IFRO Digital & Verification Summer School

UART transmitter design

UART (Universal Asynchronous Receiver-Transmitter) is a module used in asynchronous serial communication. Serial communication has the advantage of using only one wire for transmitting data from one device to another. At the reception, the data will be reconverted from serial to parallel and used as usual.

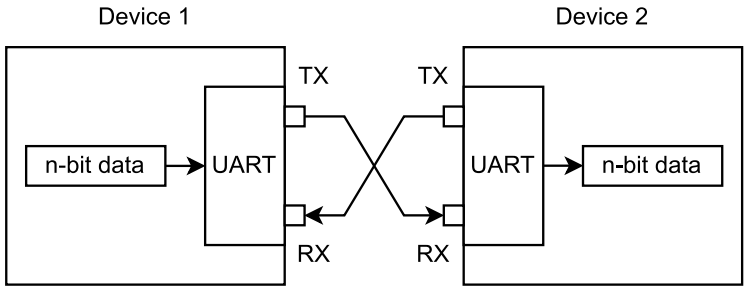

Figure 1 UART communication

The input data is transmitted bit by bit, starting with the least significant one (bit 0). Optionally, a parity bit can be transmitted with it to help detect errors. All this information is framed by a start bit (0) and one or two stop bits (1). The idle state of the line is 1. The transmission speed is controlled by the baud rate (symbol rate).

A typical UART frame for 8-bit data is presented in the figure below:

Start	Data								Parity	Stop	Stop
0	B0	B1	B2	B3	B4	B5	B6	B7	P	1	1

Figure 2 Typical UART frame

During our design sessions, we will implement an UART module capable of transmitting 8-bit data with a parity bit (XOR on all bits) and a single stop bit.

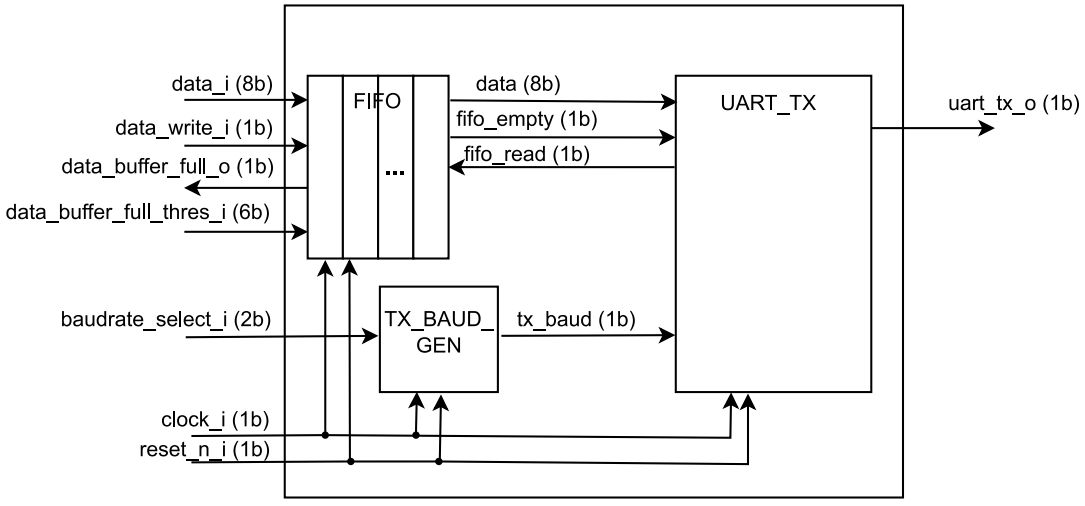

Figure 3 Block diagram of the UART_TRANSMITTER module

The main component of the UART transmitter is the UART_TX. It has the role of getting one data byte from its input, wrapping it in a frame, and sending it through the *uart_tx* output.

The interface of the system with the UART_TX will be done through a FIFO structure capable of storing a queue of data. The data will be delivered in the order of its arrival from the system to the UART_TX input when it is available. The conversion will start automatically when the FIFO has some date stored in it and end when the FIFO becomes empty.

The TX_BAUD_GEN module is capable of generating the signal that will control the baud rate (the speed of transmitting data). The baud rate will be derived from the *clock* signal using the *baudrate_select* signal.

1 FIFO BUFFER

The FIFO buffer is a storage structure that complies with the First-In, First-Out rule. Therefore, from this memory, the oldest written data will be read at each access. The FIFO memory does not need reading or writing address buses. The addresses are internally computed using counters with enable signals.

For ease of understanding, we can consider the FIFO a circular memory, with read and write addresses that "follow" each other. Memory tends to be "filled" by writing data and "emptied" by reading data.

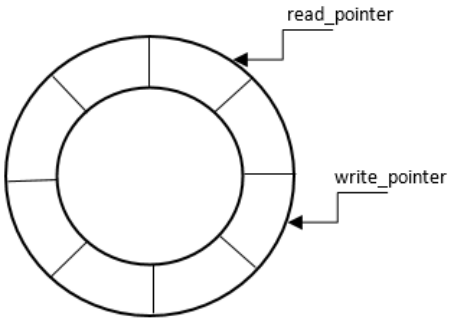

Figure 4 Circular memory

The detailed internal structure of the FIFO is described below:

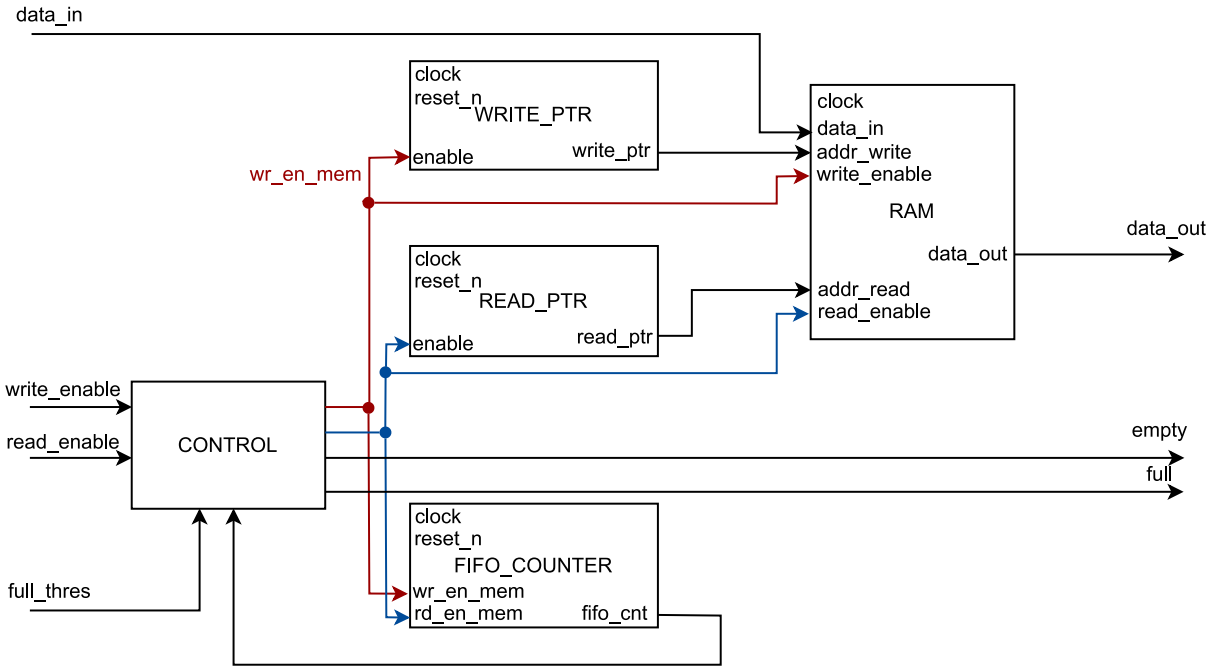

Figure 5 FIFO internal structure

The main components of the FIFO are:

- **RAM**: RAM memory with 32 8-bit memory locations and separate ports for write and read operation. The memory read operation is asynchronous.

- **CONTROL**: combinational circuit which generates the following control signals:
  - *wr_en_mem*: indicates that a FIFO write operation can be performed. It is 1 if a *write_enable* request is sent on the interface and the FIFO is not full.
  - *rd_en_mem*: indicates that a FIFO read operation can be performed. It is 1 if a *read_enable* request is sent on the interface and the FIFO is not empty.
  - *empty*: indicates that the FIFO is empty (no memory location contains data).
  - *full*: indicates that the FIFO is full (all memory locations are occupied).
- **WRITE_POINTER**: counter with enable that delivers the write memory address. This counter will be incremented if a memory write operation can be performed (*wr_en_mem* is 1).
- **READ_POINTER**: counter with enable that delivers the read memory address. This counter will be incremented if a memory read operation can be performed (*rd_en_mem* is 1).
- **FIFO_COUNTER**: counter that counts the number of occupied memory locations. It will be incremented when a write without read memory operation is performed (*wr_en_mem* is 1 and *rd_en_mem* is 0) and decremented when a read without write memory operation is performed (*wr_en_mem* is 0 and *rd_en_mem* is 1).

1.1 Module interface

Signal	Type	Dimension	Description
clock	input	1 bit	Clock signal with a 50% duty cycle
reset_n	input	1 bit	Active 0 reset signal
write_enable	input	1 bit	Write request from the user. This can be accepted if there are free memory locations in the FIFO (FIFO not full), or ignored otherwise.
read_enable	input	1 bit	Read request from the user. This can be accepted if there is data in the FIFO (FIFO not empty), or ignored otherwise.
data_in	input	8 bits	Input data for the FIFO.
full_thres	input	6 bits	Signal used to control the threshold from which the FIFO returns a full state.
data_out	output	8 bits	Output data of the FIFO.
empty	output	1 bit	Empty status bit. This is 1 when the FIFO has no occupied memory locations.
full	output	1 bit	Full status bit. This is 1 when the number of occupied memory locations reaches the value of *full_thres*.

1.2 Tasks:

1) Identify the dimensions of the three counters (WRITE_PTR, READ_PTR, FIFO_COUNTER).
2) Implement the circuit in Verilog. All the logic should be implemented in the same module.
3) Implement a short test bench to test the circuit. In the test bench you should write the FIFO until it becomes full and then read it back until it becomes empty. You can use a constant value during simulation for *full_thres*.

2 TX_BAUD_GEN

This module is used to generate a periodic signal called *tx_baud* that will be used as baudrate reference by the UART transmitter. The signal frequency will be derived from the clock frequency. Unlike the clock signal, the duty cycle is not 50%. During one period, *tx_baud* will be in High state for only one clock period, as it is shown in the figure below:

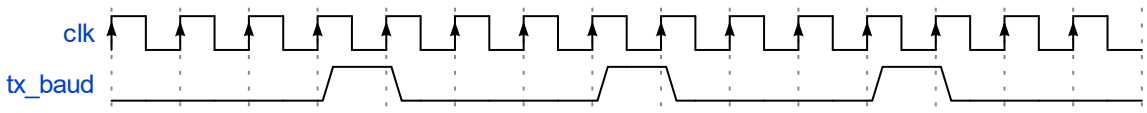

Figure 6 *tx_baud* having a period equal to Tclk/4

2.1 Module interface

Signal	Type	Dimension	Description
clock	input	1 bit	Clock signal with a 50% duty cycle
reset_n	input	1 bit	Active 0 reset signal
baudrate_select	input	2 bits	This signal is used to select the period of the output signal: 0: the period of the output signal is Tclk /16 1: the period of the output signal is Tclk /32 2: the period of the output signal is Tclk /64 3: the period of the output signal is Tclk /128
tx_baud	output	1 bit	Periodic signal with frequency derived from the clock frequency. During a period, the signal will be in High state for only one clock period.

2.2 Tasks:

1) Propose a circuit that can generate the required signal.
2) Implement the circuit in Verilog.
3) Implement a short test bench to test the circuit. In the test bench you have to test the correct generation of all four possible periods.

3 UART_TX

UART_TX is the module that extracts data from the FIFO when available and converts it to the UART format. The simplest way to implement data serialization is with a shift register. In order to properly control the behavior of the shift register and correctly pack the data, we will also need a finite state machine.

An overview of the UART_TX module is presented in the figure below:

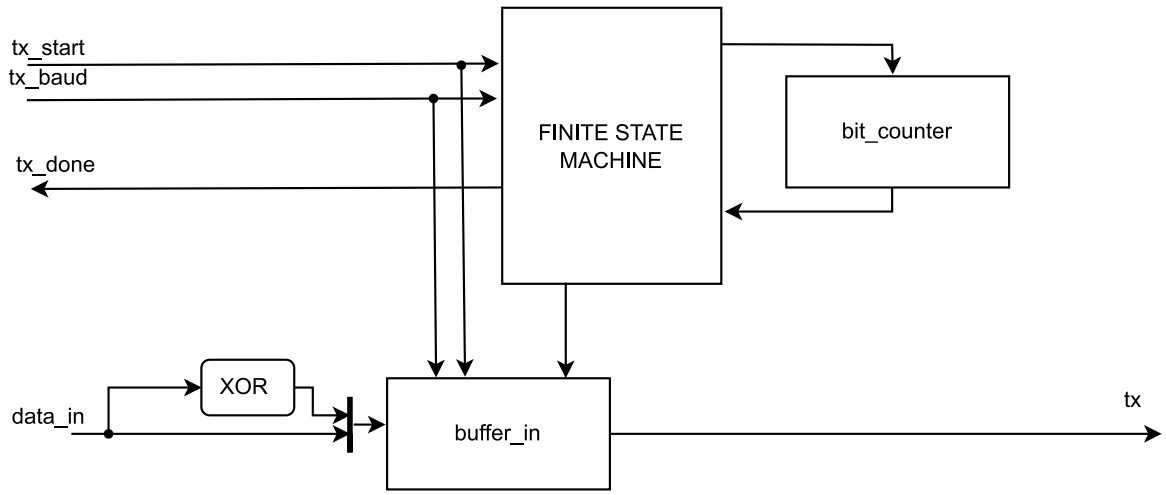

Figure 7 Overview of the UART_TX module

The finite state machine controls the activity in the module. It has four states, corresponding to the four stages of the transmission:

- **IDLE**: The FIFO is empty and there is no data that needs transmission. In this state, the *tx* is 1, while waiting for data from FIFO. If a byte is written to the FIFO, the *tx_start* signal will become 1, preparing the state machine for transmission by initializing the *buffer_in* with the data read from the FIFO and the computed parity bit.
- **START**: The finite state machine is in this state when the start bit (0) is transmitted. When the bit time period passes (given by the baud rate), the finite state machine will go to the DATA state when the actual data transmission occurs.
- **DATA**: In this state, the data and the parity bit are transmitted. The finite state machine will remain in this state as long as there are still bits to be transmitted. This can be measured using a counter (*bit_counter*), incremented each time a bit time period passes.
- **STOP**: The finite state machine is in this state when the stop bit (1) is transmitted. After this, it will go back to the IDLE state.

According to the finite state machine description, we will also need additional resources to implement the functionality of the UART_TX module:

- *buffer_in*: a 9-bit shift register (8 data bits and 1 parity bit) which is initialized in the IDLE state when a new data comes from the FIFO and shifted one position to the left during the DATA state. The periodicity of the shift is given by the periodicity of the baud rate signal. The tx output will

always get the value of bit 0 and can optionally be synchronized with the clock.

- *bit_counter*: a 4 bit counter used to count the number of bits transmitted during the DATA state. It is initialized with 0 in the IDLE state.

3.1 Module interface

Signal	Type	Dimension	Description
clock	input	1 bit	Clock signal with a 50% duty cycle
reset_n	input	1 bit	Active 0 reset signal
tx_start	input	1 bit	Signal that controls the UART transmit start. It will be 1 as long as the FIFO is not empty and there is data waiting to be transmitted.
tx_baud	input	1 bit	Baudrate signal generated by the TX_BAUD_GEN module.
data_in	input	8 bits	Input data from the FIFO.
tx_done	output	1 bit	Signal set to 1 when the UART_TX module finished to transmit one byte.
tx	output	1 bit	UART TX output.

3.2 Tasks

1) Draw the graph of the finite state machine.
2) Propose a detailed architecture of the circuit.
3) Implement the circuit in Verilog, considering all the previously presented elements.

4 UART TRANSMITTER

UART_TRANSMITTER includes all the previous modules, properly connected.

4.1 Tasks

1) Starting from the architecture presented in Figure 3, propose a detailed connection of the three submodules.
2) Implement the UART_TRANSMITTER module by instantiating the submodules and connecting them accordingly.
3) Implement a short test bench to test the circuit. In this test bench data will be sent to the FIFO until it becomes full. Check on the waveform that all the data bytes are correctly transmitted, in the correct order.