Project: UART - reception part

1. Introduction

The aim of the project is to acquire basic skills in the design and implementation of digital circuits. Learn to describe these circuits in VHDL and gain experience with their simulation using the Modelsim tool.

As an example of the component on which you will acquire these skills, we will use the component for receiving and transmitting data over an asynchronous serial line (UART - Universal Asynchronous Receiver-Transmitter).

For simplicity, we will implement only a selected part of this controller, more specifically we will focus on the receiving part responsible for processing data from the serial line and their reconstruction (parallelization). Compared to a full-fledged UART controller, we will consider a number of other simplifications so that the overall complexity of the project does not exceed the tolerable limit.

2. Asynchronous serial communication

Asynchronous serial communication has become an essential way of transferring data between computers and peripherals. It is currently used mainly in the field of embedded systems.

For data transmission between two nodes, one data wire is sufficient, along which the individual data bits are sent successively, from the semantically lowest bit (LSB) to the semantically highest bit (MSB).

Asynchronous serial communication means that the transmitted bits are not synchronized by any additional signal such as a CLK signal. The receiver is able to recognize incoming bits based on the data encoding used.

The transmission line is always set to the log level before the transmission of each multi-bit word (usually a byte) begins. 1.

The transmission of a multi-bit word begins with the so-called START bit set to the log level. 0. Transmitting this START bit (ie switching from log. 1 to log. 0) will allow the receiver to reliably identify the start of the transmission.

After the START bit, the individual bits of the data word are subsequently transmitted from the least significant bit (LSB) to the most significant bit (MSB).

The last bit of the data word (LSB) is followed by one or more so-called STOP bits, which are always set to the log level. 1.

After the STOP bit, the transmission of the next data word can start, starting with the START bit. Please note that the STOP bit of the previous data word in combination with the START bit allows reliable detection of the next data word (transition from log. 1 to log. 0).

An example of transmitting an 8-bit data word with one STOP bit is shown in Figure 1.

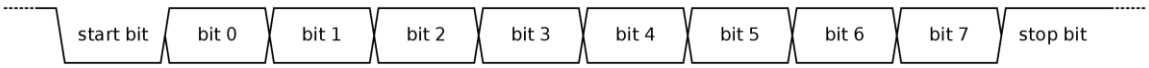


Figure 1. Example of serial transmission of an 8-bit data word with one STOP bit

For reliable detection of the transmitted word on the receiver side, it is necessary not only to identify the beginning of the transmission (through the transition from log. 1 to log. 0), but also to know at what speed the data is transmitted. Therefore, the transmitter and receiver must first be set / configured for the same baud rate.

The baud rate is given in the number of baudes transmitted per second, with one baud corresponding to one bit in this case. The basic and also the most frequently used baud rate is 9600 baud per second. If we consider the transmission of 8-bit data words bounded by one START bit and one STOP bit (a total of 10 bits), then we are able to transmit up to 960 bytes per second (9600/10) at a speed of 9600 baud.

In addition, in order for the receiver to reliably identify the values (logic levels) of the transmitted data bits, it is recommended that this circuit operate at a rate 16 times faster than the selected baud rate. The data bit should then be scanned in the middle of the single bit transmission interval, as shown in Figure 2[1].

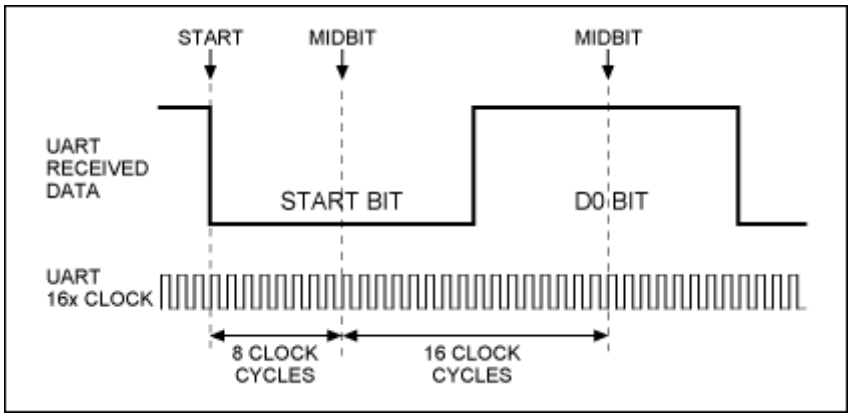


Figure 2. Example of data bit sampling in the middle[2]

[1] The specification recommends scanning the log. the level of input data in the 7th, 8th and 9th clock cycles and use the majority of these three values as the resulting bit. For simplicity, however, let's settle for the value captured at the end of the 8-hour cycle

[2] Retrieved from https://electronics.stackexchange.com/questions/207870/uart-receiver-sampling-rate

3. Work procedure

1. To develop a project, you will need a tool for simulating digital circuits described in VHDL. The Modelsim tool from Mentor Graphics is preferred, which will also be used to evaluate this project. See Chapter 4 for more information on how to obtain and use this tool.

2. Download the archive of source files from the MS Teams environment (Project channel, Files tab) *projekt.zip* and get acquainted with its contents. There are four files in the archive:
    - *uart.vhd* - VHDL source file with UART_RX component interface definition and empty architecture to be added.
    - *uart_fsm.vhd* - source file in VHDL language, which will be used to describe the finite state machine controlling other components of your UART_RX circuit.
    - *uart_tb.vhd* - source file in VHDL language, which represents a sample Testbench file for testing the basic functionality of your proposed UART_RX component.
    - *uart.fdo* - auxiliary script in TCL language, which is used to start the simulation of your proposed UART_RX circuit in the Modelsim environment.

3. Design a circuit for receiving data words over an asynchronous serial line (UART).
    - Refer to the basic information on processing asynchronous serial communication in Chapter 2.
    - Consider the input data stream in the format: START bit, 8 data bits, 1x STOP bit, sent at 9600 baud per second. The receiving circuit will operate at a 16x higher frequency (CLK signal) compared to the baud rate of the individual data bits. Your task will be to scan the data bits in the middle of the transmitted interval (see figure 2).
    - The circuit will receive individual bits on the DIN input port, perform their deserialization and write the resulting 8-bit word to the DOUT port. Confirm the validity of the data word on the DOUT port by setting the DOUT_VLD signal to the log level. 1 for one clock cycle of the CLK clock signal. An example of a timing diagram showing the expected waveform of the signals on the input / output ports of the UART_RX component is shown in Figure 3.

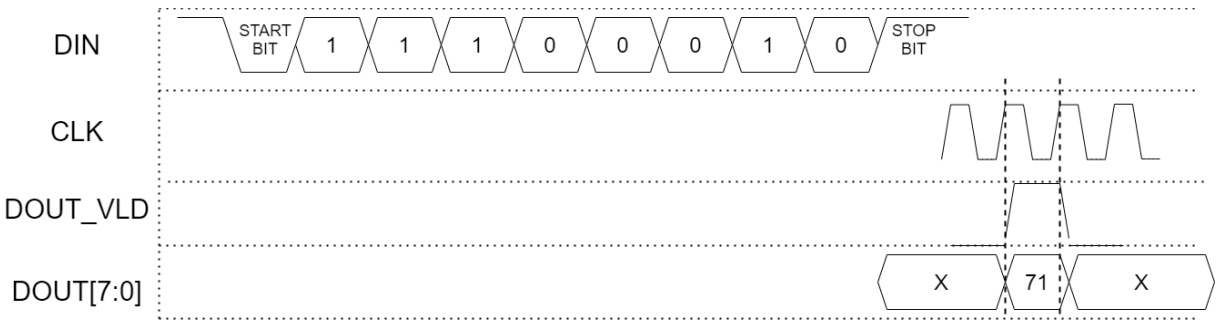


Figure 3. Example of time course on inputs and outputs of UART_RX circuit

- The individual parts of the circuit will need to be controlled through a finite state machine (*Finite State Machine).* First, compile a graph of the transitions of this automaton.
- When designing, be sure to treat the asynchronous input to the UART_RX synchronous network to reduce possible metastable states.

4. Implement the proposed circuit in VHDL and save it in a pre-prepared file *uart.vhd.* Insert the code corresponding to the finite automaton into a separate file for clarity *uart_fsm.vhd.*
5. Simulate VHDL code using Modelsim and verify its correct functionality.
6. Create a technical report that will include:
    - Name, surname and login.
    - The architecture of the designed UART_RX circuit at the RTL level and its brief description.
    - Graph of finite state machine transition and its brief description.
    - Screenshot with a demonstration of simulation time courses capturing the transmission of one data word.

    An example of the output report can be found in Appendix 1. The scope of the report should not exceed three A4 pages.

7. The outputs of the project will consist of:
    - VHDL source files (*uart.vhd* and *uart_fsm.vhd).*
    - File *zprava.pdf* with output message (in PDF format)

Pack all three files in an archive named <*login*> *.zip.*

Before submitting this archive to the information system, please test this archive through a set of test scripts available in the information system in the file *student_test.zip.* Detailed instructions for testing can be found in the attached README file.

Submit the tested archive via the information system no later than the date specified in the information system at the deadline marked Project. Subsequent submission of the project will not be taken into account.

After the experience of previous years, one more important warning!

According to the FIT Dean's Directive supplementing the BUT Study and Examination Regulations (FIT Dean's Decision No. 34/2010), Re Article 11, paragraph 4:

"All tests, projects and other assessed assignments are prepared by the student individually, unless the project or assignment has been explicitly assigned to a specified group of students."

In case of detection of plagiarism or illegal cooperation on the project, the student will be rewarded by not awarding credit in the subject INC (0 points per project). Alternatively, by summons before the disciplinary commission according to the Disciplinary Code for students of the Faculty of Information Technology of the Brno University of Technology.

4. Circuit simulation using the Modelsim tool

Obtaining the Modelsim tool

For the purpose of developing projects for courses focused on the design of digital circuits, we have prepared an image of a virtual machine for you, where all the necessary tools are installed, including Modelsim, which you will need in this course.

Download the image of this virtual machine from FITkit's private site (fitkit-vbox- file 202103.7z).

Unzip / decompress the file using the application 7z.

Install a freely available program to start this machine VirtualBox.

Starting the simulation

Modelsim is a commercial tool that regularly polls the license server and checks the validity of the purchased license. This license server is located in the fit.vutbr.cz domain and requires a VPN connection.

Please follow the instructions on the following pages and always activate the VPN connection before starting the virtual machine and Modelsim.

The actual start of the Modelsim tool (inside a running virtual machine) can be done either via an icon on the desktop, the Start menu or from the command line via the command:

```
everyone
```

After displaying the graphical user interface of Modelsim, it is possible to start simulating the circuit. This requires:

- Compile the source files through the command vcom.
- Switch Modelsim to simulation mode through the command everyone.
- Add monitored signals to the window labeled Wave.
- Run the simulation through the command run.

To avoid having to run this set of commands with each simulation, Modelsim offers the ability to create and run TCL scripts. The following commands are simply inserted into the script and this script is run through the command:

```
to <script name>
```

In addition, the actual execution of Modelsim can be combined with the execution of this auxiliary TCL script via the following command from the command line:

```
vsim -do <script name>
```

For comfortable work, we have prepared a sample TCL script designed specifically for this a project called uart.fdo and run it in one of the following ways:

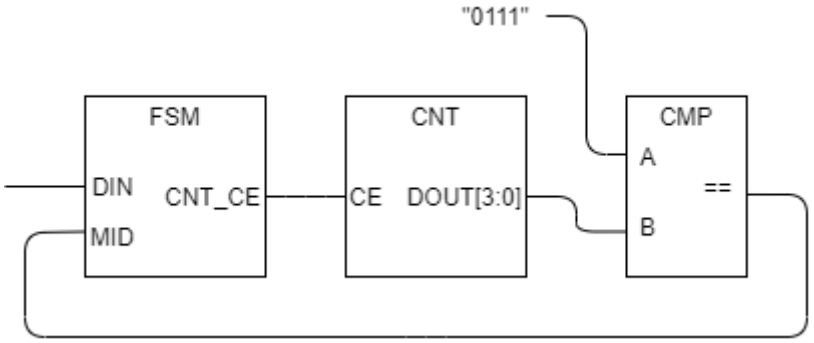
1. do uart.fdo (from Modelsim environment)
2. vsim -do uart.fdo (from the command line)

Appendix No. 1: Sample output report

Name:
Login:

Designed circuit architecture (at RTL level)

Circuit diagram



Comment:

- For clarity, do not include CLK and RESET control signals in the diagram.

Function description

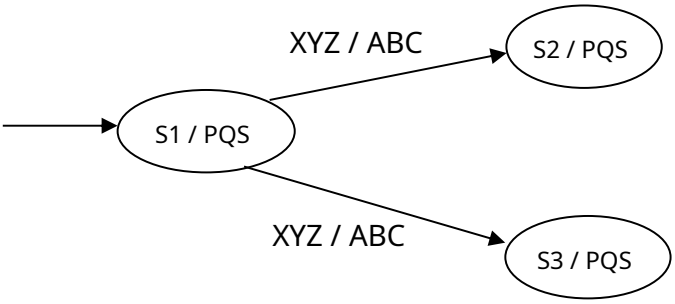
Brief verbal description of the structure and function of the circuit (max. Half of an A4 page)

Finite State Machine

Scheme of the automaton

Legend:

- State of the automaton: S1,
- S2, S3 Input signals: X, Y, Z
  Mealy outputs: A, B, C
- Moore outputs: P, Q, S



Comment:

- Use appropriate names for states, input signals, and output signals to make their meanings easier to understand.
- Insert the values 0, 1 or X (don't care) directly into the graph for the input / output signals XYZ, ABC and PQS.

Function description

Brief verbal description of the automaton function (max. Half of an A4 page)

Screenshot from simulations

Please insert a picture (screenshot) here that demonstrates the functionality of your circuit at the simulation level. Please capture the transmission of at least one data word. For clarity, you can orient the image, for example, to the width of the page.