Lab 2: Checker for the Fibonacci Sequence Generator

Nishad Saraf| ECE 593: Fundamentals of Pre-Silicon Functional Validation| Spring 2017

False failures due to bugs in the checker:

1. Rule 3 Failure due to wrapping of 'nextNum' in the function fib:

   Function fib is used to mathematical calculate the expected value of data_out to check the correctness of output data. While in loop the variable 'nextNum' store the most recent result of calculation in a particular iteration. When the values of 'currentNum' and 'prevNum' (nextNum = prevNum + currentNum) are close to the overflow value one more looping can result in a value which is greater than the value the variable 'nextNum' can hold. This causes the 'nextNum' to wrap around are as a result the expected value was computed wrongly. To resolve this issue is the loop can go into next loop only when addition operation won't result into overflow. If it overflows the result value equal to max value is returned by the function.

2. Rule 2 Failure due to the use of two-state equality operator:

   Two state equality operator (==) can only compare values with 0's and 1's. In rule 2 when you use such an operator the if condition will always evaluate to true as x can be zero or one hence even if the data_in and/or order are invalid. In such a scenario, the four-state operator must be used (===) which can handle x's as well as z's along with 0's and 1's.

Functional Verification:

1. Rule 1: When reset_n is asserted (driven to 0), all outputs become 0 within 1 clock cycle.

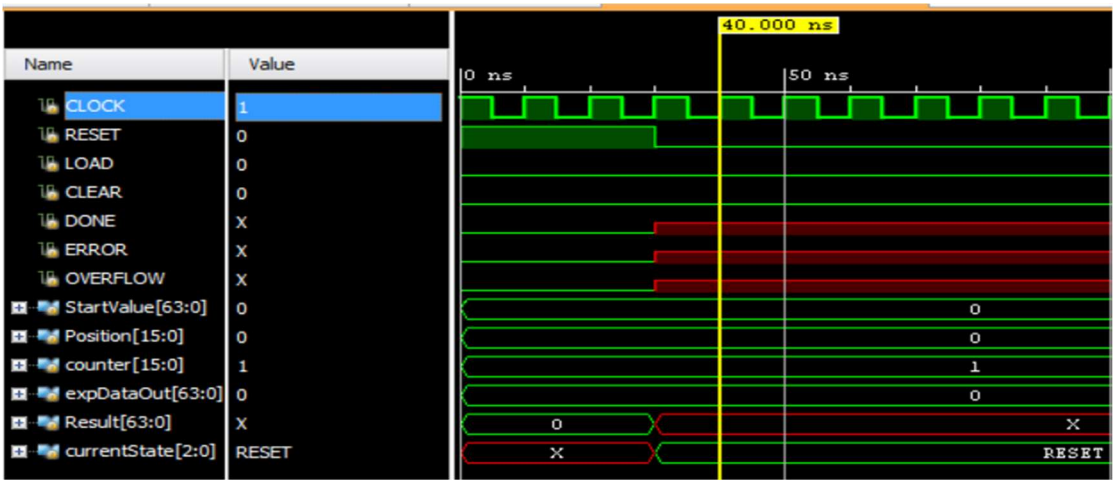
   - Original code (FSM):

```
unique case(currentState)
    RESET:
    begin
            data_out = {DATA_WIDTH{1'b0}};
            done     = 1'b0;
            overflow = {ORDER_WIDTH{1'b0}};
            error    = 1'b0;
```

   - Bug: Introduce in FSM while assigning value to output port on RESET.

```
unique case(currentState)
    RESET:
    begin
            data_out = {DATA_WIDTH{1'bx}};
            done     = 1'b0;
            overflow = {ORDER_WIDTH{1'b1}};
            error    = 1'b0;
```

   - Waveform:



- Transcript:

```
Rule 1 is ON
Rule 2 is ON
Rule 3 is ON
Rule 4 is ON
Rule 5 is ON
Rule 6 is ON
Error: Rule 1 violation. Possible Reason: Output port are
not initialized to 0’ on reset.
Time: 40 ns  Iteration: 0  Process:
/TestBench/check/Always127_9  File: D:/Coursework/ECE-
593/project_2/project_2.srcs/sources_1/new/Checker.sv
```

2. <u>Rule 2: When load is asserted, valid data in and valid order (no X or Z) must be driven on the same cycle.</u>

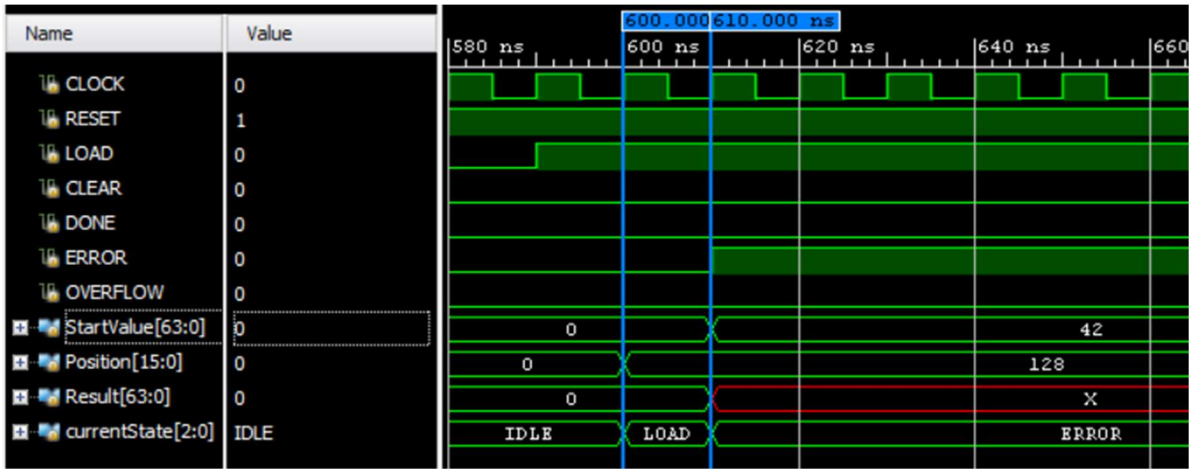
- Original code (Test module):

```
repeat (MAX_SEQ_COUNT) begin

    // Increment test sequence counter
    test_seq = test_seq+1;

    // Set load
    int_load = 1;
    repeat (1)  @(posedge clk);

    // Randomly generate order and data_in
    int_order = ({$random} % 8'hff);
    int_data = ({$random} % 8'hff);

    $display ("@ %0d ns: Test sequence: %0d:
```

- Bug1: Delayed the input data by three cycle but gave the order on time.

```
repeat (MAX_SEQ_COUNT) begin

   // Increment test sequence counter
   test_seq = test_seq+1;

   // Set load
   int_load = 1;
   repeat (1)  @(posedge clk);

   // Randomly generate order and data_in
   int_order = ({$random} % 8'hff);
   repeat (1)@(posedge clk);
   int_data = ({$random} % 8'hff);

   $display ("@ %0d ns: Test sequence: %0d:
```

- Waveform:



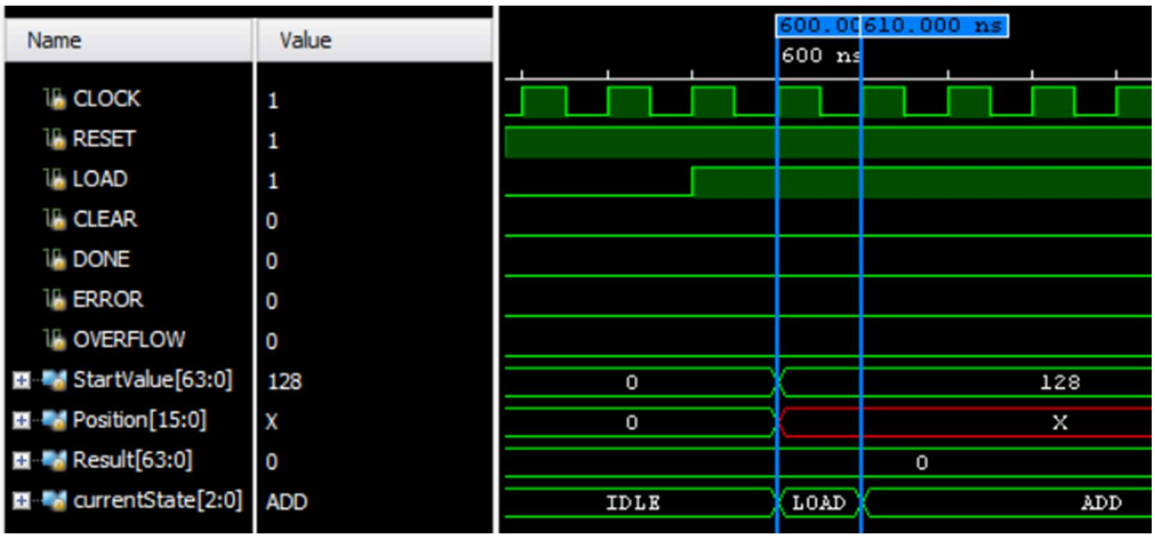
- Transcript:

```
@ 500 ns The chip is out of reset
@ 610 ns: Test sequence: 1: Initial data= 42. Fibonacci
order= 128
Error: Rule 2 violation. Possible Reason: Data and order
are not driven on the same cycle.
Time: 620 ns  Iteration: 0  Process:
/TestBench/check/Always143_10  File: D:/Coursework/ECE-
593/project_2/project_2.srcs/sources_1/new/Checker.sv
Rule 5 successful!
```

- Bug2: Gave invalid order as input through test module.

```
repeat (MAX_SEQ_COUNT) begin

   // Increment test sequence counter
   test_seq = test_seq+1;

   // Set load
   int_load = 1;
   repeat (1)  @(posedge clk);

   // Randomly generate order and data_in
   int_order = 64'dx;
   int_data = ({$random} % 8'hff);

   $display ("@ %0d ns: Test sequence: %0d:
```

- Waveform:



- Transcript:

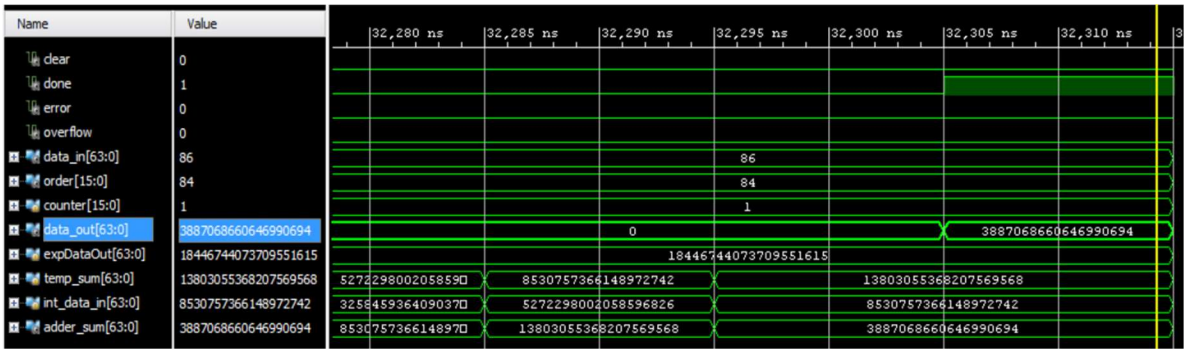
```
@ 500 ns The chip is out of reset
@ 600 ns: Test sequence: 1: Initial data= 128. Fibonacci
order= x
Error: Rule 2 violation. Possible Reason: Invalid Data and
order.
Time: 610 ns  Iteration: 0  Process:
/TestBench/check/Always143_10  File: D:/Coursework/ECE-
593/project_2/project_2.srcs/sources_1/new/Checker.sv
```

3. <u>Rule 3: Once "done' is asserted, output "data_out" must be correct on the same cycle.</u>

- The checker module could flag the first Rule 3 violation in the DUT uploaded by **Prof. Ahmad when Input data = 86 and Order = 84**.

- Bug: The bug in the design is due to overflow of registers not handled coreectly as a result the variables responsible of hold the result used to wrap around a give a different value instead of outputting the max value (All 64-bit F's).

- Waveform:



- Transcript:

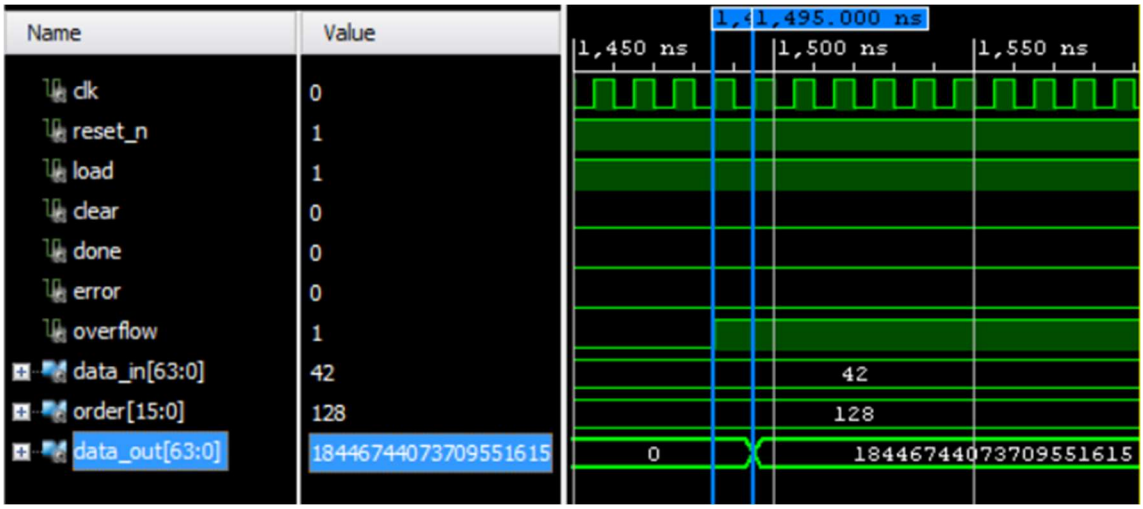
```
@ 31445 ns: Test sequence: 37: Initial data= 86. Fibonacci
order= 84
@ 32315 ns: Result: 3887068660646990694. Overflow bit: 0.
Error bit: 0

Error: Rule 3 violation. Possible Reason: Output data not
driven on the same cycle of done signal or the actual value
doesn't match the expected value.
Time: 32315 ns  Iteration: 0  Process:
/fib_num_tb/check/Always174_9  File: D:/Coursework/ECE-
593/project_3/project_3.srcs/sim_1/imports/new/Checker.sv
```

4. Rule 4: Once "overflow' is asserted, output "data_out" must be all 1's on the same cycle.

- Again, the checker module could flag the Rule violation in the DUT uploaded by **Prof. Ahmad for the first time when Input data = 42 and Order = 128**.

- Bug: The bug was due the fact that the result value data_out was not driven on the same cycle as overflow flag.

- Waveform:



- Transcript:

```
@ 500 ns The chip is out of reset
@ 605 ns: Test sequence: 1: Initial data= 42. Fibonacci
order= 128
Error: Rule 4 violation. Possible Reason: Output data is
not available on the same clock cycle as overflow
Time: 1495 ns  Iteration: 0  Process:
/fib_num_tb/check/Always189_10  File: D:/Coursework/ECE-
593/project_3/project_3.srcs/sim_1/imports/new/Checker.sv
```

5. <u>Rule 5: Once "error' is asserted, output "data_out" must be all x's on the same cycle.</u>

- Original code (FSM):

```
ERROR:
begin
        done     = 1'b0;
        overflow = 1'b0;
        readFlag = 1'b0;
        data_out = {DATA_WIDTH{1'bx}};
        error    = 1'b1;       // set error pin high
        num      = {DATA_WIDTH{1'b0}};
        position = {ORDER_WIDTH{1'b0}};
end
```

- Bug: Instead of x's wrote 0's in ERROR state on the output port.

```
ERROR:
begin
        done     = 1'b0;
        overflow = 1'b0;
        readFlag = 1'b0;
        data_out = {DATA_WIDTH{1'b0}};
        error    = 1'b1;       // set error pin high
        num      = {DATA_WIDTH{1'b0}};
        position = {ORDER_WIDTH{1'b0}};
end
```

- Transcript:

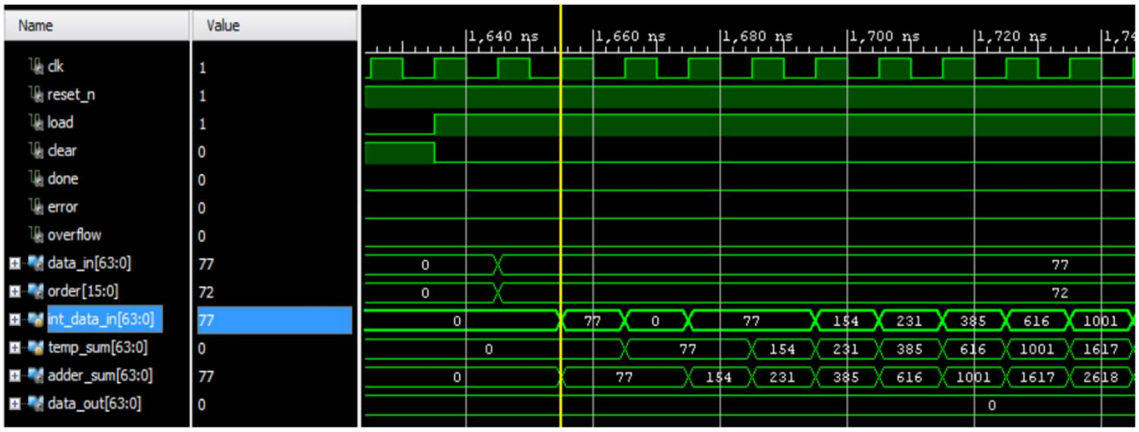
```
@ 5150 ns: Test sequence: 6: Initial data= 120. Fibonacci
order= 0
Rule 2 successful!
Error: Rule 5 violation. Possible Reason: Output data is
dosn't contain all x's or the output is not available on
the same cycle as error signal.
Time: 5170 ns  Iteration: 0  Process:
/TestBench/check/Always204_13  File: D:/Coursework/ECE-
593/project_2/project_2.srcs/sources_1/new/Checker.sv
```

6. Rule 6: Unless it's an error or overflow condition, done and correct data must show up on the output "order+2" cycles after load is asserted.

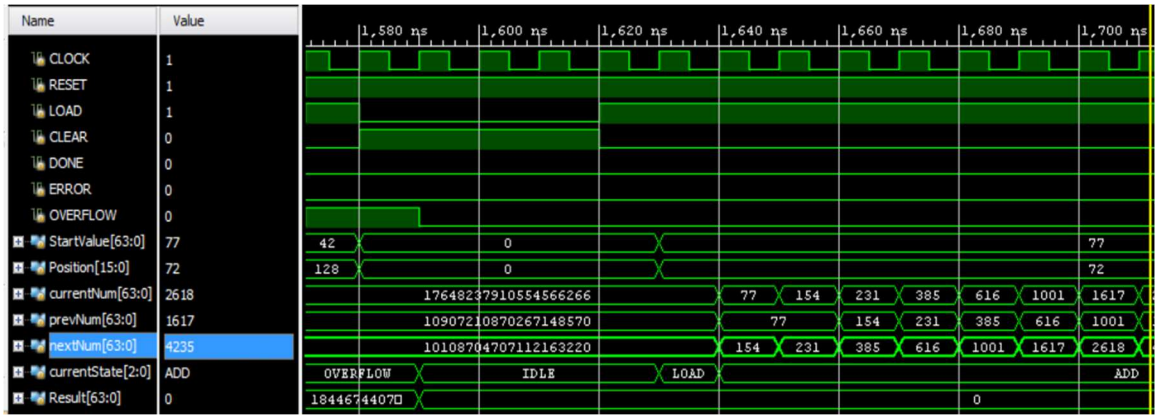
    - After load signal is asserted exactly one cycle later valid data and order are driven on the input port by the test module. And once when the computation is finished it takes one extra clock cycle to change state to DONE state from ADD. Therefore, after load signal is asserted it should take (order + 2) number of cycle to get the output result.

    - Bug: But in the DUT provided by Prof. Ahmad there is one redundant computation cycle at the start of the computation cycle. This causes a total of (order + 3) number of cycles to get the result. Referring the waveforms given below will help to understand the scenario more clearly.

- Waveform (Buggy):



- Waveform (Correctly optimized to order+2 computation cycles):



- Transcript:

```
@ 1645 ns: Test sequence: 2: Initial data= 77. Fibonacci
order= 72
@ 2395 ns: Result: 62101696044803261. Overflow bit: 0.
Error bit: 0

Error: Rule 6 violation. Possible Reason: Computation took
longer than expected.
Time: 2395 ns  Iteration: 0  Process:
/fib_num_tb/check/Always219_12  File: D:/Coursework/ECE-
593/project_3/project_3.srcs/sim_1/imports/new/Checker.sv
```