Lab Assignment 8

Title: Asynchronous Serial Receiver

Learning Objective:

Learn how to receive serial information without a shared clock.

Specification:

Design asynchronous serial receiver with baud rate = 9600, 8 data bits, no parity bits and 1 stop bit. Connect this to the micro USB port of the BASYS 3 board. Use gtkterm on PC to test and demonstrate. Instructions for gtkterm usage are posted separately.

Details:

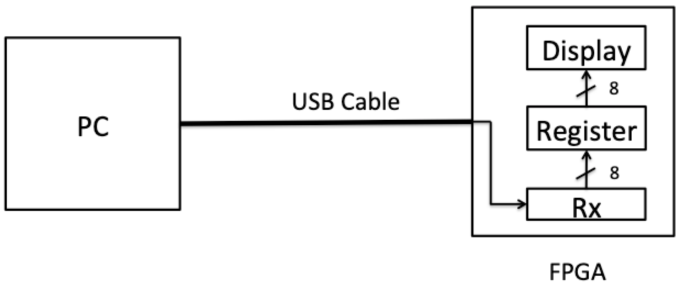
Design and implement circuit for receiving asynchronous data serially as per the above specifications. For details of the serial asynchronous protocol, refer to lecture 20 (dated 11th September). The receiver operates with a clock that is 16 x baud rate for proper detection of the start bit. The sequence of states is shown in the figure below.

For the design of the receiver, refer to slides 27 and 28 of Lecture 21 (dated 13th September). It primarily consists of a serial-in parallel-out register (rx_reg), two counters (count and i) and an FSM with 3 states (*idle*, *start* and *si*). Each of these can be described as a clocked process. Transition takes place from *idle* state to *start* state when the first '0' is seen on rx_in. On seeing the 8th '0', the state changes to *si*. After that, the input is sampled at intervals of 16 rxclk cycles, to get the data bits and the stop bit. After sampling the stop bit, the state changes to idle.

A modulo 16 counter (count) triggered by rxclk is used which is initialized to 0 when a transition is made from *idle* state to *start* state. While counter has values ranging from 0 to 6, rx_in is sampled to ensure that the start bit is wide enough. Subsequently, state changes take place coinciding with transitions of the counter from 6 to 7. While the receiver is in *start* state, observing a '1' on rx_in should lead to rejection of the start bit and transition back to *idle* state.

Make a provision for resetting the FSM to *idle* state by a push button.

Connect the 8 bit parallel output of the receiver either to eight LEDs or two 7-seg displays through a parallel-in parallel-out register as shown. The purpose of this register is to hold the data last received while the bits of the next data may get shifted into rx_reg. Once the



stop bit is received in rx_reg, in the next cycle the data bits can be transferred to this register.

The subsequent assignment will involve design and implementation of a serial asynchronous transmitter and connecting the parallel output of the receiver to the parallel input of the transmitter to form a loop. This will be used to demonstrate that the data sent from the PC is echoed back to it through the receiver/transmitter pair.