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

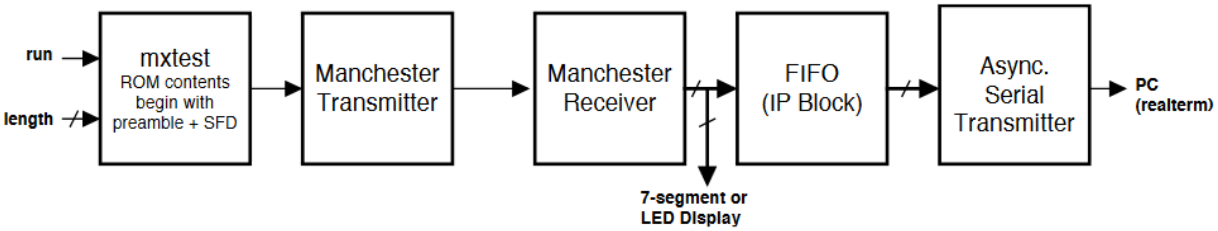
Description	Test Method	Detailed Results
1. Module Interface	Code Inspection	We will add few extra inputs and outputs compared to the Lab design, i.e. 1. Check for error in PREAMBLE-SFD via correlator use 2. Added Transmitter from Lab3 for switching in between RealTerm data input and user given switches data input
2. Module function: accepts rxd input and receives data bits one by one, err output for when there is a frame error and indicates readiness for the next receiving with cardet output	Demonstration in oscilloscope and test bench simulation and Nexys4DDR board: **TEST BENCH** 1. Display of error situations 2. Display of 10101010, 11001100 data transmissions 3. Proper display of time_count 4. Display of data received 5. Display of PREAMBLE-SFD transition 6. Proper display of reset 7. Proper display of the BaudRate 8. Proper display of correlator output	**OSCILLOSCOPE & FPGA HARDWARE** 1. Display of cardet output with a LED of the board 2. Proper display of 01010101, 00110011, 00001111, 00000000, 11111111 data inputs with seven segment display Nexys board 3. Proper display of BaudRate with oscilloscope using the cardet state 4. Proper display of cardet output on oscilloscope 5. Proper display of reset on seven segment display on Nexys board 6. Proper display of error on the oscilloscope (sticky)
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)*	Provide the clock report from vivado
4. Contains no latches	Inspection of Synthesis Report	Provide the section in vivado synthesis report
5. Test circuit – show test that test circuit functions properly to exercises circuit.	1. Demonstrate 10101010, 11001100, 11110000, 00000000, 11111111 on the seven segments and the oscilloscope 2. Demonstrate two consecutive data transmission of NM letters through RealTerm 3. Demostrate BaudRate from cardet signal 4. Demostrate sticky error signal 5. Demostrate every bit of the data after receival 6. Demostrate the transition from user input transmitter to RealTerm input 7. Demonstrate multiple data transmission (set length)	Demonstration to Professor Nadovich

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: 10/17/2016*

2. High-level Description

Our design simply combines the manchester transmitter from Lab3 and manchester receiver implementation from this lab. Therefore, the data can be transmitted through the user input and also from the mx_test module via changing the length. This would allow us to quickly change input and test it on the oscilloscope.



The data received will be stored in FIFO for displaying in RealTerm via Async Serial Transmitter from lab2. This will make it so that we can check if our input is same as the output even when there is noise. Or if it picks up random data because of a wrong error check when PREAMBLE and SFD were not present.

Correlator will be used for checking individual bits and also checking PREAMBLE and SFD. This will enable us to check with different HTRESH and LTRESH values for accuracy with different noise input. Therefore we can optimize it.

ALTERNATIVES

1. Using an FSM for SFD state where we could simply cycle through an initial state for the extra 10's left from the preamble. This was a very valid approach but we decided not to use it for few reasons.
   - If there was a noise error in SFD, that could throw the FSM off and then we may lose data transmission
   - This could be fixed to a very high accuracy by checking 16 bits in each rxd and optimizing HTRESH for 16 bit input but still it could have problems
   - Extra implementation of FSM can be time consuming and decreases readability

2. There are few little things that could be changed and would impact the design drastically
   - Changing the sampling rate for each bit, (we use 8, can use 16, or 12 or 1) (correlator)
   - Dividing SYNCRONIZE into 2 states, PREAMBLE and SFD, which is actually an alternative approach that we think about

3. Detailed Description

In our design there are few key points:

- The HTRESH for 16 bit input is 13, and 8 bit input is 7. LTRESH is 3 for 16 bit input, 1 for 8 bit input.
- We will have 8 bit checks of each bit with the correlator (Figure 1)
- We will check extra preamble bits before SFD with a correlator that will shift register 14 bits until SFD is found → goes into RECEIVE state (Figure 2)
- Please see Figures 3-4 for top module and the main FSM

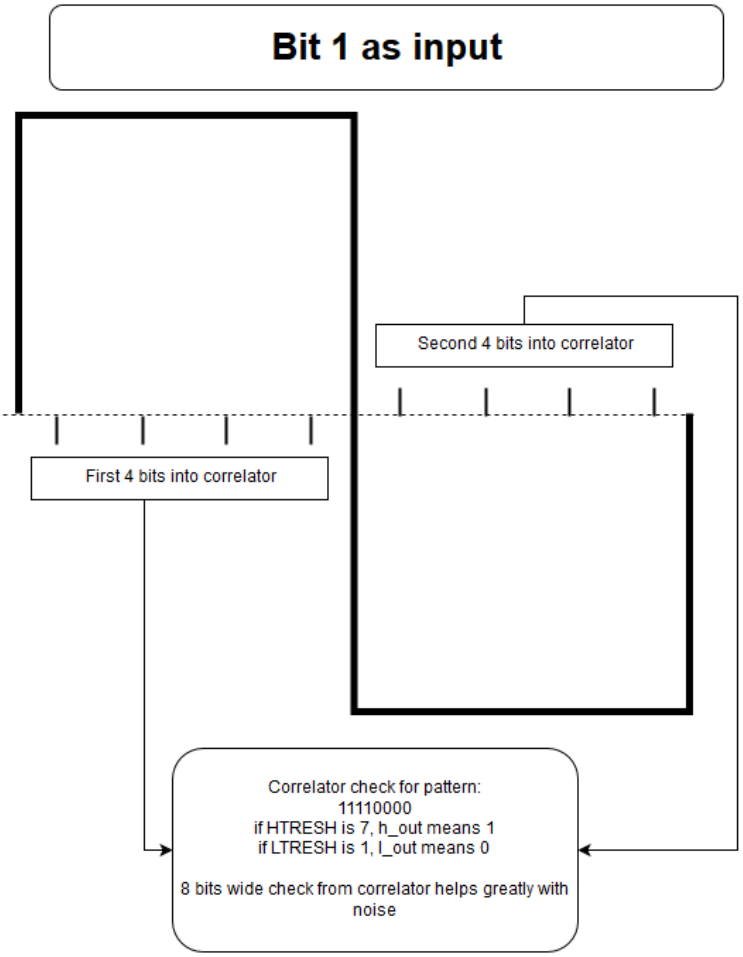


Figure 1

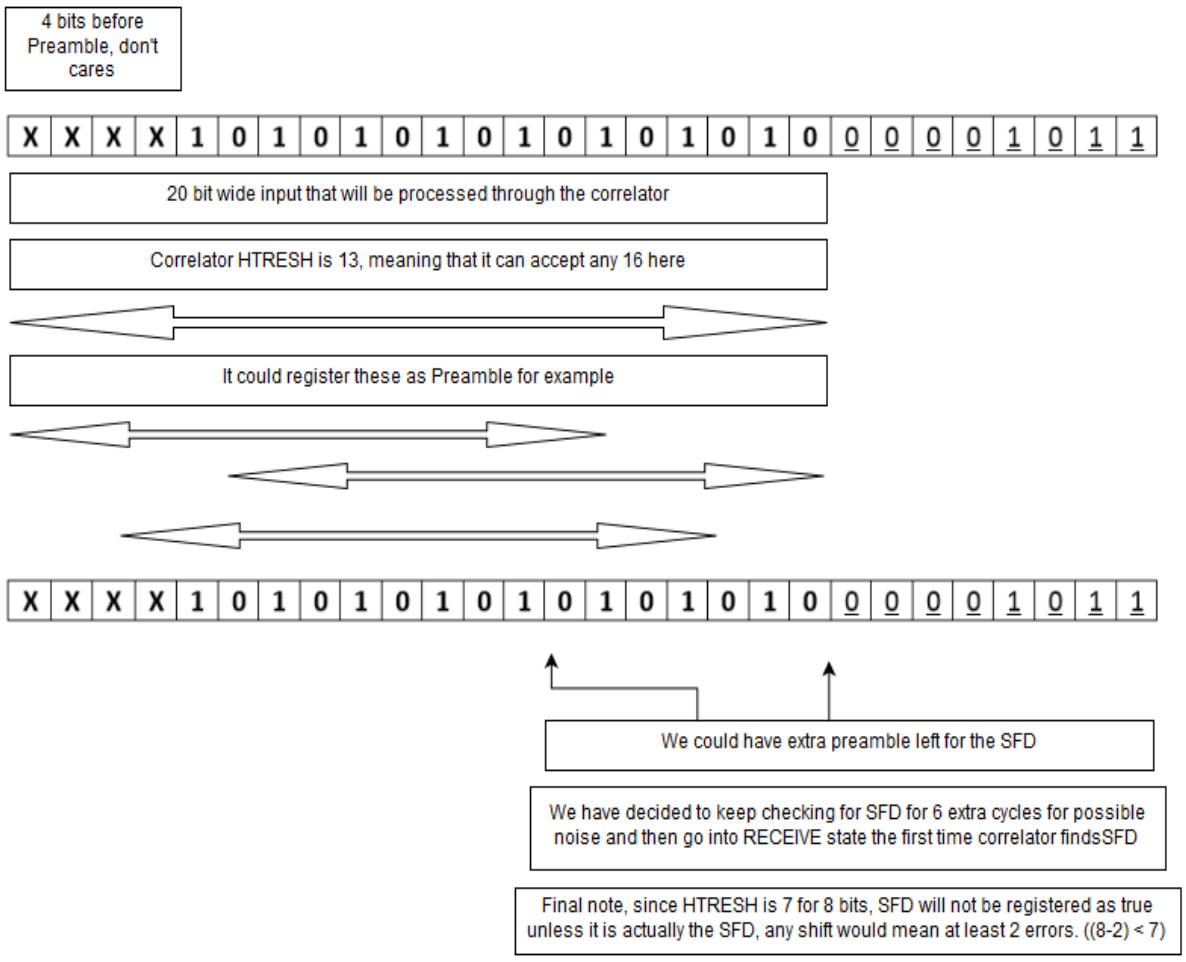

4 bits before Preamble, don't cares
X X X X 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 1 1
20 bit wide input that will be processed through the correlator
Correlator HTRESH is 13, meaning that it can accept any 16 here
It could register these as Preamble for example
X X X X 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 1 1
We could have extra preamble left for the SFD
We have decided to keep checking for SFD for 6 extra cycles for possible noise and then go into RECEIVE state the first time correlator findsSFD
Final note, since HTRESH is 7 for 8 bits, SFD will not be registered as true unless it is actually the SFD, any shift would mean at least 2 errors. ((8-2) < 7)

Figure 2

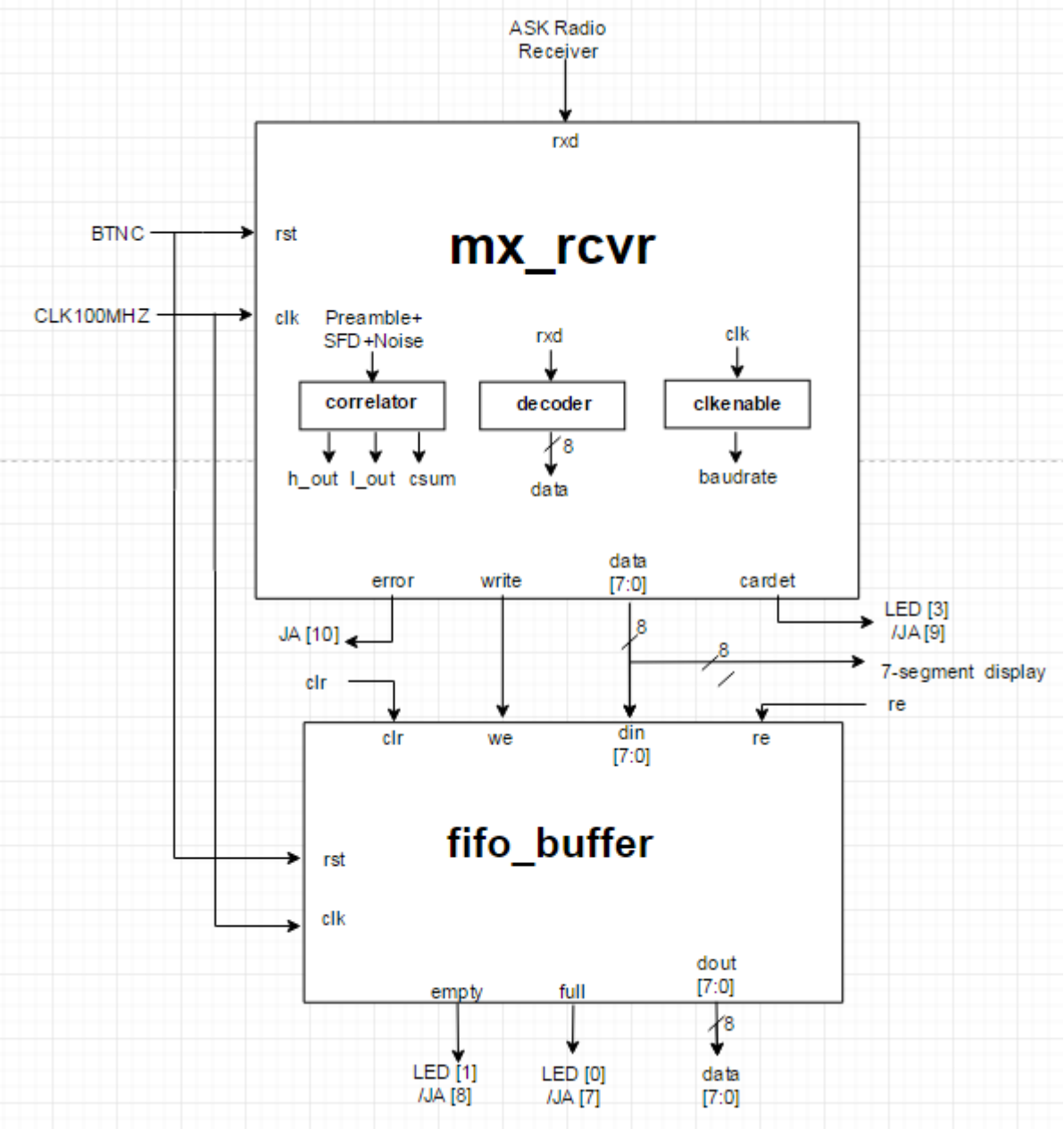


Figure 3

h_out → can mean if the correlator matched the pattern or can mean if the data is 1

l_out → can be used for recognizing 0 (put same pattern for 1 and 0 and l_out would be asserted high when the input is 0 instead of 1 because it is exactly the opposite)

csum → to check the accuracy of the output, can be used for some exceptions

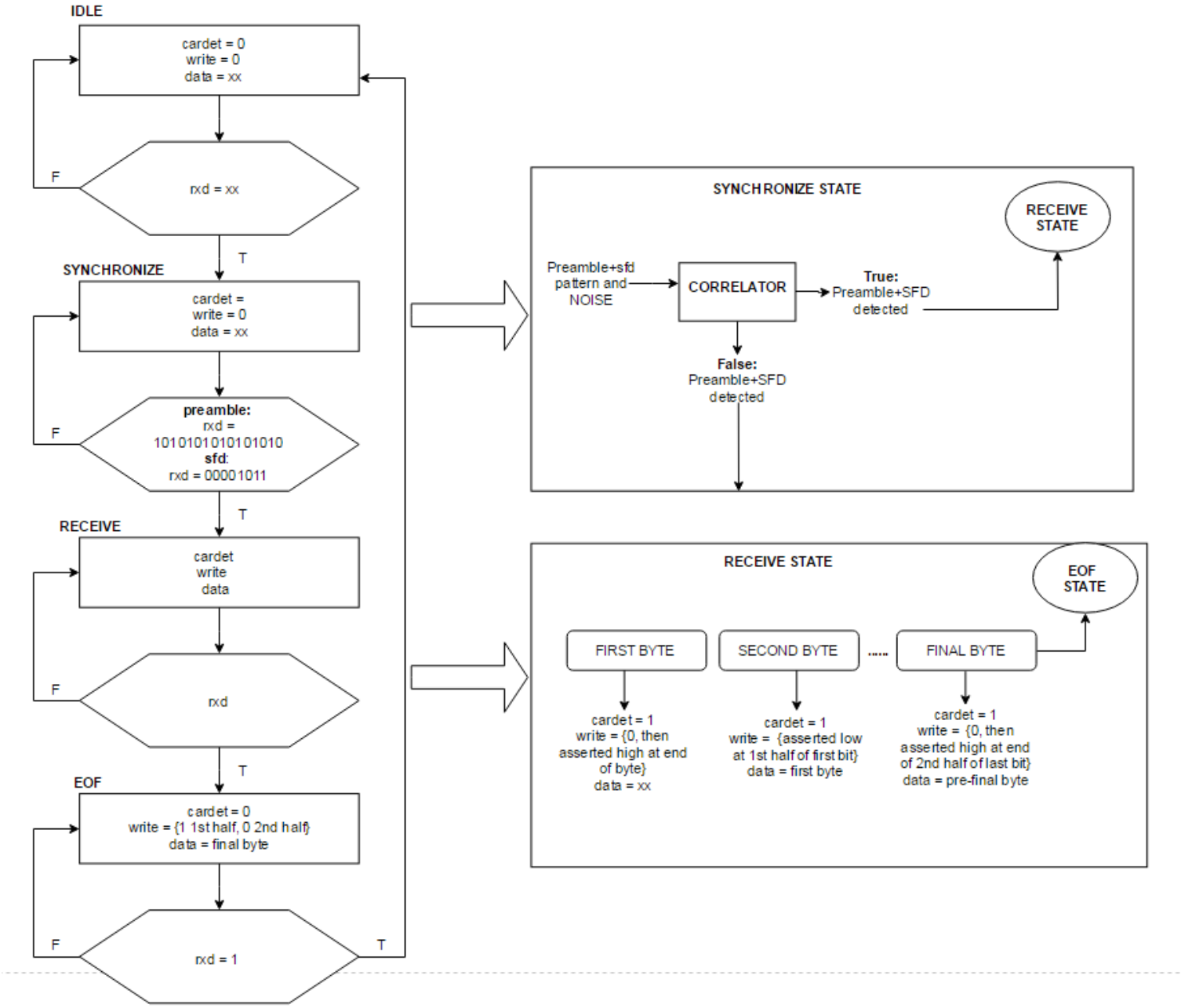


Figure 4

Explanation:

IDLE: When the correlator does not realize the Preamble, we stay in IDLE state and random values may be received due to the noise. We have to make sure that we can recognize the patterns, so while we are in IDLE, correlator checks for preamble.

SYNCHRONIZE: This state basically first checks for preamble and changes the correlator if a variable named preamble is asserted high

RECEIVE: Receives data and concatenates it after passing it through a decoder to assign the bits into individual variables

EOF: After the data transmission is received, the FSM waits for 2 clock cycles to show that