with cocotb v1.6.2 from /workspace/.pyenv_mirror/fakeroot/versions/3.8.13/lib/python3.8/site-packages/cocotb
     0.00ns INFO     Seeding Python random module with 1658580244
     0.00ns WARNING  Pytest not found, assertion rewriting will not occur
     0.00ns INFO     Found test test_seq_detect_1011.test_seq_bug1
     0.00ns INFO     Found test test_seq_detect_1011.test_seq_bug2
     0.00ns INFO     running test_seq_bug1 (1/2)
 15000.00ns INFO     #### CTB: Develop your test here! ######
 65001.00ns INFO     test_seq_bug1 failed
                     Traceback (most recent call last):
                       File "/workspace/challenges-Coditya0809/level1_design2/test_seq_detect_1011.py", line 42, in test_seq_bug1
                         assert dut.seq_seen.value == 1, f'Sequence must be detected but is not detected. Given sequence = {test_seq}. Model Output: {dut.seq_seen.value
} Expected Ouput: 1'
                     AssertionError: Sequence must be detected but is not detected. Given sequence = [1, 1, 0, 1, 1]. Model Output: 0 Expected Ouput: 1
 65001.00ns INFO     running test_seq_bug2 (2/2)
 80001.00ns INFO     #### CTB: Develop your test here! ######
140002.00ns INFO     test_seq_bug2 failed
                     Traceback (most recent call last):
                       File "/workspace/challenges-Coditya0809/level1_design2/test_seq_detect_1011.py", line 74, in test_seq_bug2
                         assert dut.seq_seen.value == 1, f'Sequence must be detected but is not detected. Given sequence = {test_seq}. Model Output: {dut.seq_seen.value
} Expected Ouput: 1'
                     AssertionError: Sequence must be detected but is not detected. Given sequence = [1, 0, 1, 0, 1, 1]. Model Output: 0 Expected Ouput: 1
140002.00ns INFO     ********************************************************************************************
                     ** TEST                            STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
                     ********************************************************************************************
                     ** test_seq_detect_1011.test_seq_bug1   FAIL       65001.00           0.00   30557493.67  **
                     ** test_seq_detect_1011.test_seq_bug2   FAIL       75001.00           0.00   38433353.51  **
                     ********************************************************************************************
                     ** TESTS=2 PASS=0 FAIL=2 SKIP=0                  140002.00           0.02    8266501.82  **
                     ********************************************************************************************

make[1]: Leaving directory '/workspace/challenges-Coditya0809/level1_design2'
vyoma /workspace/challenges-Coditya0809/level1_design2 (uptickpro) $
```

Output mismatches for the above input sequence proving that there is a design bug

Design Bug

Based on the above test inputs and analysing the design, we see the following

```
case(current_state)
    IDLE:
    begin
    if(inp_bit == 1)
        next_state = SEQ_1;
    else
        next_state = IDLE;
    end
    SEQ_1:
    begin
    if(inp_bit == 1)
        next_state = SEQ_1;
    else
        next_state = IDLE;      <==== BUG, must remain in SEQ_1
    end
    SEQ_10:
```

```
        begin
        if(inp_bit == 1)
            next_state = SEQ_101;
        else
            next_state = IDLE;
        end
        SEQ_101:
        begin
        if(inp_bit == 1)
            next_state = SEQ_1011;
        else
            next_state = IDLE;      <==== BUG, must change to SEQ_10
        end
        SEQ_1011:
        begin
        next_state = IDLE;
        end
        endcase
```

For the seq_detect_1011 design, the logic for case SEQ_1 when *inp_bit* is 1, changes state to *IDLE*. Therefore, when the sequence occurs after *odd number* of ones '1', the sequence, although occured, does not get detected.

Also, another bug in the logic is for the case SEQ_101 when *inp_bit* is 0, changes state to *IDLE*. Therefore, when the suquence occurs after the combination '10', it will not be detected.

The design was fixed by simply, reatining the state `SEQ_1` when inp_bit `1` occurs and for `SEQ_101` when inp_bit is `0`, the state is limited to change to `SEQ_10`, since a `10` has already been detected of the sequence `1011`.

Design Fix

Updating the design and re-running the test makes the test pass.

```
make[1]: Leaving directory '/workspace/challenges-Coditya0809/level1_design2'
vyoma /workspace/challenges-Coditya0809/level1_design2 (uptickpro) $ make
make -f Makefile results.xml
make[1]: Entering directory '/workspace/challenges-Coditya0809/level1_design2'
/home/linuxbrew/.linuxbrew/bin/iverilog -o sim_build/sim.vvp -D COCOTB_SIM=1 -s seq_detect_1011          -f sim_build/cmds.f -g2012   /workspace/challenges-Coditya0809
/level1_design2/seq_detect_1011.v
MODULE=test_seq_detect_1011      TESTCASE= TOPLEVEL=seq_detect_1011          TOPLEVEL_LANG=verilog \
         /home/linuxbrew/.linuxbrew/bin/vvp -M /workspace/.pyenv_mirror/user/3.8.13/lib/python3.8/site-packages/cocotb/libs -m libcocotbvpi_icarus   sim_build/sim.vvp
     -.--ns INFO     cocotb.gpi                     ..mbed/gpi_embed.cpp:76   in set_program_name_in_venv        Did not detect Python virtual environment. Using sy
stem-wide Python interpreter
     -.--ns INFO     cocotb.gpi                     ../gpi/GpiCommon.cpp:99   in gpi_print_registered_impl       VPI registered
     0.00ns INFO     Running on Icarus Verilog version 11.0 (stable)
     0.00ns INFO     Running tests with cocotb v1.6.2 from /workspace/.pyenv_mirror/fakeroot/versions/3.8.13/lib/python3.8/site-packages/cocotb
     0.00ns INFO     Seeding Python random module with 1658580560
     0.00ns WARNING  Pytest not found, assertion rewriting will not occur
     0.00ns INFO     Found test test_seq_detect_1011.test_seq_bug1
     0.00ns INFO     Found test test_seq_detect_1011.test_seq_bug2
     0.00ns INFO     running test_seq_bug1 (1/2)
 15000.00ns INFO     #### CTB: Develop your test here! ######
 65001.00ns INFO     test_seq_bug1 passed
 65001.00ns INFO     running test_seq_bug2 (2/2)
 80001.00ns INFO     #### CTB: Develop your test here! ######
140002.00ns INFO     test_seq_bug2 passed
140002.00ns INFO     ********************************************************************************************
                     ** TEST                          STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
                     ********************************************************************************************
                     ** test_seq_detect_1011.test_seq_bug1   PASS       65001.00          0.00   32394719.40  **
                     ** test_seq_detect_1011.test_seq_bug2   PASS       75001.00          0.00   42671866.32  **
                     ********************************************************************************************
                     ** TESTS=2 PASS=2 FAIL=0 SKIP=0                 140002.00          0.02    9052956.29  **
                     ********************************************************************************************

make[1]: Leaving directory '/workspace/challenges-Coditya0809/level1_design2'
vyoma /workspace/challenges-Coditya0809/level1_design2 (uptickpro) $
```

The updated design is checked in as seq_detect_1011_fix.v

Verification Strategy

The stratergy I followed to detect the bugs was simple. First, I drew a state transition diagram from the problem specifications. Then, I confirmed the state transitions by comparing my state transition diagram to the state transitions in the design file.

Is the verification complete ?

I am not quite sure! Although the functionality of the design is as per the specifications, we can never be sure because of the huge amount of possibilities of the input bit sequences. However, I am very confident that any input sequence will be correctly detected if the `current_state` values don't get forced to change due to other hardware malfunctions.