Design of UART controller for FPGA with Simulink®

1 Author(s)

Kralev, Jordan ; Department of Systems and Control Technical University of Sofia Sofia, Bulgaria

Simple UART (Universal Asynchronous Receiver Transmitter) controller, compatible with RS232 standard is implemented on Spartan-3E FPGA, through automatic HDL code generation and hardware synthesis. The paper shows how the design can be entirely conveyed in MATLAB/Simulink® environment. There is no need for additional behaviour description in HDL. Results from simulation and experiments verify functionality of the controller.

Published in:
Embedded Computing (MECO), 2014 3rd Mediterranean Conference on

Date of Conference:
15-19 June 2014

Page(s):
44 - 47

Print ISBN:
978-1-4799-4827-7

Conference Location :
Budva, Montenegro

DOI:
10.1109/MECO.2014.6862655

Publisher:
IEEE

Design of UART controller for FPGA with Simulink®

Jordan Kralev
Department of Systems and Control
Technical University of Sofia
Sofia, Bulgaria
jkralev@yahoo.com

Abstract— **Simple UART (Universal Asynchronous Receiver Transmitter) controller, compatible with RS232 standard is implemented on Spartan-3E FPGA, through automatic HDL code generation and hardware synthesis. The paper shows how the design can be entirely conveyed in MATLAB/Simulink® environment. There is no need for additional behaviour description in HDL. Results from simulation and experiments verify functionality of the controller.**

FPGA design flow, Simulink® HDL Coder, digital system simulation

I. INTRODUCTION

The purpose of the paper is to demonstrate a possible approach to design of classical UART controller for Spartan 3E FPGA. There are many ready to use products in the market [1,2], however some applications may require building a custom communication device or modification of a classical solution.

MATLAB/Simulink® environment is appropriate for high-level specification of information processing systems. Simulink® allows dataflow modelling and state space representation of hybrid dynamical systems. Also it supports hierarchical structures and modulation, which are natural ways for managing complexity [3]. MATLAB® is high-level algorithmic language oriented to engineering calculations. Simulink HDL coder® is a tool that can produce VHDL or Verilog description from Simulink® diagrams. Generated code is compatible with standard synthesis tools and environments offered by FPGA vendors, for example Xilinx® Inc.

According to general OSI model [4], which had proved to be appropriate for development of large and reliable communication systems, an interconnection between communicating devices must have layered architecture. For embedded applications, typical communication system is composed of Application Layer, Data-Link Layer (DLL) and Physical Layer (PHL) [5].

UART (Universal Asynchronous Receiver Transmitter) controller is a serial communication device. Typically it implements Data-Link Layer and the upmost sub layer (PLS) of Physical Layer. RS-232 is a standard concerning electrical levels of data and control signals between data terminal equipment (DTE) and data communication equipment (DCE) [6]. In addition it specifies the physical size and pin out of connectors. Hence it corresponds to MDI and PMA sub layers of Physical Layer in the OSI model.

For transmission, the controller converts a byte to a sequence of bits, then forms a message frame by addition of protocol information (Fig. 1) and transfers the bits one by one. Reception is a reverse process. Asynchronism results from undefined timing between consecutive messages because there is no shared timing system between devices. Each message is composed of one start bit, 5-8 data bits, 1 to 2 stop bits and optional parity bit. Parameters for the presented in the paper UART controller are: 115200bps baud rate, 8 data bits, 1 stop bit, no parity.

Figure1. Schematic representation of typical UART data frame

II. SIMULINK MODEL

Fundamental principle in Simulink modelling is to represent every system in its state space. Each system can be treated as a map:

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} = F\begin{pmatrix} x_{k-1} \\ u_k \end{pmatrix}$$

x_k - current state of the system; x_{k-1} - previous state of the system; u_k - input from external environment; y_k - output to external environment; F – system's map or transfer function

To model an UART controller in Simulink, we need to represent it in such a form. The controller is composed of two subsystems – Receiver module (Fig. 2) and Transmitter module. For modelling of digital devices it is convenient to represent corresponding Simulink model in discrete time. Simulation configuration is set to Fixed Step Discrete Solver with Sample Time $T_s = 1$ (clock period).

Unit Delay blocks store the current state of each module. It acts as a D-Trigger, according to equation

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} = \begin{pmatrix} u_k \\ x_{k-1} \end{pmatrix}$$

Embedded MATLAB Function block represents system's transfer function. This block allows describing system behaviour in algorithmic language of MATLAB®. The block

executes each time step only once, according to static relationship $y_k = F(u_k)$.

Input port and *Output port* blocks in Simulink model correspond to *in port* and *out port* declarations of the generated entity definition in VHDL.

Each Simulink signal is associated with specific data type for data it carries. *Data Type Conversion* blocks are used to change the data type of a signal.

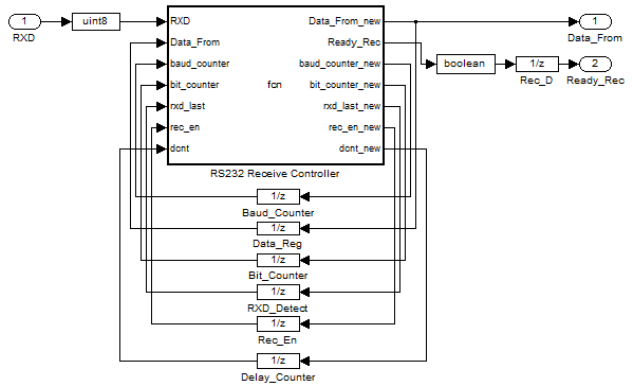

Figure2. Simulink diagram for UART receiver

A. Receiver

Receiver module has one input – state of RXD signal line. It has two outputs. First is the content of data register, containing last received byte. Second is a logical signal, asserted when reception is completed, to inform next processing layer.

Receiver module has 6 states (Table 1). Three of them are for counters. Others are for storage of received byte, for detection of start bit (rxd_last) and for stage of reception. Reception is a three stage process. First a start bit is detected. Second a delay counter is waited to finish. Third a sequence of bits is shifted into data register.

TABLE I. STATES OF RECEIVER

Name	Size	Description
Data_From	8 bit	Data register
baud_counter	9 bit	Baud rate counter
bit_counter	4 bit	Number of received bits
rxd_last	1 bit	Last state of RXD line
rec_en	2 bit	Stage of reception
dcnt	5 bit	Delay counter

In the third stage a baud rate counter is used to synchronize the proper time of bit capture from RXD line. A bit counter stores the number of received bits. A delay counter is used to generate small delay after start bit detection to assure proper bit capture. A pseudo code for Receiver module is:

```
if (start bit detected) then
    initialize baudrate counter;
    initialize delay counter;
    initialize bit counter;
    stage = delay;
elseif (stage = delay) then
    increment delay counter;
    if (delay counter = N_D) then
        stage = reception;
    end
elseif (stage = reception) then
    increment baudrate counter;
    if (baudrate counter = N_115200) then
        reinitialize baudrate counter;
        Data_Reg = (Data_Reg shl 1) bitor RXD;
        increment bit counter;
        if (bit counter = 8) then
            stage = idle;
            rise ready signal;
            reinitialize bit counter;
        end
    end
end
```

This algorithm is coded in MATLAB® language in the *Embedded MATLAB Function* block.

B. Transmitter

Transmitter module has two input ports. First is data byte to be transmitted and second is signal line to trigger transmission event. The module controls the state of TXD output line. When a byte is sent a ready signal is generated. Simulink model of Transmitter module is analogous to that of Receiver module (Fig. 2). The difference is on signal and algorithmic level.

Transmitter module has 5 states (Table 2). Two of them are for counters. Others are for storage of transmitted byte, for storage of TXD line state and for stage of transmission. Transmission is a three stage process. First is an initialization stage of counters and internal shift register. Second state is for generation of a start bit. And in the third stage the sequence of bits is shifted from data register to the TXD line.

TABLE II. STATES OF TRANSMITTER

Name	Size	Description
Data_To	8 bit	Data register
baud_counter	9 bit	Baud rate counter
bit_counter	4 bit	Number of sent bits
txd_last	1 bit	Last state of TXD line
send_en	2 bit	Stage of transmission

A baud rate counter is used to achieve the desired transmission speed. A bit counter is used to indicate when the transmission has finished. A pseudo code for Transmitter module is:

```
if (send request detected) then
      initialize baudrate counter;
      initialize bit counter;
      Data_Reg = Data_To_Send;
      TXD line is low (start bit);
      stage = transmit;
elseif (stage = transmit) then
      increment baudrate counter;
      if (baudrate counter = N_115200) then
            TXD line = Data_Reg (bit 0);
            Data_Reg = Data_Reg shr 1;
            increment bit counter;
            reinitialize baudrate counter;
            if (bit counter = 8) then
                  stage = stop bit;
                  TXD line is high (stop bit);
            end
      end
elseif (stage = stop bit) then
      increment baudrate counter;
      if (baudrate counter = N_115200+N_D) then
            rise ready signal;
      end
end
```

III. HDL CODE GENERATION

HDL code generation from Simulink model is possible only if certain subset of supported blocks is used [7]. Embedded MATLAB Function, Unit Delay, Data Type Conversion are all supported. For every atomic block in the model there is corresponding section in VHDL file. This allows for verification of the generated code. In our case we have one main file and two subsystems corresponding to Receiver and Transmitter module. Generated VHLD code is about 700 lines.

Unit Delay Block is represented in VHDL as a *process* block which is sensitive to clock signal. When clock rise event is detected then the input signal is assigned to the state signal.

Embedded MATLAB Function block is represented in VHDL as a *process* block [8] with a sensitivity list composed of input ports of Simulink block. Sensitivity list of *process* block contains temporary variables used in MATLAB Function and auxiliary variables needed to represent arithmetic and logic operations in HDL. The body of *process* block is translation of algorithm defined in MATLAB Function.

From generated VHDL code we build a Xilinx ISE project file. It contains the files corresponding to Simulink model and additional constraint file. UCF Location Constraints including I/O pin assignment and the I/O standard used must be specified in Xilinx ISE project according to target board documentation [9].

The Xilinx Synthesizer (XLS) generates representation of VHDL code in technological basis of target FPGA device [7]. We can see from the reports that Unit Delay blocks are represented by D-Trigger components. The algorithm in an Embedded MATLAB Function block is represented by combinatorial circuit. It contains elementary logic gates, summator and comparator components. This description can be easily optimized and embedded in FPGA device by Xilinx tool chain [7]. Some numerical characteristics of FPGA resource utilization are presented in Table 3, together with a comparison to a functionally equivalent Verilog based GPL solution (named mmuart [1]).

TABLE III. RESOURCE UTILIZATION

Parameter Name	Used/Available	Benchmark
Number of Slice Flip Flops	56/9312	61
Number of 4 input LUTs	231/9312	119
Number of occupied Slices	123/4656	69
Number of bounded IOBs	23/232	61
IOB Flip Flops	2	34

IV. RESULTS

Design of the UART controller is validated on several levels corresponding to different stages of design process which are:

- Simulation in Simulink
- Behaviour simulation of VHDL code
- Real experiments

Figures 3,4,5,6 represent simulation results in Simulink environment for Transmitter module. The purpose of simulation is to transmit byte 73 (letter 'I') two consecutive times. To have correct results for time scale we set the sample time, $T_s = 20ns$.

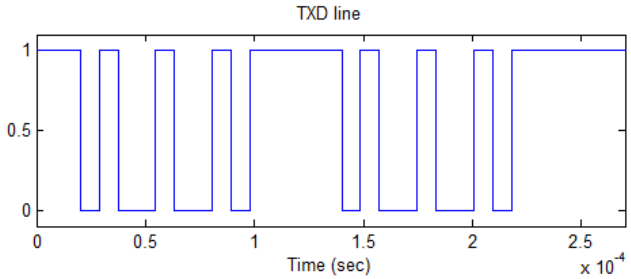

Figure3. Logical state of TXD line

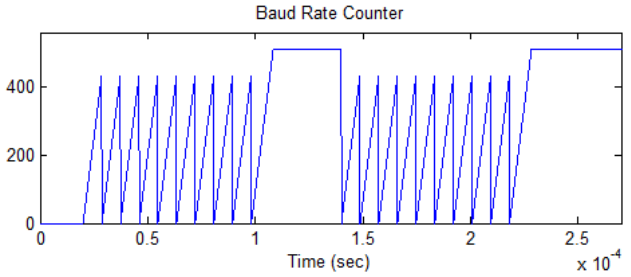

Figure4. State of baud rate counter

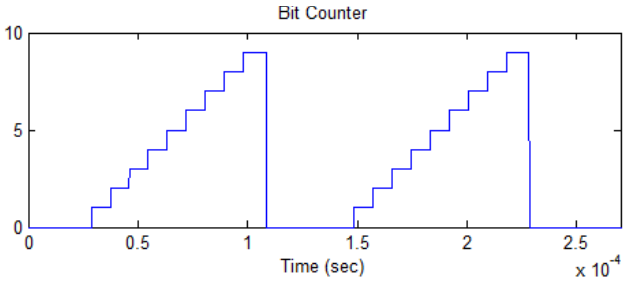

Figure5. State of bit counter

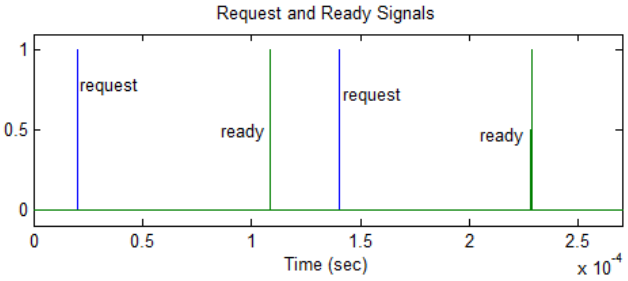

Figure6. Events of start and end of transmission

Fig. 7 represents result from behaviour simulation of generated VHDL code conveyed in Xilinx ISE. Again the byte 73 (letter 'I') must be transmitted. The period of clock signal is set to 20ns.

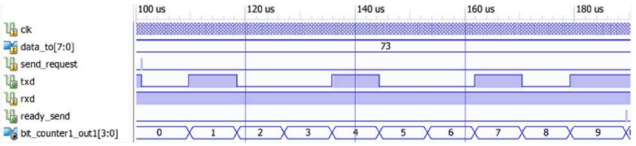

Figure7. ISim simulation results

Real experiments are done by connecting desktop PC to Spartan 3E Starter Kit board by RS232 twisted pair cable. From MATLAB environment we can use *serial* command to open serial communication port. Then by *fread* and *fwrite* commands we can receive or send data. We send a sequence of data bytes to FPGA and wait to receive the same sequence back. A diagram of experimental setting is on Fig. 8.

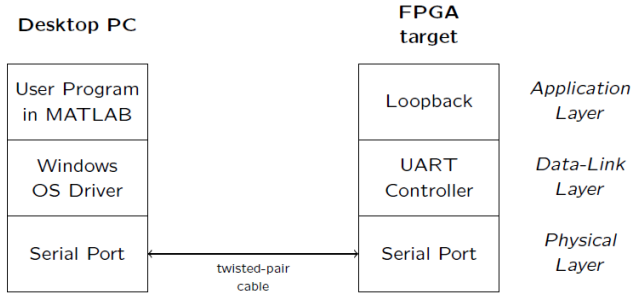

Figure8. Experimental setting for UART controller

V. CONCLUSION

From Table 3 is evident that the UART design is comparable with the benchmark solution in terms of device resource utilization. This proves the efficiency of the Simulink specification and the generated code. Proposed design solution doesn't handle special channel conditions and lacks some features like FIFO and flow control, which may lead to lower reliability in comparison to other implementations [1,5,10,11,12,13]. The aim of the paper wasn't maximal functional completeness, but to demonstrate an approach for hardware design by Simulink specification.

From a simple Simulink diagram and about 60 lines of MATLAB code, 700 lines of VHDL code is generated. In this approach the designer doesn't fully control the generated behavioural description and must trust to proper mapping of Simulink blocks to blocks of VHDL. However simulations results, before and after generated code, undoubtedly show that both descriptions are functionally equivalent. Also empirical experiments support that result. What is gained is a lot shorter design time and complex applications [14].

ACKNOWLEDGMENT

This research has been supported and funded by Technical University of Sofia, project No. 142ПД0008-08.

REFERENCES

[1] OpenCores community. URL: www.opencores.org, accessed at: February 2014.

[2] Xilinx IP cores database. URL: www.xilinx.com/products/intellectual-property, accessed at: February 2014.

[3] H. Simon, The Architecture of Complexity, in Proceedings of APS, 1962, pp. 467-482.

[4] ISO/IEC 7498-1, Information technology - Open Systems Interconnection – Basic Reference Model: The Basic Model 1996

[5] I. Maykiv, A. Stepanenko, D. Wobshall, et al., Software-hardware method of serial interface controller implementation. Computer Standards & Interfaces. 34 (2012) 509-516.

[6] EIA standard RS-232-C, Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange. URL: www.ecaus.org, accessed at: February 2014.

[7] Xilinx Inc., Command Line Tools User's Guide, URL: www.xilinx.com, accessed at: February 2014.

[8] IEEE Standard 1076-1987, VHDL Language Reference, URL: www.ieee.org, accessed at: February 2014.

[9] Digilent Inc., Spartan 3E Starter Kit Users Guide, URL: www.digilentinc.com, accessed at: February 2014.

[10] H. Chun-zhi, X. Yin-shui, W. Lun-yao, A universal asynchronous receiver transmitter design. in Proceedings ICECC, 2011, pp. 691-694.

[11] L. Ali, R. Sidek, I. Aris, et al., Design of a micro-UART for SoC application. Computers & Electrical Engineering. 30 (2004) 257-268.

[12] Z. Hu, J. Zhang, X. Luo, A Novel Design of Efficient Multi-channel UART Controller Based on FPGA. Chinese Journal of Aeronautics. 20 (2007) 66-74.

[13] Z. Shen, Y. Wang, The Design of UART Controller Based on FPGA , in: T. Honghua (Ed.), Informatics in Control, Automation and Robotics vol 1, Springer, 2011, pp. 221-225.

[14] E. Grayver, Implementing Software Defined Radio. Chapter 11, first ed., Springer, New York, 2013.