Journal of Siberian Federal University. Engineering & Technologies, 2016, 9(2), 189-196
~ ~ ~

УДК 621.391-026.51:004.4′2

Platform Independent Implementation of High Speed Serial Communication Based on FPGA

Chanchal Raj[a] and Santosh Mohan Rajkumar[b]*
[a]Department of Electronics & Communication Engineering
National Institute of Technology
Silchar, India
[b]Department of Electrical Engineering
National Institute of Technology
Silchar, India

Received 03.06.2015, received in revised form 09.08.2015, accepted 23.12.2015

In various embedded system applications, the high speed multi-serial communication is necessary. Various embedded systems require DSP (Digital Signal Processing) to process information & FPGA to control the peripheral devices. UART (Universal Asynchronous Receive Transmit) is designed in FPGA & connected to DSP so as to meet the real time capability of system along with compact, stable & reliable data transmission. In this paper we propose a software-implementation technique of an UART to get a platform independent UART-core for high speed serial communication in FPGA. Here the core is written in Verilog & implemented using XILINX ISE.

Keywords: FPGA, Verilog, UART.

Citation: Raj C., Rajkumar S.M. Platform independent implementation of high speed serial communication based on FPGA, J. Sib. Fed. Univ. Eng. technol., 2016, 9(2), 189-196, DOI: 10.17516/1999-494X-2016-9-2-189-196.

* Corresponding author E-mail address: craj.nits@gmail.com, rajkumar.nits@gmail.com

Платформа независимой реализации высокоскоростной последовательной связи на основе FPGA

Чанчал Радж[а], Сантош Мохан Ражкумар[б]
[а]Кафедра электроники и техники связи
Национальный технологический институт
Силчар, Индия
[б]Департамент электротехники
Национальный технологический институт
Силчар, Индия

В различных приложениях необходимы встроенная система, высокая скорость мульти-последовательной связи необходимо. Различные встроенные системы требуют DSP (цифровой обработки сигналов) для обработки информации FPGA для управления периферийными устройствами. UART (Universal Asynchronous Transmit) предназначен в FPGA и DSP и подключен таким образом, чтобы удовлетворить в реальном времени возможности системы наряду с компактной, устойчивой и надежной передачей данных. В этой статье мы предлагаем технику программного обеспечения Внедрение UART, чтобы получить платформу независимого UART-ядра для высокоскоростной последовательной связи в FPGA. Здесь ядро написано в Verilog и реализовано с использованием Xilinx ISE.

Ключевые слова: FPGA, Verilog, UART.

Introduction

It goes without saying that serial communication is advantageous over parallel. Asynchronous serial communication has advantages of less transmission line, high reliability & therefore it is widely used in exchanging data between computer & peripherals. Asynchronous serial communication is usually implemented by UART (Universal Asynchronous Receiver Transmitter). UART permits full-duplex communication in serial link for short distance off-board data transfer, thus has been widely used in the data communications, embedded control systems. In real world applications only a few of the key features of UART are required. Specifically interfaced UART chip means waste of resources & increased cost, which is undesired problem. Therefore in this research we are trying to implement an UART-core in FPGA using Verilog which effectively solves the above problem.

The UART protocol is an asynchronous serial connectivity protocol that takes data in bytes in parallel & transmits that data as individual bits in a sequential manner. At the target end, another UART rearranges the bits into complete bytes. UART protocol is widely used for short serial transmission interface, low-cost communication. The UART usually does not directly interact with external signals between different items of equipment. We use separate interface devices to convert the logic level signals of the UART to and from the external signaling levels.

Proposed architecture

A. Character Encoding

In our UART architecture, each data is transmitted with a Start Bit (S) at logic low which shows transmission is about to begin, 8-bit data bits (D0-D7), a parity bit (PB) to determine the data received without error (this is optional) & Stop Bit(s)(P) at logic high to mark the end of a data transfer. The time interval from a start bit to stop bit is termed as a frame (Fig. 1).

B. UART Receiver

All tasks of the UART hardware are controlled by a clock signal which runs at a multiple of the data rate. The receiver checks the state of the incoming signal on each clock pulse, looking for the beginning of the start bit. UART receiver receives data in serial manner & converts it into parallel data (Fig. 2).

The proposed receiver Finite State Machine has five states:

- Start state : the receiver Finite State Machine awaits the start bit, when the receiver is reset.

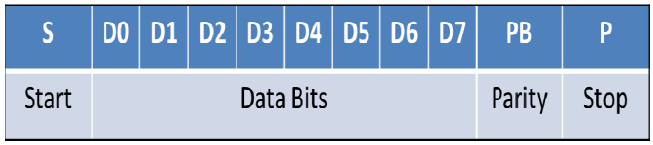

Fig. 1. Data Frame

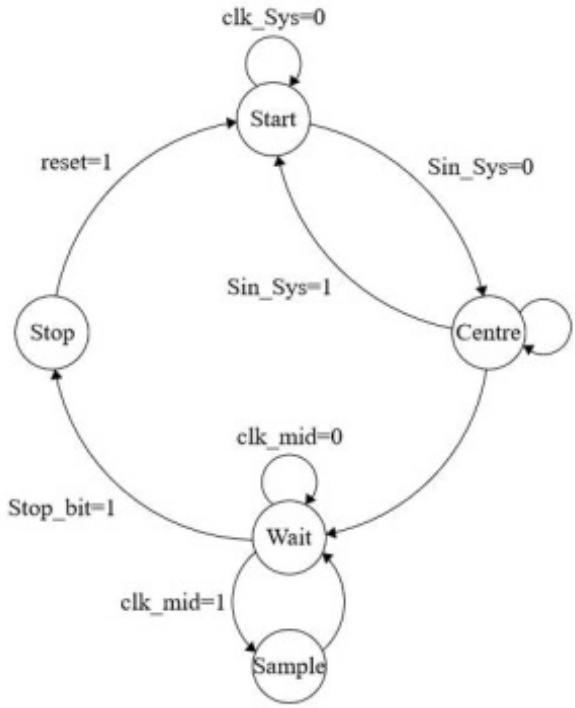

Fig. 2. Receiver Finite State Machine

- Centre state: receiver detect data after synchronizing the clock pulse. If baud rate synchronization fails then it returns to Start state.
- Wait state: it determines the length of bit frame for the state machine. After getting stop bit, next stop state comes.
- Sample state: when data bit has been sampled and tested, state machine would always transfer into Wait state and wait for the next data bit to come.
- Stop state: after getting new data, it changes the state to Start state.

C. UART Transmitter

The UART transmitter takes data in parallel manner & transmits the data serially in basic UART frame format. Transmitter sends data after every transmitter clock cycle and according to Baud rate (Fig. 3).

The proposed transmitter Finite State Machine has five states:

- Rest state: before transmitting signal is initiated, the state machine remains in this state. After feeding data into input data bus of UART, Start state comes.
- Start state: in this state clock synchronization takes place after the transmitter transmits the start bit (logic 0 signal) & the state of the machine changes to P2S (Parallel to Serial Conversion state)
- P2S state: this state waits until data conversion from parallel to serial completes. After conversion, Shift state comes. Also, it is checked whether data frame has been completed or not. After completion of data frame state changes to Stop state.
- Shift state: The state machine immediately returns to P2S state after shifting the data to Tx.
- Stop state: in this state, stop bit (logic 1 signal) is transmitted, after completing the data frame, the machine returns to Rest state.

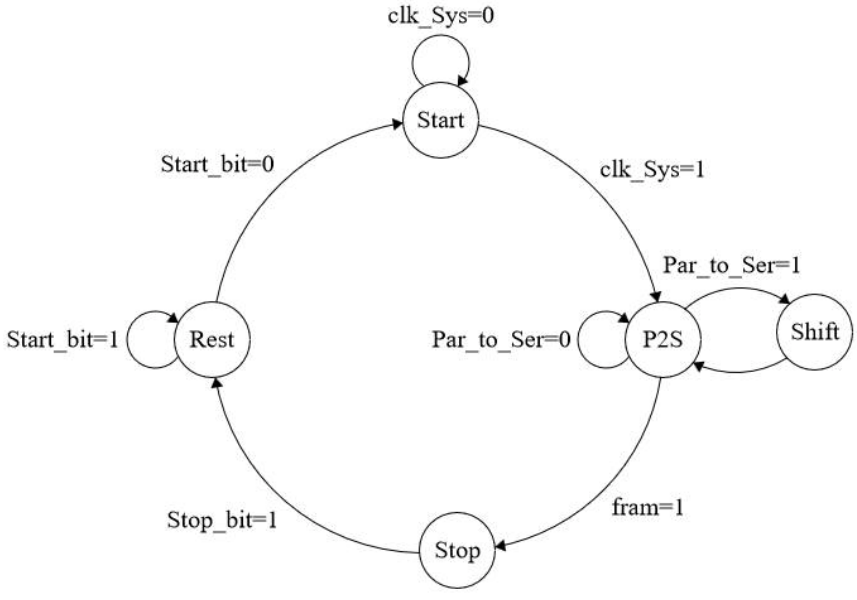

Fig. 3. Transmitter Finite State Machine

Table 1. Bit pattern of control register

C7	C6	C5	C4	C3	C2	C1	C0

Bit description: C7: – don't care; C6: – no. of stop bits; C5, C4: – data bits; C3: – parity enable/disable bit; C2, C1, C0: – baud rate setting bits.

Table 2. Divisors for Baud rates

Clock = 50 MHz			
Baud rate	Divisor	Baud rate	Divisor
57600	54	4800	651
19200	162	2400	1302
9600	325	1200	2604

D. Control Register

The control register is programmable register. The no. of data bits to be transmitted, the baud rate, the no. of stop bits, whether parity is present & the type of parity is determined by programming of this register (Table 1).

E. Baud rate generator

Baud rate generator is actually a clock divider circuit. It divides the system clock with suitable divisor to generate the desired baud rates. In this proposed model Baud rate error is minimized automatically. Table 2 represents the divisors used for the baud rate generated. The system clock is considered as 50 MHz.

The proposed uart module

The UART top module as shown in Fig. 1 has i) two data bus (bus_in & bus_out) used for data in or data out purpose in parallel form to the processor or from the processor it is connected with, ii) One clock (clk) line for baud rate control i.e. activity control, iii) one chip selection (cs) line, iv) two lines (Rx & Tx) for receiving & transmitting data to or from the device it is connected with, v) two lines (interrupt & nRW) for handshaking purpose with the processor it is communicating with, vi) one line (opcode) for programming the registers & selecting the operations, vii)one line (reset) for reset purpose.

The UART module has 4 internal blocks:

- Baud_Generator
- Operational_Unit.
- Interface_Unit.
- Tx_Rx_Unit.

A. Simulation results

In this simulation we are making parity bit disabled & using one stop bit.

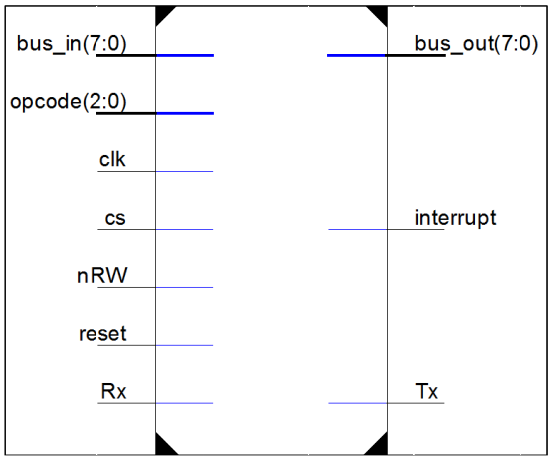

Fig. 4. UART top module

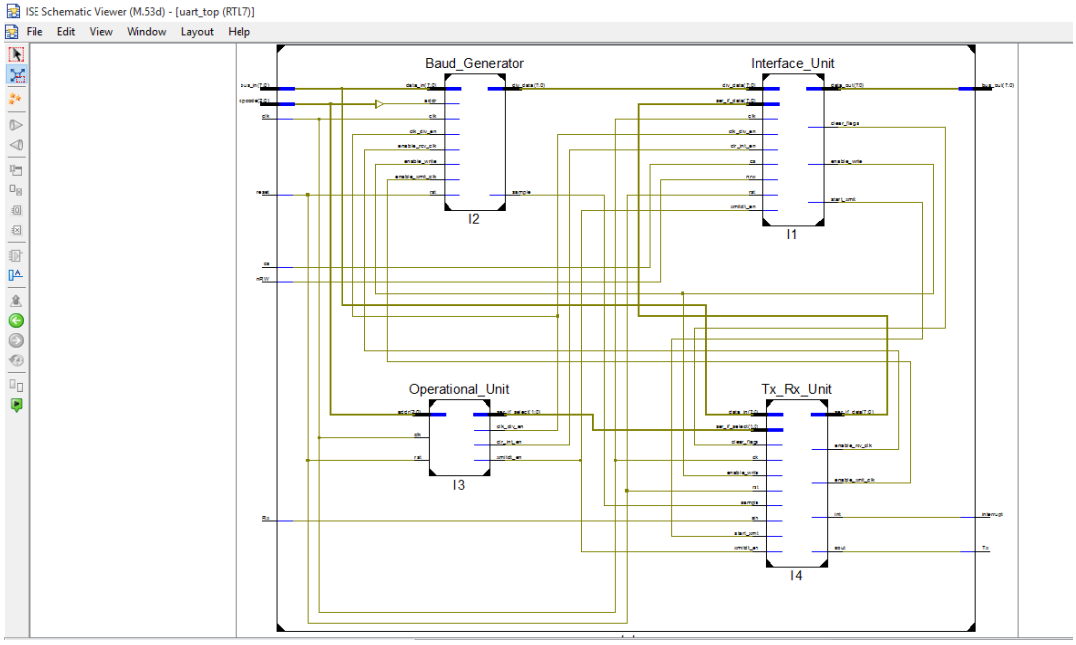

Fig. 5. Internal blocks of UART

The test bench wave forms for the receiver is shown in Fig. 6. Test bench generated signal '11001110' (sequentially) is fed to the Rx line of the UART & simulation shows that the signal is appearing at 'bus_out' port.

The test bench wave forms for the transmitter is shown in Fig. 7. The test bench is feeding 8-bit data '00001010' to 'bus_in' port of UART in parallel. Simulation wave forms shows that the data is appearing at Tx line in accordance with bit frame.

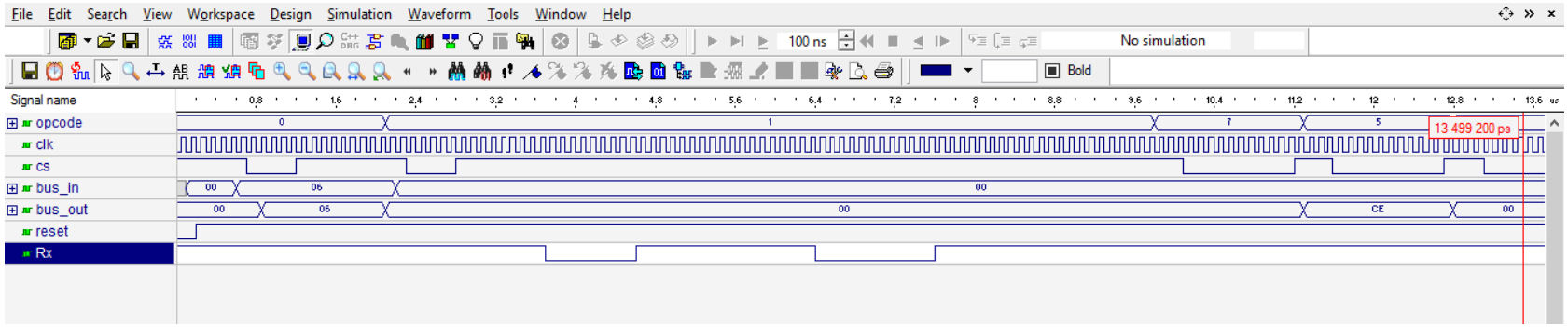

Fig. 6. Simulation waveform result for receive signal

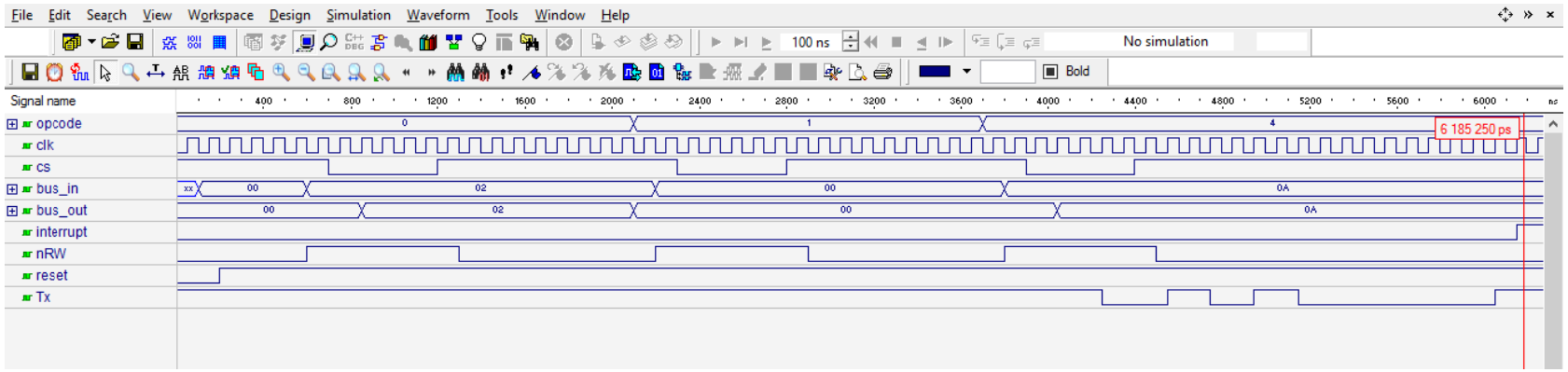

Fig. 7. Simulation waveform result for transmit signal

Conclusion

Therefore we have successfully implemented UART for high speed serial communication in FPGA & the simulation results came satisfactory. The software implementation of UART is platform independent & hence can be used in system of any size.

References

[1] Zou Jie Yang,Jianning. Design *and Realization of UART Controller Bas ed on FPGA*.

[2] Gallo R, Delvai M, Elmenreich W, Steininger A, *Revision and verification of an enhanced UART*, 2004, Proceedings, 2004 IEEE International Workshop on Factory Communication Systems, pp. 315- 318, 22-24, Sept. 2004.

[3] Zhang xue, Xu xiaosu, Zhang guolong. Design of High Speed Serial Communication Based on DSP and FPGA. *Ship Electronic Engineering*, 2009, 29(5), 66-68.

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

[5] James O. Hamblen, Tyson S. Hall, Michael D. Furman. *Rapid prototyping of digital systems*, 2nd ed., springer Publication, 2001.