CONTROLLING FPGA WITH MATLAB

Marek Bártů, Jakub Šťastný

Dept. of Circuit Theory, Faculty of Electrical Engineering
Czech Technical University of Prague

Abstract

In this article we describe simple UART (Universal Asynchronous Receiver and Transmitter) controller macro developed at our department. The macro implements simple proprietary communication protocol suitable for controlling application specific macro in FPGA (Field-Programmable Gate Array) This paper also describes developed Matlab communication framework for FPGA .

1 Introduction

We are interested in biomedical digital signal processing on FPGA (brain-computer interface or voice processing) at the FPGA laboratory at our department. We implement various DSP algorithms in FPGA (Field-Programmable Gate Array). For on-chip verification, measuring and realization itself we use various FPGA development kits - prototyping boards. All these kits are equipped with JTAG (Joint Test Action Group) programming interface, but it is suitable only for testing purposes and for bitstream loading into FPGA. However, we demanded another controlling interface between the master PC and slave FPGA board for loading of the coefficients, DSP system configuration, setting of various parameters, reading measured values and similar purposes.

2 Specification

For these purposes we did not demanded a fast link. The interface was intended for control purposes, not for data transfers. The interface should had ability to control prototyping board directly with particular commands and also we planned to use it within Matlab environment. There were also demands to use hardware available on FPGA prototyping boards. We needed an intelligent protocol, independent on target application and easy to use. We also wanted to use the controller as an example for students in the future.

3 Controller Description

We decided to use standard UART (Universal Asynchronous Receiver and Transmitter), also known as the serial interface. Our decision was also supported by the availability of all the necessary hardware (logic level shifters) right on prototyping boards. Practically usable baud rate achievable with the serial interface is sufficient for controlling purposes as well. To unify the communication link between various expected designs we designed a simple ASCII based communication protocol. The protocol allows to control the target application directly from terminal emulating program (for example standard windows Hyperterminal).

In the FPGA, the controller is implemented as a simple, two layer core. It is shown in Figure 1. The first (link) layer consist of UART core. It is an implementation of UART receiver, transmitter and buffers provided by Xilinx as a free IP core. This particular core is not target independent, it is aimed only to Xilinx devices. Anyway, for the different FPGAs or ASICs, similar cores implementing UART (Link layer) could be used. We have not tested another core, but there are a lot of free available cores, possible to adapt, for example at [8].

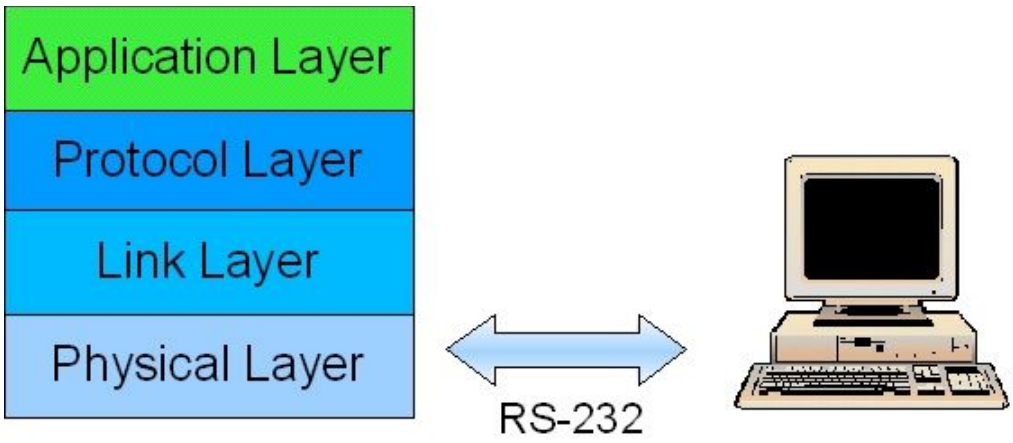

Figure 1: Communication scheme

The second (protocol) layer consists of the UART controller macro equipped with address, data and controlling signals. This part is our own design. The second layer is based on FSM (finite state machine) that implements the protocol logic.

The protocol controller has two groups of ports:

- first, there are link interface ports. This group consist of two only signals: *serial_in* and *serial_out* are connections to voltage-level transducer for RS232 at the Physical layer (outside the FPGA).

- The second group is group of system signals *clk*, which is a clock input, and *res_n* signal for resetting.

- The third, last group is application interface. It contains the following signals: *reg_in* (input) and *reg_out* (output) are independently controlled and are thought to be used for connection with logic out of memory mapped space. Memory mapped space is connected via *addr* (32-bit wide address), *data_in* (8-bit wide data input from user macro to controller – to personal computer), *data_out* (8-bit wide data output from controller to user macro – from personal computer), *rd* (data read) and *wr* (data write) ports. Functions of ports are similar to that used at static RAMs (Random Access Memory).There is also an address register auto-incrementation feature, so blocks of data could be easily read and write. The macro is equipped with address, data and controlling signals. All the commands and data transmitted via the serial interface are passed to these buses.

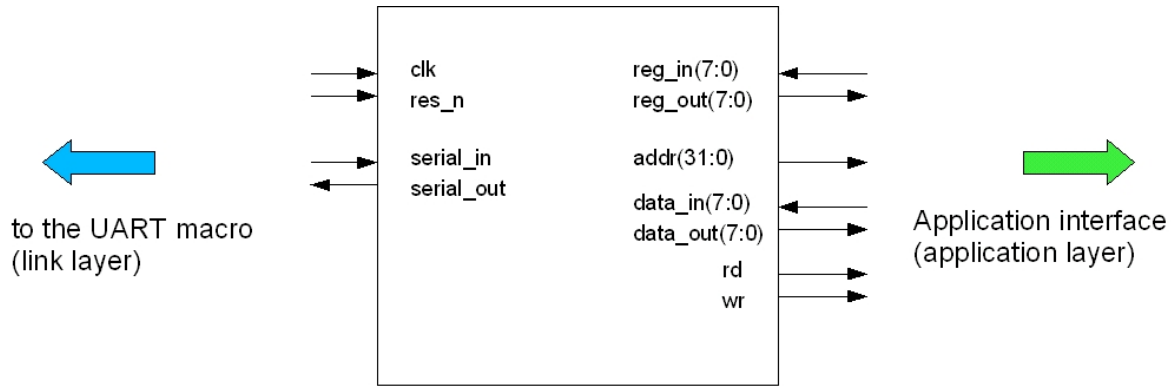

Figure 2: Controller macro (protocol layer)

4 Hardware Parameters

VHDL-RTL description was used for implementation. The controller has been successfully synthesized and proved on-chip. We used Digilent prototyping board with Xilinx Spartan3 XC3S200 chip. The size of the whole controller, including UART macros, is 196 slices. The minimum clock period is 9ns (110MHz). If maximum optimization for area is used, the whole UART controller occupies 89 slices, but the minimal clock period increase to 24ns (42MHz). It gives up opportunity to use all possible baud rates up to 115kbps. We also successfully implemented the macro another board at the FPGA laboratory, equipped with Xilinx XC2V600 chip.

5 Verification Environment

For verification of the interface design and future verification of applications using the controller we developed a simple verification environment. The VHDL source of the environment is a part of the controller pack available at WWW pages of the FPGA laboratory [1]. The verification environment top-level scheme is drawn in Fig. 3. The whole environment is contained in entity *control*. The part in FPGA is represented by entity *fpgatop*. The main signals are reset (*b_reset*), clock (*b_clk*), serial line input (*sin*) and serial line output (*sout*). The verification environment works in two mutually exclusive modes:

1. Simple mode – In this mode, the whole contents of the input file *input.txt* is send to *fpgatop* with minimal time spaces between. Every line of input file contain ASCII decimal value of char to send. All the data sent by the FPGA are written to *output.txt* file.

2. Timing mode – By setting generic variable *timing_enable* to *'0'* we can activate mode sending data with defined delay between them. These spaces are in *in_timing.txt* file. Every line contain number which indicates the amount of clock cycles before send of data on the same line in *input.txt*. file. The delays between output data characters from *fpgatop* are written to *out_timing.txt* file.

These two modes allow either simple verification of a simple system, or a more complex approach with a precise-defined timing.

The source of that verification environment is a part of the UART controller pack. The verification environment is tested in ModelSim XE III/Starter 6.0d (part of Xilinx ISE8 WebPack) freely available at [7]. The detailed description of verification environment is in [5] or [7]. Some parts of the system are based on a generic verification environment [9].

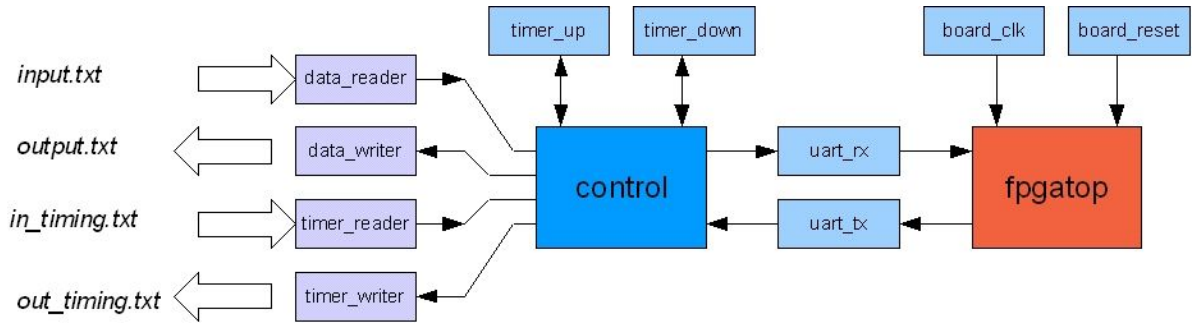

Figure 3: The verification environment

6 Protocol

The controller command set contains only five commands. The following operation accessible with these commands: data reading, data writing, changing address, control output set and control input read. Commands are briefly described in Table 1.

command	*description*	*example*
A	set address register	A1234
R	read from current address	R
W	write data to current address	W98
G	get control signal status	G
S	set address register	S78

Table 1: CONTROLLER COMMANDS DESCRIPTION

All the commands and also the data are transferred in ASCII format. Because of the ASCII format, the throughput is a bit lower than in the case the UART without any protocol is used. When there are demands of use the whole possible bandwidth, the UART IP could be used alone. Detailed protocol description is a part of [3].

7 Matlab Environment

On the PC side we need only the serial interface. For controlling target application, it is possible to use any software, which is able to communicate with the serial port. As a part of the controller, we have developed support functions for Matlab and verification environment in VHDL (Very high-speed integrated circuits Hardware Description Language). The Matlab environment is described in the text bellow.

function	*description*
board_read	read data from specified address
board_write	write data to address
board_get	get control signal status
board_set	set control signals

Table 2: MATLAB FRAMEWORK FUNCTIONS

The whole communication support pack for Matlab consist of four functions. In fact, they implement the commands of the protocol. For successful using of these functions, it's necessity to open communication port with standard Matlab command *fopen*. At the end of communication, port need to be closed using *fclose* command. The reference to the open port is given to function via descriptor variable (*my_uart* in code bellow). All the used functions are described in the table bellow. For clear understand there is also short Matlab code that demonstrate whole communication process. It is also possible to use event driven style of communication, but it is not described here. The function syntax and more information is in [4].

Simple demonstration code is below:

```
% parameters
port = 'COM3'; rate = 38400;   % default in VHDL design

% setup serial port
my_uart = serial (port,'BaudRate',rate,'DataBits',8,'Parity',...
     'none','StopBits',1,'Terminator','CR');
```

```
% open the port
fopen(my_uart);

%%%%%%%%%%%%%%%%%%
% work with port %
%%%%%%%%%%%%%%%%%%
board_write(my_uart,1234,[0,1,255])   % write data to FPGA
data_block = board_read(my_uart,1234,3)   % read block from FPGA
board_set(my_uart,status)   % set independent output
statut = board_get(my_uart)   % read independent input

% close port
fclose(my_uart);
```

8 Demo

For demonstration and education purposes we prepared simple demo application. The controller is connected to a seven-segments digit display and there is a memory block of 12 × 8 bits connected to system address and data buses. This simple peripherals are available from Matlab. This demo is aimed to introduce controller and its usage to students. Students can simply add their own peripherals and verify it right in Matlab.

9 Conclusion

We have designed, completed and verified the controller of the serial line protocol and implemented all the associated software to enable Matlab communicate with the FPGA kit. The design is usable for controlling our research FPGA systems as well as an educational example of such a systems. All the features and functions are well documented and a verification framework allowing a simple simulation of the prospective design is also a part of our system.

The controller could be synthesized in freely available ISE WebPack software pack [7]. and simulated in ModelSim XE edition which allows our students to work with the system even at their homes. The simulation could be done without hardware, just on the PC.

Right now we plan use the controller with the voice activity detector design on FPGA which is currently under development.

The controller with its documentation and the complete verification environment is freely available for non-commercial (educational and research) purposes at the WWW pages of our laboratory [1]. Prospective users are kindly asked to cite this work in their publications.

Acknowledgement

This work has been supported by the research program Transdisciplinary research in Biomedical Engineering II No. MSM6840770012 of the Czech Technical University in Prague.

References

[1] FPGA and Brain Computer Interface Research Laboratory Site at http://amber.feld.cvut.cz/fpga/ .
[2] K. Chapman. *UART Transmitter and Receiver Macros.* Xilinx Ltd. 2003.
[3] RS-232 page on Wikipedia at http://en.wikipedia.org/wiki/RS-232 .
[4] M. Bártů. *Demo of the UART Xilinx FPGA macro block + Matlab communication framework.* Unpublished technical report Z05-6 (in Czech), FEE CTU Prague, 2005. Available at http://amber.feld.cvut.cz/fpga/ .
[5] M. Bártů. *Implementation of communication protocol between the FPGA kit and the PC via the serial interface.* Unpublished technical report Z06-3 (in Czech), FEE CTU Prague, 2006. Available at http://amber.feld.cvut.cz/fpga/ .
[6] ***Matlabu R14 Help*** – section *Matlab/External Interfaces.*
[7] Xilinx ISE WebPack. Available at http://www.xilinx.com/ise/logic_design_prod/webpack.htm .
[8] OpenCores.org at http://www.opencores.org .
[9] L. Ručkay. V*erification environment for FPGA*. Unpublished technical report Z05-3 (in Czech), FEE CTU Prague, 2005. Available at http://amber.feld.cvut.cz/fpga/ .

Marek Bártů
Czech Technical University, Faculty of Electrical Engineering, Dept. of Circuit Theory
Technicka 2, 166 27 Prague 6, Czech Republic
email: bartum1<*at*>feld<*dot*>cvut<*dot*>cz

Jakub Šťastný
Czech Technical University, Faculty of Electrical Engineering, Dept. of Circuit Theory
Technicka 2, 166 27Prague 6, Czech Republic
email: stastnj1<*at*>feld<*dot*>cvut<*dot*>cz