UART\_Clock\_Generator

Diagram

Description automatically generated

The following is a breakdown of my implementation of the different modules for the UART\_Clock\_generator, as well as corresponding verification tests.

**Module 1:** Divide\_by\_13

For verification, I simply recreated the waveforms from the book.

The source file is: divideby13.v, and the testbench is divideby13\_tb.v

Diagram

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated

**Module 2:** Divide\_by\_8

For verification, I simply recreated the waveforms similar to Module 1. The output “clock\_by\_8” is high when the internal counter variable asserts a value of 7 (in line with clk\_1 from the previous code).

The source file is: divideby8.v, and the testbench is divideby8\_tb.v

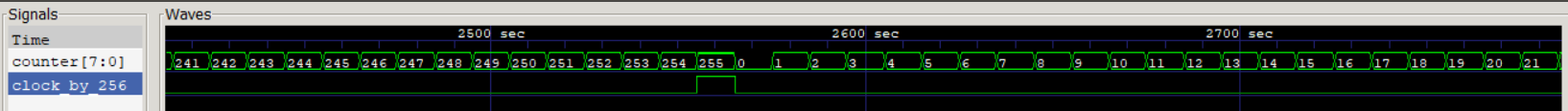
**Graphical user interface, application

Description automatically generated**

**Module 3:** Divide\_by\_256

For verification, I simply recreated the waveforms similar to Module 1. The output “clock\_by\_256” is high when the internal counter variable asserts a value of 255 (again, in line with clk\_1 from the previous code), as shown in the attached waveform.

The source file is: divideby256.v, and the testbench is divideby256\_tb.v

****

The 8-bit counter is exposed as an output since it has to be used as an input to the multiplexer for the UART\_Clock\_Generator module.

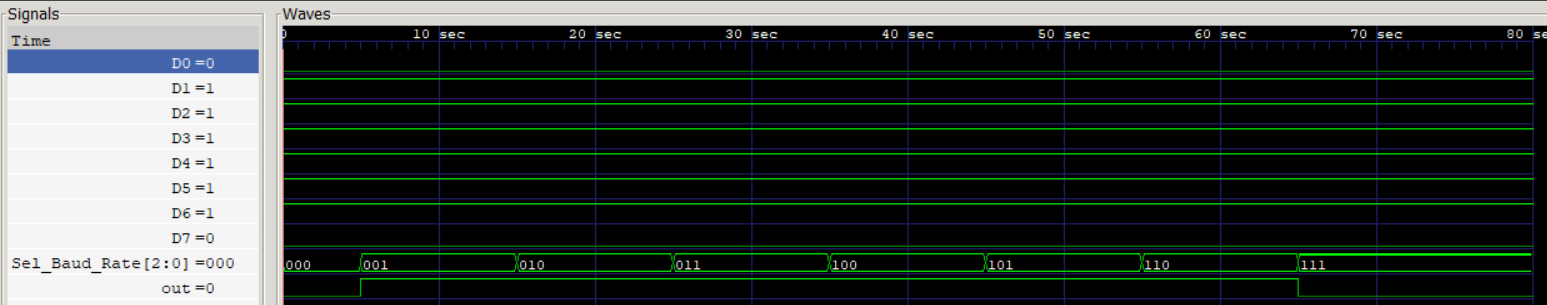
**Module 4:** 8 to 1 multiplexer

The source file is: 8to1mux.v, and the testbench is 8to1mux \_tb.v

For testing, I initialize each of the 8 inputs randomly, cycle through the select pins, and assert the correct output. I also implemented error monitoring, watchdog signals, as well as self-reporting in my testbench. The testbench executes with 0 errors.

Text

Description automatically generated



The waveform shows that the inputs are indeed correctly multiplexed on the out pin, based on the 3-bit Sel\_Baud\_Rate input pin.

Overall Module: UART\_Clock\_Generator