ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijareeie.com

Vol. 6, Issue 4, April 2017

A System Bus Extension and FPGA Implementation of RS232 to USB Converter Integrate on SOC

Shridhar.S.M[1], Santhosha.B.M[2], Mahammad Anwar[3]

Assistant Professor, EEE Dept. Ballari Institute of Technology and Management, Ballari, Karnataka, India[1]

Assistant Professor, EEE Dept. Ballari Institute of Technology and Management, Ballari, Karnataka, India[2]

Assistant Professor, EEE Dept. Ballari Institute of Technology and Management, Ballari, Karnataka, India[3]

ABSTRACT: The modern communication systems requires the faster reliable and standard protocol for error free communication which can be operated standard for all devices as a common communication node, the paper presents the bidirectional shift converter method for embedded system converter RS232 to Universal Serial Bus into FPGA by using Verilog Hardware Description Language which can be implemented on chip.

KEYWORDS:RS232, USB,FPGA, UART, Shift Converter, FIFO.

I.INTRODUCTION

The common interface connection to an computer with an external word have become easier and more effective by use of Universal Serial Bus (USB) [1-4]. USB connectivity is efficient, reliable and faster communication protocol with reduced cost [1-5]. Using USB more number of peripherals can be connected, which are necessary for the particular system [6]. Now a day's modern communication system advances the use of communication protocol which is compact and user friendly, it's not limited to a one sector, the industries and telecommunication domains are being constantly developing a protocol which has a ready to use Plug and Play option.

In communication system the connection between the master device and slave device can be in two different ways serial port connection and parallel port connection. Mainly RS232 uses serial port communication for long distance transmission and with simple communication protocol [9]. Serial transmission follows two ways of data transmission, synchronous and asynchronous. Synchronous is mainly based on block oriented transmission. Asynchronous is character oriented transmission. Universal Synchronous Receive/ Transmission Integrated circuit (USART) and Universal Asynchronous Receive/ Transmitter Integrated circuit (UART) for serial communication. This serial transmission depends on the hardware implementation on chip of the master or slave device.

A Field Programmable Gate Array (FPGA) is a programmable logic device which is used to implement the larger logic circuits [11]. FPGA is an recommended device to implement simple or a complex system interface or an high integrated system interface based on the system applications. FPGA can be differentiated based on architecture, internal gates, programming methods, robustness of functional unit, variation of pin out, speed, and based on connectivity [12-13]. FPGA are the recommended devices used to implement simple circuit or an complex system implementation, including implementation of state machine notations to satisfy the system requirements. FPGA are reprogrammable device which make them flexible for trial and error - code implementation so that the more reliable and more power full system can be implemented including implementation of power saving options, this is the major concern of embedded designers [14-15].

This paper presented here the design of converter for an RS232 to USB protocol based on sum add implementation of an Shift Converter Block method interfaced with FIFO (First In First Out) and FPGA programmed in Verilog HDL code. Theconnectivity UART with FPGA device, connecting through RS323 port, compared with directly through USB port to receive and transfer and this technique is much easier. The sum add implementation of shift converter block method is to control the bits of data, for transmit and receive from via USB protocol between FIFO design and UART protocol of RS232, here UART protocol takes bytes of data and transmit or receive individual data bit in sequential manner.

II. SURVEY

A) Universal Serial Bus (USB)

Universal Serial Bus the most commonly used protocol in many of computer peripherals nowadays. Depending upon the speed of the interface, it has been classified as:

1. A low-speed (USB 1.0) rate of 1.5 Mbit/s is defined by USB 1.0. It is very similar to full-bandwidth operation except each bit takes 8 times as long to transmit. It is intended primarily to save cost in low-bandwidth human interface devices (HID) such as keyboards, mouse, and joysticks [6].
2. The full-speed (USB 1.1) rate of 12 Mbit/s is the basic USB data rate defined by USB 1.0. All USB hubs can operate at this speed [6].
3. A high-speed (USB 2.0) rate of 480 Mbit/s was introduced in 2001. All hi-speed devices are capable of falling back to full-bandwidth operation if necessary; i.e., they are backward compatible with USB 1.1. Connectors are identical for USB 2.0 and USB 1.x [3].
4. A Super Speed (USB 3.0) rate of 5.0 Gbit/s. The written USB 3.0 specification was released by Intel and partners in August 2008. The first USB 3 controller chips were sampled by NEC May 2009 [6] and products using the 3.0 specification arrived beginning in January 2010 [6]. USB 3.0 connectors are generally backwards compatible, but include new wiring and full duplex operation [3, 6, 7].

The USB Host System basically focuses on the specification features such as descriptions of bus attributes, protocol definition, programming language code interface and other features for each different device type [6]. Any peripheral which needs to be compliant with USB protocol must follow the USB specification otherwise it might result in unpredictable results.
USB work as a master/slave interface such that where USB Host (typically PCs) is the Master while the peripheral devices are the Slaves. The Host is responsible to receive and transmit an I/O request packet to perform the task of scheduling the transactions over the USB (which is composed as a series of USB packets) [6]. The number of register to store the data and memory needed to depend on the types the device connected. This called Endpoint. An Endpoint is uniquely identifiable portion of a USB device as the final stop of communication flow between host and peripheral device (as buffer) [6].

B) Universal Asynchronous Receive Transmit (UART)
Asynchronous method of data transfer is mainly based on byte/ character oriented, in which the data byte has to frame between start and stop bits in order to know when the data is started and end. However the data byte consists of ‘0’ and ‘1’, we need to differentiate the two data’s hence the data byte has to be framed between start and stop bits. One more method is to use parity bit to frame the data byte. Usually start bit is one bit which is logic ‘0’ and stop bit may be one or more bit and it is logic ‘1’ shown in Fig.1.

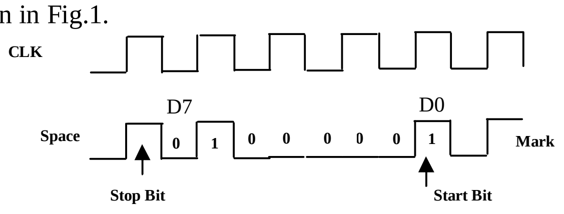

Fig.1Asynchronous method of data transfer example of Framing ASCII ‘A’ (41h)

When there is no transfer, the signal is ‘1’ which is Mark, the ‘0’ is Space. When transmission begins with start bit followed by D0(LSB), the rest of the bits until the D7(MSB), and finally, the one stop bit indicating the end of

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijareeie.com

Vol. 6, Issue 4, April 2017

character. In asynchronous communication methods it can be for 7 or 8 bits wide, in addition to the number of stop bits 1 or 2. To transfer 8-bit data, we have a total of 10-bits including one start and stop bits. Therefore for each 8-bit character there are extra 2-bits, which give 20% overhead.Fig. 2, shows the various component blocks of UART.

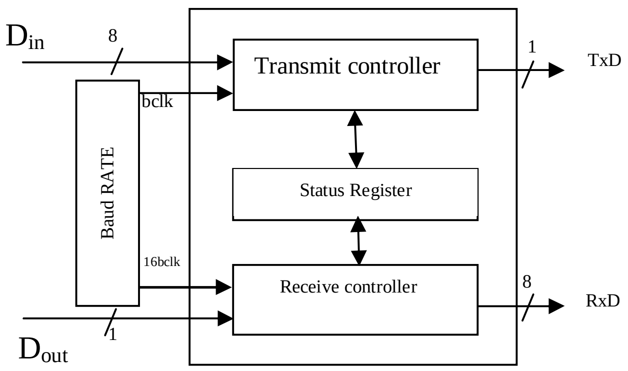

Fig. 2.Various component blocks of UART

The baud rate generator generates a suitable output clock for transmitter and receiver block. The default baud rate to be applying for future work which is 104.16µs needed for bit cell duration. The percentage tolerance occurred in the baud rate generator were also determined to check whether it is acceptable or not in order to minimize lost signal and getting a good transmission. The percentage accuracy is known by using formula as shown in Equation (1) below. Therefore, the equation the percentage tolerance obtained as 0.124%. It is within the acceptable tolerance for transmission signal as in theoretical expectation.

$$\text{Percentage Accuracy} = ((X-Y)/X)\times 100$$

Where, X= Baud rate needed and
Y= Actual Baud rate

C) Shift Converter

A shift register is constructed from Flip-Flops, the data bits can be shifted from left to right or right to left as required. As it an sequential block, the data bits can be loaded in serial or parallel manner constituting the converter as serial to parallel converter and parallel to serial converter. The data rotation modes depend on the type of data packet from transmission when transmitting or receiving the data. The major component blocks of shift converter module are represented in the Fig.3 below.Shift converter contain the different blocks within it such as status register, Parallel to Serial Converter, Current State register, Serial to Parallel converter and Controller which receives the data and responsible for processing within the block as shown in the above Fig.3.

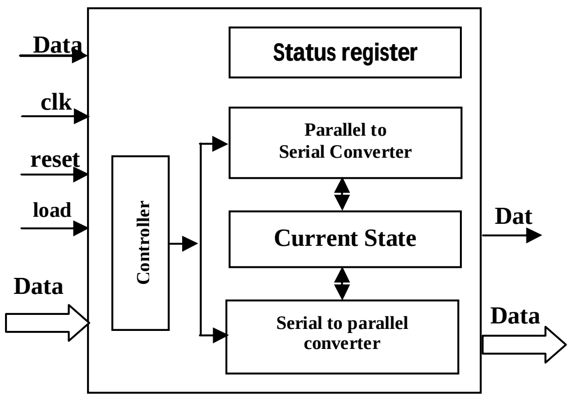

Fig.3. Module of Shift Converter Block

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijareeie.com

Vol. 6, Issue 4, April 2017

If the parallel data enters the controller selects the Parallel in Serial out (PISO) will receive the parallel binary data and converts in serial data in binary form. The register selected as left shift, as the data is shifted from right to left the successive zeros are add onto the vacated position. Adding the respective zeros to from the right end position, then the three conditions is determine the bit shift into the vacate position thus the serial binary data is generated at the output.When the sift converter is selected as Serial in Parallel Out (SIPO) the register is loaded with serial data bit by bit serially through left, the width depends on the data packet, after that the data full indicated by the register, and parallel data will be transferred. And the same repeated with the new data, and continues [16]. The converter module is fed with reset then the internal operation of the converter in idle state and cleared. The load data packet can be detected by the load signal and data in and out can be detected by the respective data signals as illustrated in the above Fig.3.

D) First In First Out (FIFO)
A First in First out (FIFO) is a special type of buffer. In FIFO structure the data written into the buffer first comes out of it first. There are other kinds of buffers like the LIFO (last in first out), often called a stack memory, and the shared memory. The choice of buffer architecture depends on the application to be solved [15, 19, 20].FIFOs can be implemented with software or hardware. The designing software or hardware depends on the application and the features desired. When requirements change, a software FIFO easily can be adapted to them by modifying its program, while a hardware FIFO may demand a new board layout. Software is more flexible than hardware. The advantage of the hardware FIFOs shows in their speed. A data rate of 3.6 gigabits per second is specified for a Texas Instruments (TITM) SN74ABT7819 FIFO [21].Every memory in which the data word that is written in first also comes out first when the memory is read is a first-in first-out memory [21]. There are three kinds of FIFO:

SHIFT REGISTER – FIFO with an invariable number of stored data words and, thus, the necessary synchronism between the read and the write operations because a data word must be read every time one is written. The proposed design is based on shift register technique. FIFO buffer memory between the receiver shift register and the host system interface. This allows the host processor even more time to handle an interrupt from the UART and prevents loss of received data at high rates [21].

EXCLUSIVE READ/WRITE FIFO – FIFO with a variable number of stored data words and, because of the internal structure, the necessary synchronism between the read and the write operations [21].
CONCURRENT READ/WRITE FIFO – FIFO with a variable number of stored data words and possible asynchronism between the read and the write operation [21].

In this paper the Asynchronous FIFO is used, and is mainly used when function read are slower than the write functions. It includes the with two clocks for writing and reading operations that are asynchronous to each other. This means the data reading from FIFO is achieved by using another clock domain and data write to the FIFO is from other clock domain. FIFO can be parallel or series and to avoid data loss the two types of signals are used in corresponding with clock, FIFO Full Flag and FIFO Empty Flag. Generally, two types of counters are used as a FIFO pointers. Binary and Gray counter. In this paper the binary counter is used because it has an advantage read write with clock and synchronization. The care is taken to avoid the metastable state between the signal transitions.

III.DESIGN OF SYSTEM

The AMBA AHB bus protocol is designed to be used with a central multiplexer interconnection scheme. Using this scheme all bus masters drive out the address and control signals indicating the transfer they wish to perform and the arbiter determines which master has its address and control signals routed to all of the slaves. A central decoder is also required to control the read data and response signal multiplexor, which selects the appropriate signals from the slave that is involved in the transfer [18]. This design of a converter for an RS232 protocol to Universal Serial Bus (USB) protocol using Shift Converter Block technique on FPGA using the FIFO. The FIFO is designed as bi-directional and hence would be used to convert the RS232 protocol to the USB protocol and also to convert the USB protocol to the RS232 protocol shown in Fig.4.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijareeie.com

Vol. 6, Issue 4, April 2017

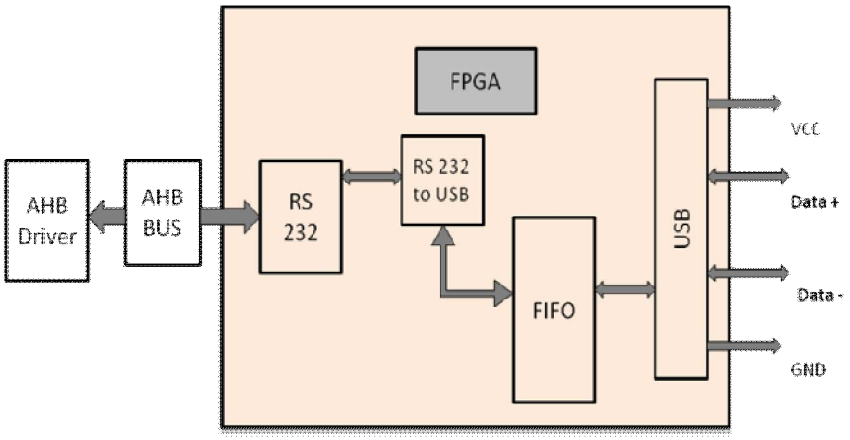

Fig.4. Block Diagram of Proposed System,the FIFO is designed as bi-directional, used to convert the RS232 protocol to the USB protocol and also to convert the USB protocol to the RS232 protocol

IV. IMPLEMENTATION

A)Design Implementation

The implementation of bidirectional shift converter block technique in a converter for UART to transmit and receive data to USB Protocol Engine via FIFO was designed using FPGA device as illustrate in Fig.5. Firstly, based on Fig. 5. below, the behaviour of these entire blocks are characterized using Verilog HDL language. In this programming derivation stage, design specification such as input and output elements was determined depending on the functionality of each created module.

Then, each code for the module was compiled in order to check the syntax of the design and synthesized to update or convert the designed from Verilog HDL code into a symbol logic block. After this stage, the simulations waveforms of each module had presented to ensure the designed output waveform matched with theoretical expectation. These simulations had done using ModelSim. Next, all these blocks are connected and combined together into one as a main symbol block logic. Lastly, it will be compiled and analysed the simulation output waveform again via the same tool.

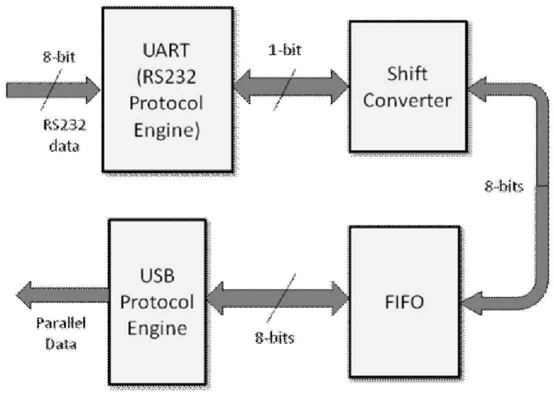

Fig.5. Design Implementation

The first part is designing UART of RS232 Protocol Engine to transmit and received signal from bidirectional Shift Converter. It considered with three main parts, which are Transmit Controller, Received Controller and Current State Register. In the following sections the UART controller, transmitter and receiver controller are discussed.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijareeie.com

Vol. 6, Issue 4, April 2017

B) UART Controller Protocol Engine

It consists of controller, which has the update from the status register and the current state. This in turn enables the parallel in serial out or the serial in parallel out depending upon the control logic and the dataflow shown in Fig 6.

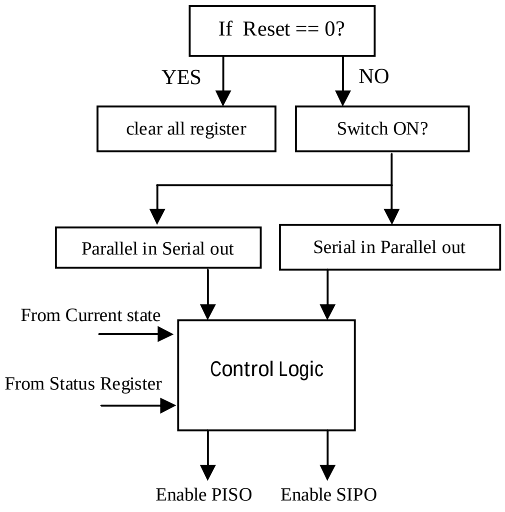

Fig.6. UART Controller Protocol Engine Design

The below figure 6(a) shows the RTL view of the controller

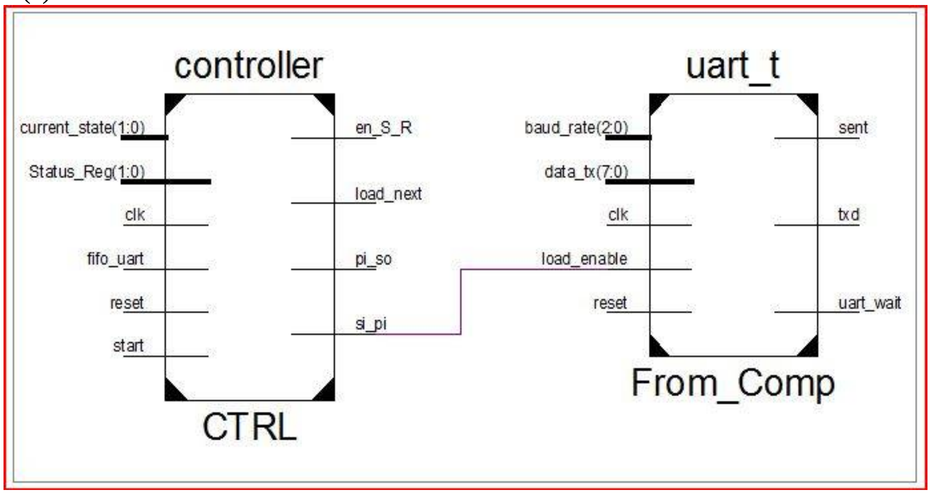

Fig 6.a.RTL view of the controller

Control signals for controller are given as :

Current state(1:0)-It is 2 bit register which indicates the mode of operation weather it is in serial to parallel or parallel to serial mode, Status Register(1:0)- It is 2 bit resister which indicates the status of transmission completion. Fifo_uart- It's a one bit signal coming from uart, which is to be converted to parallel and store it in fifo. Pi_so- It's switch when one parallel to serial. Si_pi- It's switch when one serial to parallel. START- It's a enable signal for the whole module. Load_next- It is used to load the 8 bit data from fifo to rs232 buffer. En_S_R- It enables the shift convertor. Clk- Global clk. Reset- Global reset.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijareeie.com

Vol. 6, Issue 4, April 2017

Transmitter and Receiver Controller: In this Transmit Controller part, it consists of two major components, which are Transmit Buffer and Shift Register. Transmit Buffer is a function to load and transmit data from local CPU into Load Register. Then, Shift Register will load and shift the data from Transmit Buffer before send it through TXD output pin one by one bit to the shift converter. While, Receive Controller also have two main components, which are Receive Shift Register that functions to receive the data from the shift converter from RXD pin one by one bit and Receive Buffer to load and accept the data from Shift Register to local PC read. Based on this, UART had a special function register to analyze and control the register to transmit or receive data. The below figure 6(b) shows the RTL view of UART transmitter.

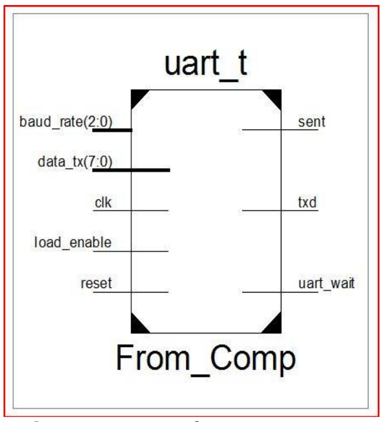

Fig 6.b. RTL view of UART Transmitter

Control signals for UART Transmitter given as:
Baud_rate(2:0)- This is used to switch the baud rate of the UART. Data_tx(7:0)- It’s a 8bit buffer; date to be transmitted is stored on “data_tx” register. Clk-Global clock. Reset-global reset. TXD- Serial data is transmitted on txd pin with start and stop bits. Load enable- Once the data is ready in the buffer, load enable signal is toggled, so that this module can start transmitting. Sent- This data is toggled once when stop bit is transmitted. Uart_wait- UART wait signal is be asserted once the start bit is transmitted and will be pulled low when stop bit is transmitted. This signal will make sure that on new date is over written till all the previous 8 bits are transmitted

The control signals for receiver is given as:
Baud_rate(2:0)- This is used to switch the baud rate of the UART. Data_rx(7:0)- It’s a 8bit buffer; received 8 bit data is stored in it. Clk-Global clock. Reset-global reset. RXD- Serial data is received on rxd pin. Done- Once the stop bit become “done”: signal is togged once, so that the successive module can copy the received data.

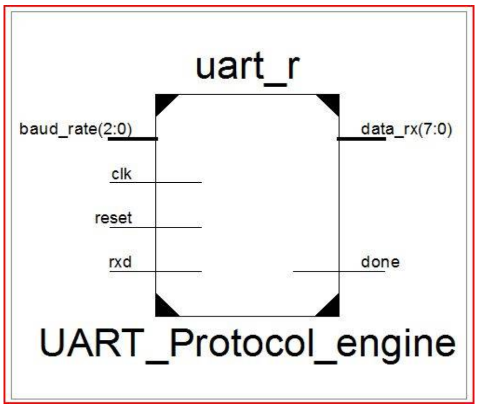

Fig 6(c). RTL view of UART Receiver

Shift Register: Any register on SoC should be molded as shown below figure 6(d). It should contain a clock and reset as global signals, Chip Select (C_S) which is the highest priority signal after reset signal, address bus (addr),Write data bus (Wdata), Read data bus (Rdata), Write/read signal (RW) which will diced weather to write or read.The register may have n-number of internal register (reg0, reg1, reg2, reg3) which may be made output or input, also the width of the internal register need not to be same size as in memory module. The internal registers are bit addressable but in memory it is always byte andmultiples of byte addressable. This provides the flexibility to design. Here we will be using 2 such register one is for status register and the other is the shift register.

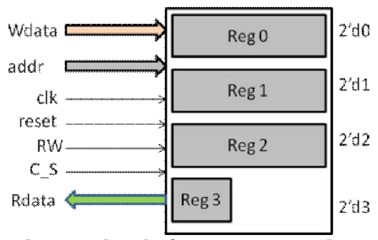

Fig 6(d) Simple Shift Register used in SoC

C) Bidirectional Shift Converter

The second part is designing the bidirectional shift converter that has two functions as illustrated in Figure.7 below. First function is to receive data from RS232 Protocol Engine one by one bit using port “datain_tx” before load and shift bit into the special internal register to convert it into eight bits data width. After it had done, the data were transmits to FIFO using port “dataout_tx” to read and write the data. Then, FIFO will transmit in byte data width to the USB protocol engine.

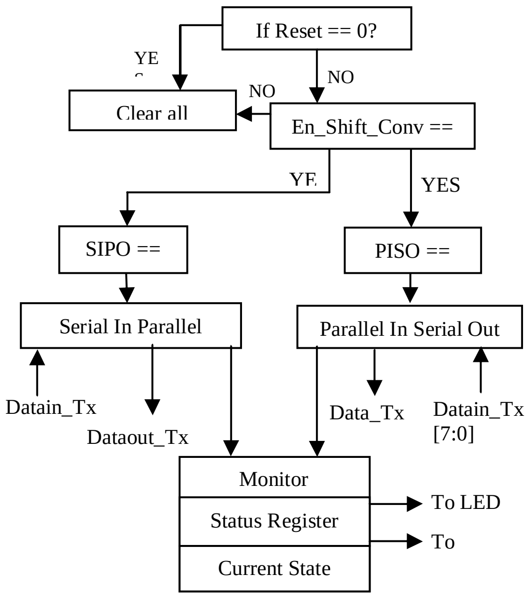

Fig .7. Bidirectional Shift Converter

The control signal for Bidirectional Shift Converter given as:
Data_bus(7:0)- 8-bit data bus(7:0). En_S_R- Enable Shift Convertor.Pi_So- Do parallel in serial out.Si_Po- Do serial in parallel out.Current_status (1:0)- 2 bit register shows the current status. Data_bus_out(7:0)- Shift convertor to FIFO. Status_register(1:0)- 2 bit register shows the status. Write_fifo- Write enable to fifo. Load- Fifo to shift convertor enable signal.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijareeie.com

Vol. 6, Issue 4, April 2017

The below figure7(a) illustrates the RTL view of bidirectional shift converter and table.4 control signals of bidirectional shift converter.

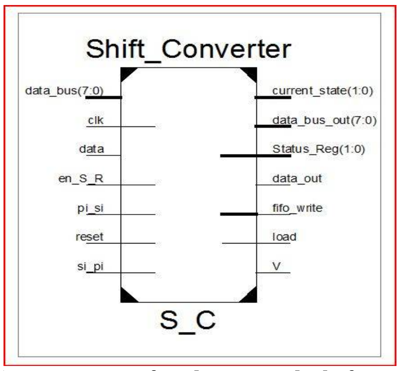

Fig 7(a) RTL view of Bidirectional Shift Converter

FIFO (First In First Out): The second functions are to receive data in the parallel bytes of data width to FIFO. Afterwards, after it interfaces operation in FIFO, byte data bus was sent to the bidirectional Shift Converter using port “data_in” before load and shift bit into a special internal register that function to convert from eight-bit data width into one-bit data width. Then the data will be transmitted to RS232 Protocol Engine using port “data_ rx." In order to check and analyze the bit operation “cs_tx” and “cs_rx” stand for current state for transmit data to UART (RS232protocol engine) and four transmit data to FIFO are created, respectively.

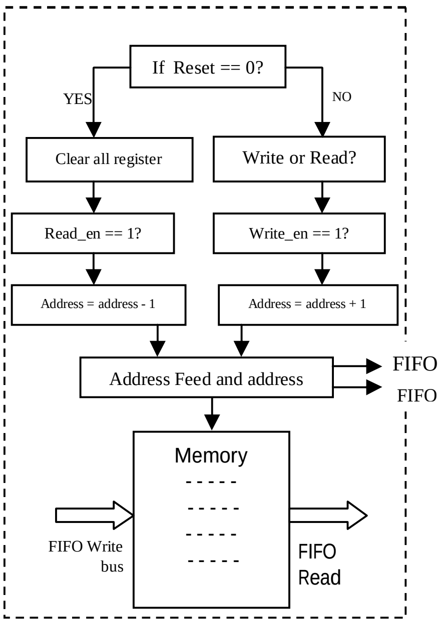

Fig.7(b) Internal architecture of FIFO

The operation of FIFO is it will have internal write address and internal read address also it will generate “FIFO Full” And “FIFO Empty” status signals, which will help to start or stop the read or write operations. So it writes the data only when “write en” is high and it will write the through “FIFO Write” bus. Once the write operation is done the internal write address will be incremented by once. In the same faction to read the data from FIFO the “Read en” should maid high and read from “FIFO Read data” bus. Once the read operation is done internal read address will be incremented. If the write address is at “0” the “FIFO Empty” signal will be high, if the write address is at “>0” the “FIFO Empty”

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijareeie.com

Vol. 6, Issue 4, April 2017

signals will be low, In the same manner write address reaches maximum then “FIFO Full” signal will go high or else “FIFO Full” will be low. Internally the architecture of FIFO would be like as shown in figure 7(b).Below figure 7(c) shows the RTL view of the implemented FIFO.

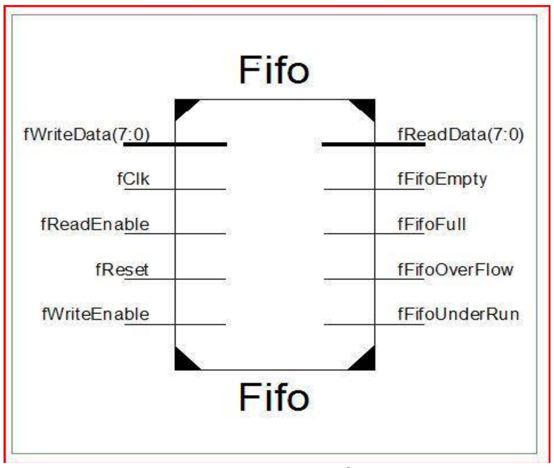

Fig.7(c) RTL view of FIFO

Control signals for FIFO given as:
Fwritedata(7:0)- Write data bus to FIFO. Freaddata(7:0)- Read data bus to FIFO. fReadenable- Read enable signal. Fwrite enable- Write enable signal. Fifo_full- Indicates when the FIFO is full. Fifo_empty- Indicates when the FIFO is empty. Fclk- Global clk.Freset-global reset.

V. FLOWCHART

The below figure 8. shows the flowchart of the controller, shift converter block and transmission of the serial to paralleland parallel to serial data.

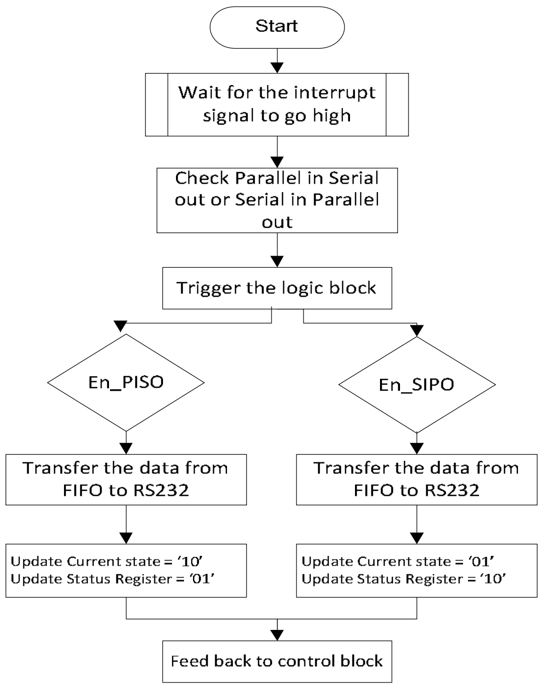
Start
Wait for the interrupt signal to go high
Check Parallel in Serial out or Serial in Parallel out
Trigger the logic block
En_PISO
En_SIPO
Transfer the data from FIFO to RS232
Transfer the data from FIFO to RS232
Update Current state = '10' Update Status Register = '01'
Update Current state = '01' Update Status Register = '10'
Feed back to control block

Fig 8. Flowchart of Controller, Shift Converter Block and Transmission of the Serial-Parallel and Parallel-Serial Data.

VI. RESULTS AND DISCUSSIONS

The architecture is model in Verilog HDL. The functionality and characteristic all designs was compiled and synthesized using Xilinx tool. Then the results of the simulation in ModelSim-PE are presented below.
a) Simulation Waveforms: The simulation is done using Modelsim, the snapshot of simulated waveforms are included in this section. The below figure 9.1 shows the FIFO depth and FIFO write operation. The FIFO write starts with the 'start' and 'fifo_uart' enable signal, and the 'data_in' signal which indicates the data presence on the data bus. After the 'data_rx' received the 'done' signal indicates the data presence. The serial data is converted to parallel data by shift converter with the activation of 'si_po' signal, followed by the 'load_next' signal after the conversion. The FIFO memory designed of 32x8, but for convenience FIFO depth of 8x8 is considered for illustration. The FIFO status is indicated by high to low pulse 'fFifoEmpty' signal, therein FIFO is written with the data to the consecutive locations as shown in below fig.9.1

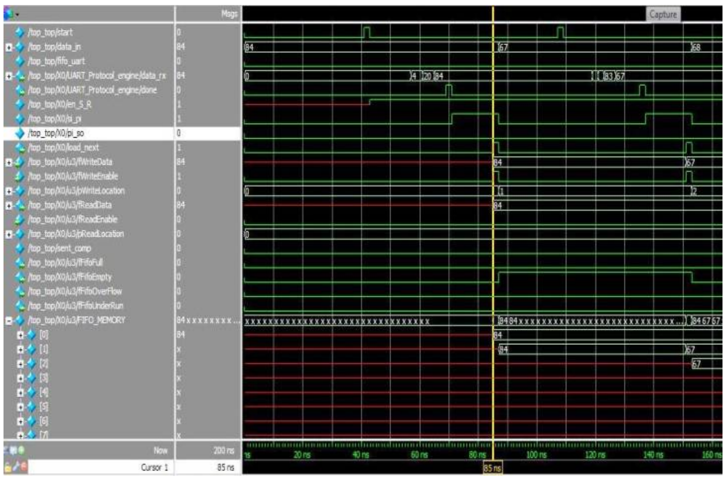
/top_top/start
/top_top/data_in
/top_top/fifo_uart
/top_top/X0/UART_Protocol_engine/data_rx
/top_top/X0/UART_Protocol_engine/done
/top_top/X0/en_S_R
/top_top/X0/si_pi
/top_top/X0/pi_so
/top_top/X0/load_next
/top_top/X0/u3/fWriteData
/top_top/X0/u3/fWriteEnable
/top_top/X0/u3/pWriteLocation
/top_top/X0/u3/fReadData
/top_top/X0/u3/fReadEnable
/top_top/X0/u3/pReadLocation
/top_top/sent_comp
/top_top/X0/u3/fFifoFull
/top_top/X0/u3/fFifoEmpty
/top_top/X0/u3/fFifoOverFlow
/top_top/X0/u3/fFifoUnderRun
/top_top/X0/u3/FIFO_MEMORY
Now 200 ns
Cursor 1 85 ns

Fig.9.1 FIFO Write Operation with the Shift Converter, Converting SIPO and Stored on FIFO Memorywith the data to the consecutive locations (snapshot)

When the data has to be read from the bus the 'data_in' signal indicates the presence of the data, and the 'done' pulse is raised. The parallel data from the FIFO memory has to convert to the serial which is done by the shift converter, by the 'pi_so' enable shown in below figure 9.2 the FIFO depth of 8x8 and FIFO read operation. The read operation starts with 'start' enable followed by the 'fifo_uart' activation signals.

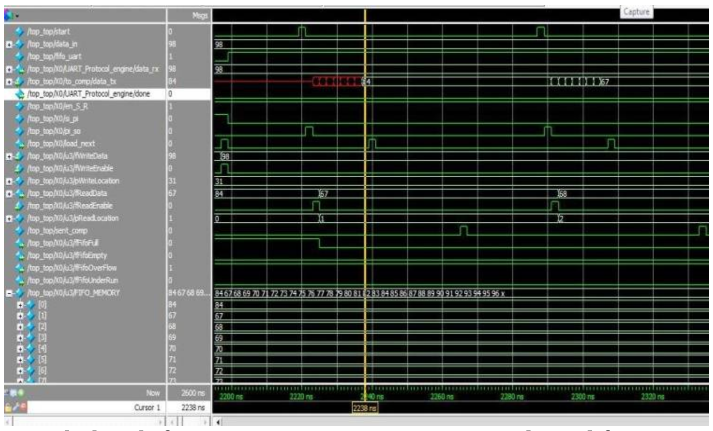
/top_top/start
/top_top/data_in
/top_top/fifo_uart
/top_top/X0/UART_Protocol_engine/data_rx
/top_top/X0/to_comp/data_tx
/top_top/X0/UART_Protocol_engine/done
/top_top/X0/en_S_R
/top_top/X0/si_pi
/top_top/X0/pi_so
/top_top/X0/load_next
/top_top/X0/u3/fWriteData
/top_top/X0/u3/fWriteEnable
/top_top/X0/u3/pWriteLocation
/top_top/X0/u3/fReadData
/top_top/X0/u3/fReadEnable
/top_top/X0/u3/pReadLocation
/top_top/sent_comp
/top_top/X0/u3/fFifoFull
/top_top/X0/u3/fFifoEmpty
/top_top/X0/u3/fFifoOverFlow
/top_top/X0/u3/fFifoUnderRun
/top_top/X0/u3/FIFO_MEMORY
Now 2600 ns
Cursor 1 2238 ns

Fig 9.2 FIFO Read Operation with the Shift Converter, Converting PISO and Read from FIFO Memory,the FIFO depth of 8x8 and FIFO read operation (snapshot)

The parallel data '00010001' which is converted to the serial upon the activation of 'en' signal, between the 'valid' portions of the conversion, similarly the parallel data's '01000101' , 0111011' and '00110010'.From fig 9.2 upon the Parallel to serial conversion, based on the status of the 'fFifoFull' and 'fFifoEmpty' the FIFO memory is read, the 'sent_comp' signal is activated after each data is sent.Likewise the FIFO memory can be read and the locations of the FIFO memory are overwritten by the new entries of the data.

VII. CONCLUSION

In this work, a novel technique to convert the RS232 protocol to USB has been presented. It consists of designing a shift converter and a FIFO. The shift converter will take the serial data from RS232 and convert to the parallel data. This data would be stored on the FIFO which can be retrieved by the USB. In the same fashion, the USB will provide the parallel data to the FIFO, which would be transmitted to the shift converter. The shift converter would be convert this parallel data to serial data and provide to the RS232 peripheral device. A FSM has been designed to control the data flow for this system. This full design has been coded using the Verilog HDL. The functionality of the design has been verified using the Mentor Graphics Modelsim simulator. A separate test bench has been coded to verify the design. All the simulations result have been presented here and it is shown that design is meeting its requirement. Finally, the design has been ported on to a FPGA.

REFERENCES

[1] NurulFatihahJusoh, Muhammad AdibHaron, FuziahSulaiman, "An FPGA Implementation of Shift Converter Block Technique on FIFO for RS232 to Universal Serial Bus Converter",IEEE Control and System Graduate Research Colloquium (ICSGRC 2012), pp 219-234,16-17 July 2012.
[2] Vijaya. V, Valupadasu. R, Chunduri. B.R, Rekha, C.K, Sreedevi. B, "FPGA Implementation of RS232 to Universal Serial Bus Converter", IEEE Symposium on Computers & Informatics (ISCI), pp 237-242, 2011.
[3] Ana Luiza de Almeida Pereira Zuquim, C.J.N. Coelho ; A.O. Fernandes ; M.P. de Oliveira ; A.I. Tavares,"An embedded converter from RS232 to Universal Serial Bus", Integrated Circuits and Systems Design, pp 91, 14th Symposium on 15 Sept. 2001.
[4] Jan Axelson, "USB Complete, Everything you need to develop custom USB Peripherals", Penram Intl. Publishing (India), 1999.
[5] Sunggu Lee, "Advanced Digital Logic Design Using Verilog, State Machines and Synthesis for FPGAs", in Proc. 1st Edition, pp 46-107, ISBN 0-534-55161-0, 2006.
[6] www.usb.org
[7] Mindshare, Inc., D. Anderson, "Universal Serial Bus System Architecture", Addison-Wesley Developers Press, U.S., 1997.
[8] R.Wyrmas,W. Zhang, M.J. Miller, and R.Anjaria, "Multiple Access Option for Multimedia Wireless System", in Proc. 3rd Workshop on Third Generation, pp 289-294, Apr.1992.
[9] L. K. Hu and Q.CH. Wang, "UART-based Reliable Communication and performance Analysis", Computer Engineering, Vol 32 No. 10, pp15-21, May 2006.
[10] Cavanagh, Joseph J.F, "Digital Design and Verilog HDL Fundamentals", in Proc. 1st Edition, pp 566- 713, ISBN 978-1-4200-7415-4, 2008.
[11] Brown, Stephen D., "Fundamental of digital logic with Verilog design", in Proc. 2nd Edition, pp77-166, ISBN 978-0-07-338033-9, 2008.
[12] Chan PK, Mourad S., "Digital Design Using Field Programmable Gate Arrays", Upper Saddle River, NJ: Prentice-Hall, 1995.
[13] Oldfield JV, Dorf RC., "Field programmable Gate Arrays", New York: Wiley Inter-science,1995.
[14] Ciletti, Michael D, "Advance Digital System with the Verilog HDL", in Proc. 1st Edition, pp103-132, ISBN 0-13-089161-4.
[15] Shouqian, Y., Y. Lili, et al. (2007). "Implementation of a Multichannel UART Controller Based on FIFO Technique and FPGA". 2nd IEEE Conference on Industrial Electronics and Applications, 2007. ICIEA 2007.
[16] "Spartan-6 Family Overview., DS160 (v2.0)". October 25, 2011. www.xilinx.com.
[17] Cavanagh, Joseph J.F, "Digital Design and Verilog HDL Fundamental",in Proc.1stEdition, pp620-622, 2008, ISBN 978-1- 4200-7415-4.
[18] "AMBA™ Specification (Rev 2.0)", 13th May 1999 A First release.
[19] Clifford E. Cummings, Sunburst Design, Inc., "Simulation and Synthesis Techniques for Asynchronous FIFO Design", SNUG San Jose 2002.
[20] Sanjeev Kumar, Department of Electronics and Communication, RKDF Institute of Science and Technology, Bhopal., "Design and Implementation of UART using FIFO for Serial Communication"., (ISSN : 2277-1581), Volume No.2, Issue No.7, pp : 737-740 1 July 2013.