Assignment 2

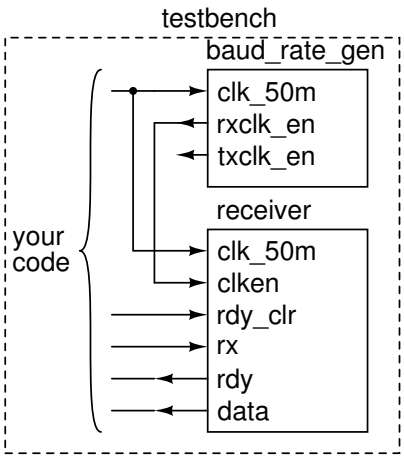
Due Tuesday, March 27. Submit your assignment using the appropriate dropbox on the course web site. Assignments submitted after the solutions are made available will be given a mark of zero.

Question 1

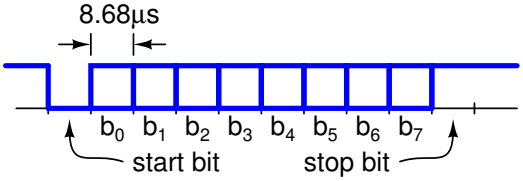
In this assignment you will write a testbench to test the receiver portion of a UART (Universal Asynchronous Receiver-Transmitter). The design you are to test can be downloaded from https://github.com/jamieiles/uart.

You will only test the receiver module (`receiver.v`) and the associated baud rate generator module (`baud_rate_gen.v`).

The modules will be connected to your testbench as follows:



Your testbench must generate the following waveform on the `rx` signal:

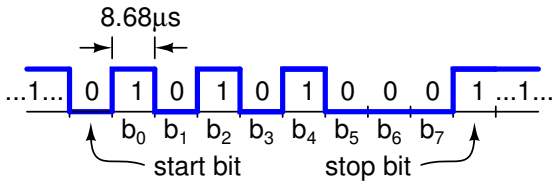


The `rx` signal should be set to 1 when no data is being transmitted. A byte is transmitted using a waveform that consist of 10 bit periods, each of duration of $\frac{1}{115200} \approx 8.7\ \mu s$. During each bit period the waveform can be 1 or 0.

The first bit duration, the "start bit," must be 0; the last bit duration, the "stop bit," must be 1.

The other bits, $b_i, i = 0 \ldots 7$, represent the bits of an 8-bit binary number. These bits are sent in order from least– to most–significant bit.

You must verify the UART using a unique test value computed by adding up the digits of your BCIT ID. For example, if your BCIT ID is A00123456 you would send the value $0+0+1+2+3+4+5+6 = 21_{10} =$ `2'b0001_0101`. This would result in a waveform with the following logic levels: 0 (start bit), 1 (l.s. bit), 0, 1, 0, 1, 0, 0, 0 (m.s. bit), 1 (stop bit):



Your testbench should do the following:

1. output a 50 MHz clock on `clk_50m` continuously
2. assert `rdy_clr` for at least two periods of `clk_50m`
3. set `rx` to the idle value (1) for at least one bit duration
4. output the test waveform described above on `rx`
5. wait until `rdy` is asserted (set to 1)
6. check that `data` has the expected value
7. wait one more bit period (to make it easier to see the final `data` value in the simulation waveform)

Your testbench should use `$display()` to output an appropriate message showing "passed" or "failed" and the value of `data`.

Submit your solution to the appropriate dropbox (in PDF format, as usual) including:

1. The computation of the decimal and binary values of your test value.

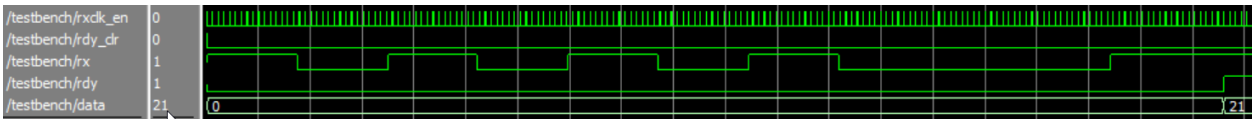


Figure 1: Sample simulation waveforms.

2. The sequence of logic levels (0 and 1 to be transmitted, including idle, start, data and stop bits).

3. The source code for your System Verilog testbench (do not include `receiver.v` or `baud_rate_gen.v`).

4. A screen capture of your console output showing the "passed" or "failed" result output. For example:

```
VSIM 4> run -all
# passed: data= 21
# ** Note: $stop    : C:/
#    Time: 96330 ns  Iter
```

5. A screen capture of the simulation waveforms including at least the following signals: `rxclk_en`, `rdy_clr`, `rx`, `rdy`, and `data`. An example is shown in Figure 1.

Hints

- Inverting a signal every 10 ns produces a clock with a period of 20 ns.
- `repeat(8) @(posedge clk50_m);` waits for 8 clock cycles.
- `#8.7us;` delays 8.7 μs.
- The expression `(n & (1<<i))` tests the i'th bit of the integer `n`
- `$display("the value of x is %d",x);` prints the value of x in decimal on the console.
- `$stop;` stops your simulation (without terminating the simulation software).