**PRELIMINARY DESIGN REVIEW**

**Lafayette College**

**Department of Electrical and Computer Engineering**

Title: Asynchronous Serial Receiver

Authors: Zainab Hussein and Kemal Dilsiz

Date: September 26, 2017

1. Requirement checklist

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Test Method** | **Detailed Results** | |
| 1. Module Interface | Code Inspection | We will add few extra inputs and outputs compared to the Lab design, i.e.   * A WAIT state to determine the baudrate using the **rdy** output. * Check for error in the start bit by adding a 3rd check in the START state. | |
| 2. Module function: accepts rxd input and receives data bits one by one, ferr output for when there is a frame error and indicates readiness for the next receiving with rdy output | Demonstration in oscilloscope and test bench simulation and Nexys4DDR board:   * Proper display of rdy output with a LED of the board * Proper display of 01010101, 00110011, 00001111, 00000000, 11111111 data inputs with seven segment display Nexys board * Proper display of BaudRate with oscilloscope using the rdy state * Proper display of rdy output on oscilloscope * Proper display of ferr output on testbench * Display of data received in test bench * Display of STOP-START transition in the test bench * Proper display of reset on seven segment display on Nexys board | | We will provide diagrams |
| 3. Uses Nexys4 board 100Mhz clock; all flip-flop clock inputs tied directly to this signal | Code inspection  *(all the instances of the clk use in the modules are provided)* | We will provide the code | |
| 4. Contains no latches | Inspection of Synthesis Report | TBD | |
| 5. Test circuit – show test that test circuit functions properly to exercises circuit. | Demonstration in hardware |  | |
| In submitting this checklist as part of our report, I/We certify that the tests described above were conducted and that the results of these tests are accurately described and represented. I/We understand that any misrepresentation of the tests or the results constitutes a violation of the College policy on academic dishonesty. | | | |
| *Name(s):Kemal Dilsiz & Zainab Hussein Date: 09/26/2016* | | | |

The test cases of the data received: 01010101, 00110011, 00001111, 00000000, 11111111 can be observed:

* Seven segment display on the FPGA board
* Oscilloscope through observing a single bit sum value, 1 or 0, of the data byte received.

To display the proper baudrate, we can use the **rdy** output for a WAIT state we introduce to receive one data byte. We have 10 states for when we receive a data byte, which is the duration of which **rdy** should be 0. Displaying the **rdy** signal on the oscilloscope for this wait state should allow for determination of baudrate by dividing the value by 10 (this is the number of states, but baudrate is for 1 state).

Testing the reset on the seven segment display will be through observing the data going to a default value set for reset.

1. High-level Description

We have thought about using 8 different states for the RECEIVE state showing each individual data bit received. Instead we decided to use a counter for the RECEIVE state to make our code cleaner and brief.

We decided to use two counters: time counter and state counter with two different enables for each to keep track of change in state from receive to stop state, and also the baudrate division of 16 per state. This made it easier to decide when to sample the data and when to check for spurious START. We decided on two counters than one counter for clarity when keeping track of either the baudrate or for RECEIVE-STOP state transition.

> we decided to implement a wait state for single data receival for being able to test rdy output on the oscilloscope

> we could have implemented an extra feature to generate errors for checking the ferr in the oscilloscope but we decided that this would be unnecessary since we already check it in the test bench and it would also be extra code

> we have implemented multiple checks during the START state to ensure that the error in the START bit…

1. Detailed Description

>Self checking test benches will be used to blab la

We have 4 states, START,STOP,RECEIVE and IDLE and this state does this

We have two counters and we change the state every time the time counter reaches 16, whereas we change the data receival bit with each state counter value

Our WAIT state works only when BTNC is pressed and stops working when the button is let go. Therefore, while the button is pressed we can only WAIT for the button to be released before the module starts receiving data again.

