Ryan Laur, Benjamin Wheeler

Lab 2 Report

Reconfigurable Computing 2

# Explanation of Testbench

## Random Test for 1000 tests

## Consecutive Test for 200 Tests

## Repeats 4 Times

## Watchdog to Check for Done and Break simulation if TIMEOUT

## Scoreboard checks internal signals against the signals used in the result model

## Checks if signals are cleared/reset

## Checks if signal is the same upon completion

# Bugs with done signal

## Done cleared too early

Preconditions:

* DUT in done state (done == 1)
* Testbench restarted DUT via go == 1

Violation:

* Done not cleared on following cycle (violates design specification)

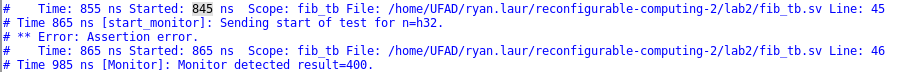
Detected in fib\_tb.sv line 100:

*// if go and done are both asserted, done should be cleared on the next cycle*

  assert property (@(posedge bfm.clk) disable iff (bfm.rst) bfm.go && bfm.done |=> !bfm.done)

  else $error("Time %0t [Assert Property]: Done=1, go=1, done not cleared next cycle.", $time);

Modelsim errors:



Timeline

Description automatically generated

## Done cleared too late

Preconditions:

* DUT in done state (done == 1)
* Testbench restarted DUT via go == 1

Violation:

* Done took two cycles to clear instead of one.

Detected in fib\_tb.sv line 100 (see assertion property for first error).



A picture containing timeline

Description automatically generated

# Infinite loop bugs

## Max number as an input causes state machine to never finish.

Preconditions:

* Circuit fed input of max number (e.g., WIDTH’(1’b1))
  + This input is the largest number the circuit can handle for its specified width.

Violation:

* Circuit never asserts done, gets caught in infinite loop

Steps to detect the bug:

* Created a watchdog timer that reported an error and stopped the simulation if done was not asserted within 100k cycles. (monitor.svh line 131)

Reported errors from the simulation:

# WATCHDOG : started at 643185

# \*\* Error: Time 643195 ns [Assert Property]: Done=1, go=1, done not cleared next cycle.

# Time: 643195 ns Started: 643185 ns Scope: fib\_tb File: /home/UFAD/ryan.laur/reconfigurable-computing-2/lab2/fib\_tb.sv Line: 101

# \*\* Error: Time 643205 ns [Assert Property]: Done=1, go=0, done not stable.

# Time: 643205 ns Started: 643195 ns Scope: fib\_tb File: /home/UFAD/ryan.laur/reconfigurable-computing-2/lab2/fib\_tb.sv Line: 109

# \*\* Error: Time 643205 ns [Assert Property]: done not cleared after go asserted.

# Time: 643205 ns Started: 643205 ns Scope: fib\_tb File: /home/UFAD/ryan.laur/reconfigurable-computing-2/lab2/fib\_tb.sv Line: 113

# done is not asserted time:1643180

# WARNING::WATCHDOG BITED

# \*\* Note: $stop : /home/UFAD/ryan.laur/reconfigurable-computing-2/lab2/monitor.svh(146)

# Time: 1643180 ns Iteration: 1 Region: /fib\_tb\_sv\_unit::start\_monitor #(6, 32)::watchdog

# Break in NamedForkStat fib\_tb\_sv\_unit/start\_monitor::watchdog/watch\_dog at /home/UFAD/ryan.laur/reconfigurable-computing-2/lab2/monitor.svh line 146

# Bug: Result register changes when circuit done

Preconditions:

* Circuit reaches done state
* Circuit asserts done signal

Violation:

* Result register changes as done is asserted, which should not happen.

Detected in fib\_tb.sv line 120, 123:

*// upon completion, (ie done = 1), result and overflow retain their values until circuit is restarted*

  assert property (@(posedge bfm.clk) disable iff (bfm.rst) bfm.done && $stable(bfm.done) |-> $stable(bfm.result))

  else $error("Time %0t [Assert Property]: Done=1, result not stable.", $time);

*// upon completion, (ie done = 1), result and overflow retain their values until circuit is restarted*

  assert property (@(posedge bfm.clk) disable iff (bfm.rst) bfm.done && $stable(bfm.done) |-> $stable(bfm.overflow))

  else $error("Time %0t [Assert Property]: Done=1, overflow not stable.", $time);

Graphical user interface, application

Description automatically generated

# Bug: Circuit reacts to changes on the input

Preconditions:

* Circuit is started with go
* Input n is changed before the circuit finishes (before done == 1)

Violation:

* Circuit does not produce correct output

Discovered via scoreboard logic that ensures l, x, and y registers have correct values compared to the model function. Since they were different, this means the DUT is stopping before it should and thus producing incorrect output.

Graphical user interface

Description automatically generated

# Overflow bugs

Preconditions:

* DUT was passed an input that would cause it to overflow

Violation:

* Overflow flag not asserted

Detected by utilizing an overflow model function, which truncates the actual value of the correct result to OUTPUT\_WIDTH, and compares with the original longint result. If they’re different, overflow occurred and the testbench will assert that the DUT asserts its overflow flag.

Accompanying testbench error:

# Time 445 ns [Scoreboard] Result test failed: full\_add\_r = h00000008 instead of h1e8d0a40 for n = h31.

# Time 445 ns [Scoreboard] Overflow test failed: overflow = 0 instead of 1 for n = h31 and theoretical result = h000000011e8d0a40 and result = h00000008.

# Bugs fixed in fib\_good

This list includes some of the major bugs we fixed in fib\_good, but is not exhaustive.

* Save circuit inputs in registers
* Reset temporary variables to the correct value on circuit reset
* Fix issues where non-blocking assignments to temp values were being read on the same clock edge
  + replaced with blocking assignments or rearranged states to ensure values were updated and read properly
* Defined state enum with logic [2:0] specifier to ensure synthesis tools recognize it as a state machine and operate/optimize accordingly
* Moved done assertions to proper states to follow guidelines on done signal

## Quartus synthesis screenshot of fib\_good

Graphical user interface, text, application

Description automatically generated