**Project Goal**

Transmitting 8 data bits through USB-UART on Basys 3.

**Implementation**

When users turn on / off any switch, this form a binary value of an ASCII character. These 8 bits will transmitted through the UART to the computer. The character will show in the terminal. All transmission is triggered when a button is pressed.

**Design Logic**

When the user pushes the button, it will trigger the transmission by setting the UART start bit from “1” to “0” (high voltage to low voltage). Button (Debounced button) creates the trigger signal for all transmission. Transmit is to transmit 8 bits from the FPGA to UART terminal. Receive is to receive the 8 bits from keyboard to FPGA. Clock create the clock divider and supply the clock to the whole design.

**Button module**

Use the debounced button module at <http://www.instructables.com/id/Debonucing-Button-on-Basys-3-Xilinx-FPGA-Developme/> and comment out signals that won’t be used.

**Transmitter module**

There are following inputs and outputs

|  |  |  |
| --- | --- | --- |
| **Inputs / Outputs** | **Signal name** | **Description** |
| Input | clk | Master clock |
| Input | reset | Reset signal |
| Input | transmit | btn signal to trigger the UART communication |
| Input | [7:0] data | 8 bits data transmitted. No parity bit |
| Output | TxD | Transmitter serial output. Serial data frames are transmitted via this pin. TxD will be held high during reset, or when no transmissions are taking place. |

There are following internal signals (variables)

|  |  |  |
| --- | --- | --- |
| **Type of signals** | **Signal name** | **Description** |
| Register (reg) | [3:0] bitcounter | 4 bits counter to count if 10 bits data transmission complete or not |
| Register (reg) | [3:0] counter | 14 bits counter to count the baud rate |
| Register (reg) | [1:0] state | Initial state variable for Mealy State Machine |
| Register (reg) | [1:0] nextstate | Next state variable for Mealy State Machine |
| Register (reg) | [9:0] rightshiftreg | 10 bits data needed to be shifted out during transmission. For storing the serial package and sending its bits one by one. When the load signal is 1, it is loaded with a new character to transmit (8 bits). The least significant bit is initialized with the binary value “0” (a start bit) A binary value “1” is introduced in the most significant bit |
| Register (reg) | shift | Signal to show shifting data is ready |
| Register (reg) | load | Signal to show loading data is ready |
| Register (reg) | clear | Signal to clear all signals |

**Baud rate generator**

The master clock frequency on Basys 3 is 100MHz. We need baud rate of 9600 for UART transmission. To make a baud rate generator, we will use a counter. For example, the baud rate is only 9600 bits per second (bps).

baud rate \* counter = master clock frequency

counter = master clock frequency / baud rate

Therefore the counter value is 10,417 for the baud rate of 9600.

**UART Transmission**

When the counter for baud rate count up to 10415 (from 0 to 10415 = 10416) for a baud rate of 9600, we will reset the counter to be 0 and then change the state in the finite state machine. In other words, the counter is used to slow down the 100MHZ FPGA clock for our program to work at 9600 baud. The reason for doing this is to make the program synchronous to the clock (or counter). Therefore the transition from IDLE to TRANSMIT and vise versa only happens at each 10416 count of the counter. Counter is also reset each time it reaches 10416 so that it can start to count up to 10416 again.

After that, we will check if **clear**, **shift** and **load** signals are asserted (“1”) one by one. If the load signal is asserted (“1”), the rightshiftregister loads the data that is to be transmitted and add start and stop bits. This is why, once the state goes from IDLE to TRANSMIT, load goes high. It ensures that data is loaded into the register and is ready to be transmitted. When shift is asserted (“1”), rightshiftreg = rightshiftreg>>1. This shifts the data right one time. Bitcounter is also incremented once. This tells the program that one bit was already sent. When clear is asserted (“1”), we have finished transmitting and bitcounter goes back to zero; waiting for another new transmission.

**Finite State Diagram**

We will use synchronous [Mealy State Machine](https://en.wikipedia.org/wiki/Mealy_machine) to implement the transmit module.

1. IDLE State (S0):

If the transmit is zero, we stay at the IDLE state and keep waiting until that button is pressed.

Set load, shift and clear signals to 0. This prevent shifting when the transmission doesn’t start. Do not set the bitcounter to zero. Otherwise, we can’t increment the bitcounter when we are transmitting the data. We do not have any data to load from just yet. TxD is held high to show that nothing is being transmitted (IDLE State).

We move to the TRNSMIT State when the input **transmit** is 1. This **transmit** is the push button on Basys 3 we press to tell the program we are going to transmit. Set the load to “1” (high) but keep shift and clear signal to be “0”(low).

1. TRANSMIT State (S1):

Remember, we have 8 data bits, 1 start bit and 1 stop bit. When the bitcounter reaches 10 (equal or greater than 10), it means that it has completed its task of transmitting. When we reach our goal of a 10 bit transmission, the clear must be “1” (high) to clear the bitcounter back to zero and get ready for the next transmission. Also the next state must go back to IDLE. Keep load and shift to be “0” (low)

If bitcounter is less than 10, we stay in TRANSMIT state. Also, we must put shift to 1 because we want to shift our rightshift register so the next bit can be shifted onto our TxD output pin for UART transmission. This is done in TxD = shiftreg[0]. Load and shift are “0” (low).

Reference:

<http://www.ehow.com/how_5667684_create-uart-transmitter-verilog-hdl.html>