ENSC 350 Project - UART

Project Description and Overview

Members:
Samin Moradkhan
Rose Epstein

Instructor:
Behnam Ghavami

Table of Contents

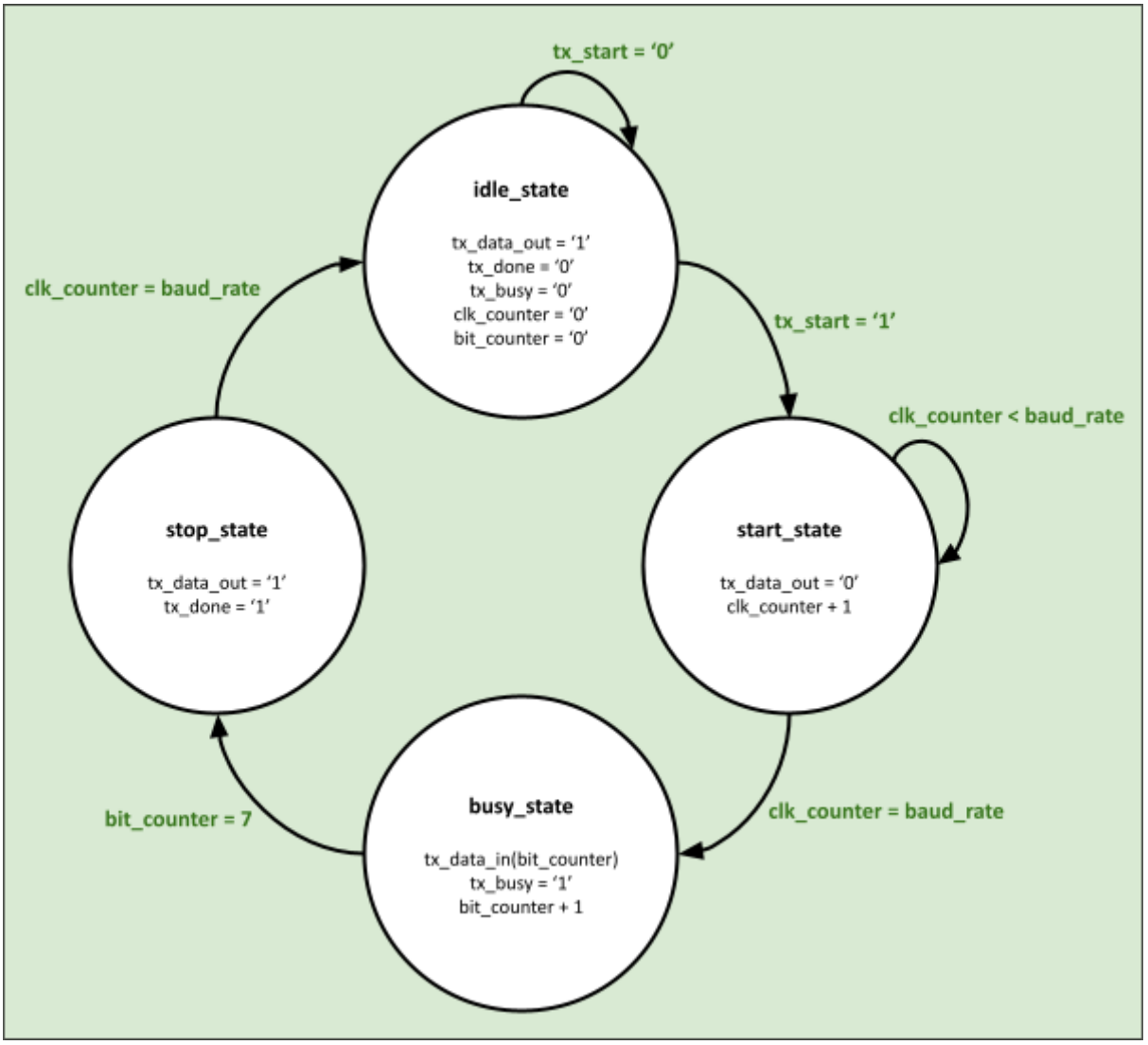
1. Transmitter

Description

This module transmits data to the device (computer) serially, or bit by bit. The 8-bit data is transmitted LSB to MSB to the receiver. Transmission starts when the start bit is low, and finishes when the stop bit is high. The process is triggered by a key pressed on the FPGA board.

Modules

1. FSM: The FSM diagram for the transmitter is below.



2. Bit Counter: The bit counter keeps track of the bits that are transmitted and is incremented after each shift.
3. Shift Register: The shift register loads the 8-bit data to a 11-bit buffer (including the start, stop and parity bit). This is used to send data serially, or bit by bit, to the receiver.

Diagram 1.1 - RTL Diagram of Transmitter Entity

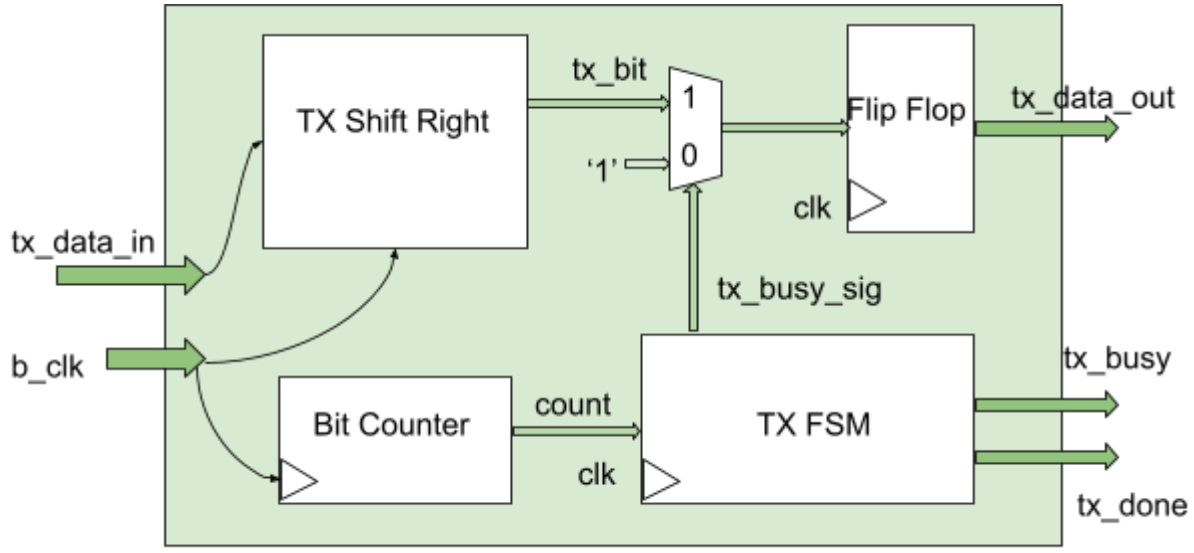


Table 1.1 - Transmitter Entity Port Description

Port	Direction	Type	Width (bits)	Notes
b_clk	IN	std_logic	1	Synchronised with receiver using baud rate generator
clk	IN	std_logic	1	CLOCK_50 used for internal computations
resetn	IN	std_logic	1	Asynchronous reset
tx_data_in	IN	std_logic_vector	8	8 bit data received by transmitter
tx_start	IN	std_logic	1	Control signal for valid data. Start signal for transmission. [active low]
tx_data_out	OUT	std_logic	1	Data bit sent to receiver.
tx_busy	OUT	std_logic	1	Control signal for ongoing transmission of data [active high]
tx_done	OUT	std_logic	1	Control signal for signalling transmission is done. [active high]

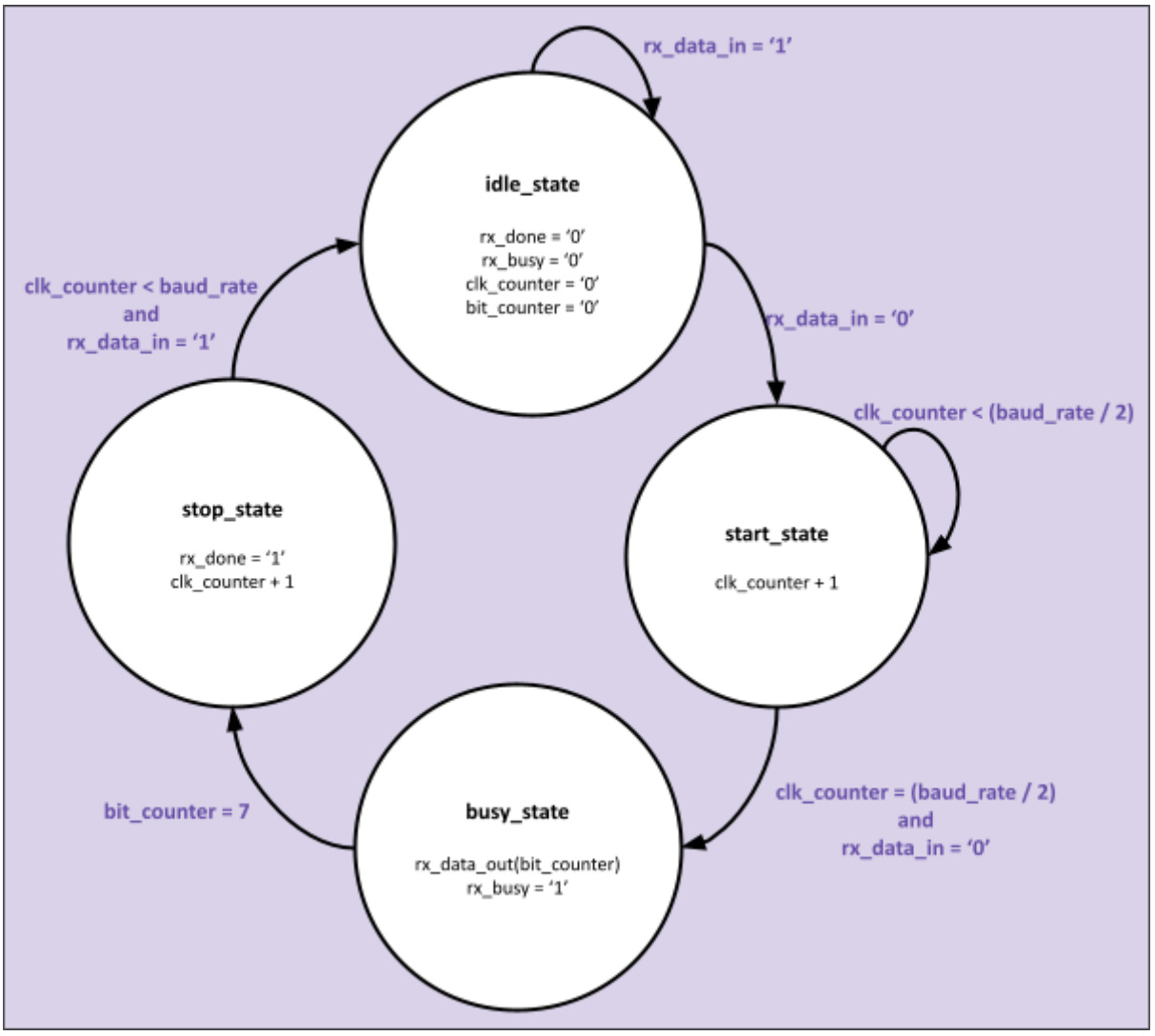
2. Receiver

Description

This module receives data serially, or bit by bit, from the device (computer). The 8 bit data is received LSB to MSB. This module will take the data received from the device, and display it on the HEX/LCD/LEDs displays on the FPGA board.

Modules

1. FSM: The FSM diagram for the receiver is below.



2. Bit Counter: The bit counter keeps track of the bits that are received and is incremented after each shift.
3. Shift Register: The shift register stores the 8-bit data by shifting right. This is because the order of bits received is LSB to MSB.

Diagram 2.1 - RTL Diagram of Receiver Entity

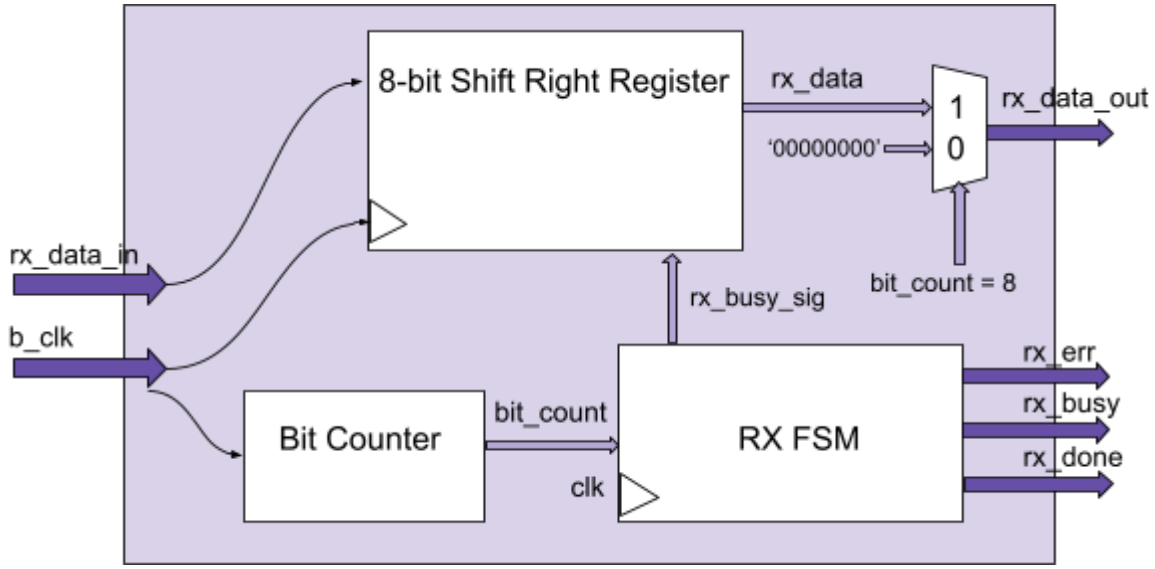


Table 2.1 - Receiver Entity Port Description

Port	Direction	Type	Width (bits)	Notes
b_clk	IN	std_logic	1	Synchronised with transmitter using baud rate generator
clk	IN	std_logic	1	CLOCK_50 used for internal computations
resetn	IN	std_logic	1	Asynchronous reset
rx_data_in	IN	std_logic	1	Data bit received from transmitter
rx_data_out	OUT	std_logic_vector	8	8-bit received data
rx_busy	OUT	std_logic	1	Control signal for ongoing reception of data [active high]
rx_done	OUT	std_logic	1	Control signal for signalling reception is done [active high]
rx_err	OUT	std_logic	1	Error signal for parity bit checking.

3. Data Specification

1. Baud Rate Generation

Description

This module is used to synchronise the transmitter and receiver. The output clock signal has a period that corresponds to the specified baud rate, ie. the rate at which the data is transmitted. Below is an example for calculating a 9600 baud rate based on the 50 MHz frequency of CLOCK_50.

Example baud rate generation calculation

$$clocks\ per\ bit\ =\ clock\ frequency\ /\ baud\ rate\ frequency$$
$$clocks\ per\ bit\ =\ 50\ MHz\ /\ 9600\ baud$$
$$clocks\ per\ bit\ =\ 5208.\overline{3}$$

Note: We would round this value to 5208

Table 3.1.1 - Baud Rate Generator Entity Port Description

Port	Direction	Type	Width (bits)	Notes
clk	IN	std_logic	1	CLOCK_50 input clock signal
resetn	IN	std_logic	1	Asynchronous reset
b_clk	OUT	std_logic	1	Baud rate generator clock output. Used to synchronise the transmitter and receiver

This table contains the details about the ports for the baud rate generator entity.

Note: The baud rate generator could be instantiated as a separate component, or within both the transmitter and receiver entities.

2. Data Framing

The input data to the receiver is 1 bit and the output data from the transmitter is both 8 bits long as shown in the above figure. We add 3 bits to the original data. bits 0 and 10 to are used to mark the start and end of the data. The start bit is always ‘0’ and the stop bit is always ‘1’. This bit is used to synchronise data in the receiver.

Diagram 3.2.1 - Dataframe after adding the start, stop and parity check

Bit 0 **Start='0'**	Bit 1 **(LSB)**	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8 **(MSB)**	Bit 9 Parity	Bit 10 **Stop='1'**

3. Error Detection

Transmitter

In the transmitter, a parity check will be done by comparing the number of 1's within the 8-bit data_in. If the number of 1's is ODD, we set a parity bit to high (1). If the number of 1's is EVEN, we set the parity bit to low (0). This parity bit will be sent serially to the device as bit 9 in the dataframe (or the 10th bit sent out).

Receiver

In the receiver, we receive a parity bit from the device as the 9th bit in the dataframe (or 10th bit received). We determine the parity of the 8-bit data_in using the method described above, and compare our computed parity to the parity bit and output an rx_err signal. If the parities match, rx_err is set to low (0), and if they don't we set the rx_err to high (1). This error signal will be indicated by a LED on the FPGA board.

4. Handshaking

Handshaking is used to halt transmission of data from the sending device until the receiving device has emptied the data buffer. In our design, we will have an output signal for both the transmitter and the receiver which indicates that the data is still being transmitted between the two devices. Once we reach the last bit to be transmitted, the done output would indicate that the transmission is over and then it will enter the idle state waiting for the start signal to turn high and begin a new data transmission process.

4. Top Level Design

Diagram 4.1 - Overall Top Level RTL Diagram