An FPGA Implementation of Shift Converter Block Technique on FIFO for UART

Nurul Fatihah Jusoh, Azlina Ibrahim, Muhamad Adib Haron and Fuziah Sulaiman

Faculty of Electrical Engineering
Universiti Teknologi MARA, 40450 Shah Alam
Selangor, Malaysia
E-mail: Isku_nurul03@yahoo.co.uk

***Abstract*—To meet the standard modern system wireless communication demands, the paper represents the implementation of bidirectional shift converter technique with FIFO circuit block and UART (Universal Asynchronous Receiver Transmitter) circuit block through FPGA device using Verilog HDL language to be applied in embedded system converter RS232 to USB (Universal Serial Bus). Utilizing the ModelSim-Altera, RTL model of the shift converter was developed and synthesized then stimulated using TimeQuest Timing Analyzer to observe its functionality.**

Keywords- UART; Shift Converter; FIFO; FPGA

I. INTRODUCTION

For the past few years, there has been an exponential growth for invention and innovation in the development wireless network communication. The communication services in such networks can be high-speed and low-speed data, video, and many others with different performance and traffic requirements [1]. In certain complex systems, communications between the master controller and slaver controllers can implement in serial or parallel port. However, serial communication extensively used because it used simple structure [2] and have long transmission distance [2]. Hence, in several controller systems, UART (Universal Asynchronous Receiver/Transmitter) is an integrated circuit of serial communication protocol that used widely. It enables to control the conversion between serial and parallel data as shown in Fig. 1 below. Serial communication reduces the distortion of a signal, therefore, makes data transfer between two systems separated in great distance possible [3]. It takes bytes of data and transmits the individual bits in a sequential fashion.

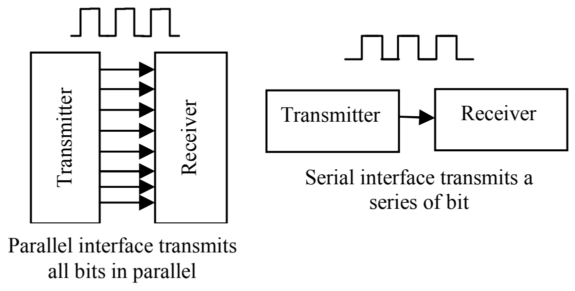
Transmitter
Receiver
Parallel interface transmits all bits in parallel
Transmitter
Receiver
Serial interface transmits a series of bit

Fig.1. General pattern of transmission

There is two types primary form of serial transmission, which is synchronous and asynchronous depending on modes that are support by the hardware. Asynchronous transmission can allow the data to be transmitted without the sender having to send a clock signal to the receiver compared with synchronous transmission. Instead that, it needs some advance parameter and special bit added to each word as synchronization for sending and receiving.

FPGA (Field Programmable Gate Array) has become one of the prevailing implementation technologies for complex digital systems in both prototyping and operational systems [4]. It is used extensively and became a popular prototype platform while develop customer electronic product such as designing of a digital circuit. Very often they are used as a sort of core processor for existing processors, for example, as the PC-bus add-on boards, or the boards for the other bus types [5, 6]. In addition, FPGAs are always became one of the recommended devices used to implement simple interface circuit or complex state machine to satisfy a different system requirement. It is because it can improve the system integration [2], reliability [2] and reduce power consumption [2]. It also contains a programmable logic components as known as "logic block" and wired together in a special hierarchy of configurable interconnect. Besides that, it can configure into a logic blocks to perform complex combinational functions in simple logic gates such as AND and XOR.

In this paper, FPGA- EPC Class-1 Generation-2 protocol produced by ALTERA using Verilog HDL language and also the implementations of shift converter block technique were formed to control package bits of data whether for transmit and received operands via FIFO technique design and UART design. It is because, as knowledge UART takes bytes of data and transmits/receive the data individual bit in sequential pattern. However, the FIFO designs will transmit/receive in same taken bytes of data. Therefore, in order to make data for transmit/receive from UART to FIFO or from FIFO to UART synchronized with data packet, the implementation of shift converter technique was used. It also gives a benefit in terms of control, synthesis and analyzed the packet of data. Then, this architecture combination of all the design can be implemented practical to create converter RS232 protocol engine to USB engine as future work.

II. BACKGROUND OF RESEARCH

A. UART (Universal Asynchronous Receiver Transmitter)

A frame in an asynchronous serial protocol for RS232 protocol engine is a non-divisible packet of a bit that embeds one start bit, seven or eight data bits, one parity bits (depend on the width data either even parity or odd parity), and stop bits. However, most of the bits in a frame are self-explanatory. Fig. 2 below, showed the component block in UART. Referring to this Fig. 2 baud rate generator is used to generate a suitable output clock for transmitter and receiver module. Bit cell duration applied for baud rate needed is 104.16μs. It is as default baud rate to be applied for future work. The percentage tolerance occurred in the baud rate generator were also determined to check whether it is acceptable in order to minimize lost signal and getting a good transmission. It is done by considered applied value master clock and baud rate needed by using a specific formula for getting percentage accuracy as shown in Equation (1) below. Therefore, using the given equation below the percentage tolerance occurs is 0.124%. It is within the acceptable tolerance for transmission signal as in theoretical expectation.

$$\text{Percentage Accuracy} = \frac{(X - Y)}{X} \times 100 \qquad (1)$$

Where,

X = Baud rate needed
Y = Actual baud rate

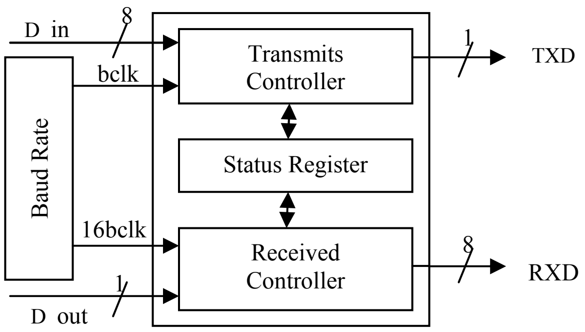

Fig. 2. Structure of UART block

B. Shift Converter

A shift register is well-known as the sequential logic module constructed from flip-flops that manipulates the bit positions of binary data by shifting data bits to the left or right depend on the instruction setting in the module. There are two methods involve in this shift converter, which is serial to parallel converter bit and parallel to serial converter bits. The rotation modes depend on the type of packet data from current transmission when transmitting or receiving process. In this paper, there are five majors components implement in designing shift converters, which are controller, status register, parallel to serial converter bit, current state, and serial to parallel converter bit as illustrated in Fig. 3 below.

When, the shift converter in parallel-in, serial-out (PISO) register it will accept binary input data onto parallel and generates binary output data in a serial form. The register in this module is set for left-shift device. Then, three conditions will determine the value of the bits shifted onto the vacated position on the right. After that, zeroes will fill that position upon the operand shifting. The serial output data is generated from the output of the internal load register. However, when the shift converter is in serial-in, parallel-out (SIPO) register, then another typical synchronous iterative network will be identical as a cell. Data will enter the register form from the right and shift serially to the left through all the stages involved inside depend on the width of packet data. 1-bit positions will do operand every clock pulse. After that, the register will be in a state fully loaded, and the bits are transferred to the destination. Lastly, a new packet byte data will replace the position and continues the operands. This SIPOs operation is a very useful application to generate the sequence of non-overlapping pulses for system timing [7].

All both register will do operation shift at each active clock transition. A converter is implementing with synchronous reset. It means that when the reset is detected, all the operation and register will be in the state idle and clear. Otherwise, it will operate as instruction given in the module.

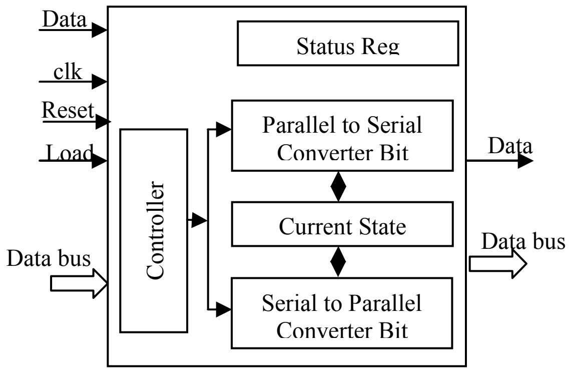

Fig. 3. Structure of shift converter block

C. Asynchronous FIFO

An Asynchronous FIFO refers to FIFO design that used to buffer data in a digital system. FIFOs are always used for data cache, storing different of frequency or phase of asynchronous signal [2]. The requirement of FIFO will arise when they read are slower than the writes. It interfaces with two clocks domains for writing and reading operations that asynchronous to each other. It means that, data is written into the FIFO from one clock domain, and it is read from another clock domain. Because of this, it required a memory architecture wherein two port memories are available one for the write input operation and other one for read input operation.

Moreover, FIFOs can be used to complete parallel or series data. Generally, to facilitate error-free operations of FIFO, there are two types of signal generated corresponding with the clock which is FIFO full flag and FIFO empty flag. When the system is in write operand, and it is a full flag, write system will not accept and write data to the read system until the system is in an empty flag again. It is to prevent the data from lost. Commonly, there are two types of counter who used as FIFOs pointers, which are binary counter and Gray counter. No matter under what circumstance, these two methods have merits and demerit. The decisive factor in choosing a right counter design as a pointer must be referring on the pitfalls between the read and write clock domain and its synchronization advantages. For that reason, the counter who chooses as FIFOs pointer in this paper is in binary counter. This counter is easy to design and implement in addressing FIFOs. To avoid metastability between the transactions of a binary bit of the pointer the widths of counter bits are set suitable with the addressing location of memory address and produce some holding register. This holding register can synchronize signal.

III. Project plan/Design Methodology

The implementation of bidirectional shift converter block technique was created to become as a converter bit data for UART to transmit and receive data via FIFO was designed using EP2C35F672C6 FPGA device as illustrate in Fig. 4. Firstly, based on Fig. 4 below, the behaviors 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 of a 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 simulation waveform of each module is presented to ensure the designed output waveform is matched with the theoretical expectation. The simulation was done using ModelSim-Altera. Next, all these blocks are connected and combined together into one as a main symbol block logic. Lastly, it will compile and analyze the simulation output waveform again via the same tool.

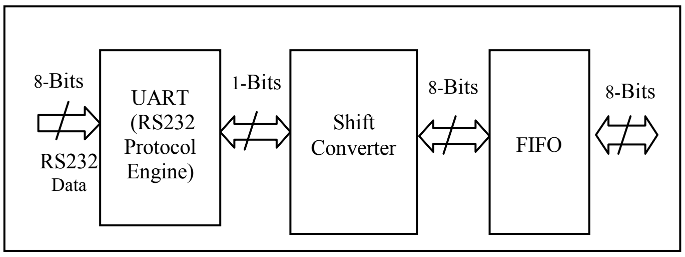

Fig. 4. Block diagram of the proposed digital transmitter via FIFO

A. *Symbol Logic Block*

The first part is designing UART of the RS232 protocol engine to transmit and received signal from bidirectional Shift Converter. It is considered with three main parts, which are Transmit Controller, Received Controller and Current State Register. In this Transmit Controller part, it consisted 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 functioned 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 has a special function register to analyze and control the register to transmit or receive data.

The second part is designing the bidirectional shift converter that has two functions as illustrated in Fig. 5 below. First function is to receive data from RS232 Protocol Engine one by one bit using port “datain_tx” before it loading and shifting a bit into a special internal register to convert it into eight-bit data widths. After it is done it will transmit the data using port “dataout_tx” to FIFO read and write the data. Second function is to receive data from FIFO eight data bus using port “data_in” before a 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 FIFO are created respectively.

Besides, from this Fig. 5 the structure of a bidirectional shift converter is named as “conv_bit” in symbol logic block. This symbol logic block was created using Verilog HDL from two directional transmit and receive sub-module converter. Then, it was compiled, synthesized and stimulated before download together into this symbol logic block onto FPGA to verify it functionality. It consists of four input ports, including the ‘clk’, 1-bit ‘datain_tx’, ‘reset’, ‘load’, ‘store’ and ‘data_in’ with 8-bit length package data. These 2 input ports which are ‘datain_tx’ were connected to UART while ‘data_in’ was connected to FIFO. The output ports are labeled as ‘cs_tx’ and ‘cs_rx ’with 4 four bus widths each act as a current state for transmit signal to FIFO and UART respectively. Furthermore, data out ports, which are ‘data_rx’ as output port connected to UART and 8-bit widths bus “dataout_tx” as output port connected to FIFO.

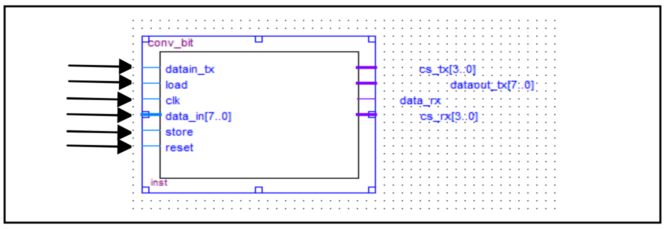

Fig. 5. The symbol block logic structure of shift converter

In addition, the methodology in this designing and an operation of a shift converter were also summarized it into two phases. For the first step, the converter for transmitting package data bus from UART (RS232 Engine Protocol) to the FIFO was created then followed with second step, which are receiving package data bus from FIFO to Universal Asynchronous Receiver and Transmitter (RS232 Protocol Engine) as illustrated in Fig. 6 and Fig. 7 respectively.

Therefore, base on Fig. 6 below, from the starting point, if “reset” is at logic ‘1’, then all the register will be cleared into ‘0’ including “load” and output “dataout_tx” bit logic. Otherwise, if “load” is at logic ‘1’ and “reset” is ‘0’ the “shift” process will operate using count operation technique once it detected. Finally, outputs “dataout_tx” will transmit data to FIFO in eight widths package data bus. In order to check the bit’s operation to FIFO output register pin “cs_tx” stand for current state inserted. The Verilog HDL code will be compiled and synthesized using Modelsim-Altera device in order to obtain and analyzed the result.

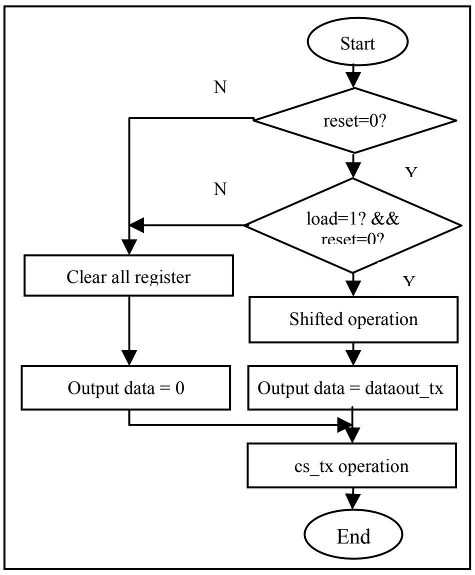

Fig. 6. Flow chart shift converter for transmit/receive signal to FIFO

Fig. 7 below illustrates the summarized operation shift converter receiving package data bus from FIFO to UART (RS232 Protocol Engine). From the starting point, if "reset" is at logic '1', then all the register will be cleared into '0' including "data_hold" and output "data_rx" bits logic. This 'data_hold' will act as special internal register and decoded by sampling each frame received. Otherwise, if "store" and "reset" at bit '1' and '0' logic respectively the "shift" operation was force into on and "data_in" as bit input is loaded into the register "data_hold." Next, if "shift" is at logic '1' will start shifting operation using 'count' concept operation. Finally, outputs "data_rx" will transmit bit to RS232 Protocol Engine. In order to checking the bit's operation to UART output register "cs_rx" stands for current state inserted. The Verilog HDL code will be compiled and synthesized in simulation using Modelsim-Altera device in order to obtain and analyzed the result.

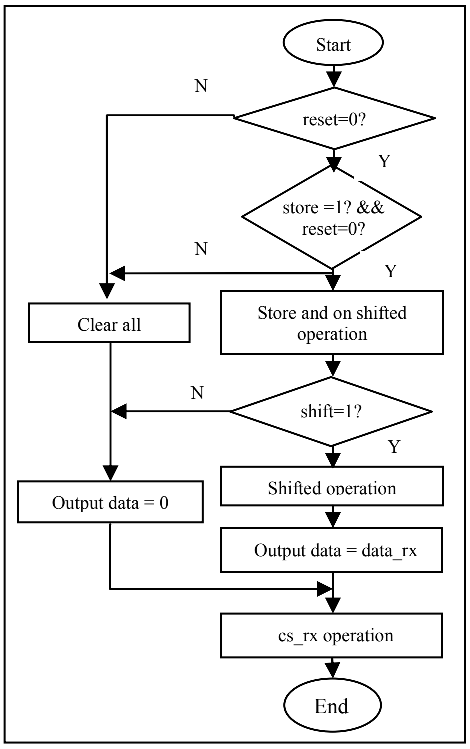

Fig. 7. Flow chart shift converter for transmit/receive signal to UART

The last part in the design was the asynchronous FIFO where data values are written sequentially into a FIFO Buffer using a one clock domain, and the data values are sequentially read from the same FIFO Buffer using another clock domain. These two "clk" domains are asynchronous to each other. The FIFO depth used is 16. In order to facilitate error-free operation, FIFO Full Signal and FIFO Empty Signal were used. FIFO Full Signal used a "wr_clk" while FIFO Empty Signal is driven by "r_clk." When the FIFO is in-state Full Signal, it will stop writing into FIFO. However, if it detected in a state Empty Signal, it will stop reading from it.

IV. RESULT AND DISCUSSION

The architecture is modeled in Verilog HDL. The functionality and characteristic of all designs were compiled and synthesized using Quartus II software. Then, the results of the simulation in ModelSim-Altera were compared and analyzed with the theoretical expectation. The "conv_bit" was applied with 1ns\1ns timescale. Final simulation results for a designed shift converter bit was carried out and divided into two. The first waveform results from a converter bit "conv_bit" analyzed between RS232 Protocol Engine to FIFO and second waveform result between FIFO to RS232 Protocol Engine.

All the operations involved were designed based on their characteristics. After synthesized using ModelSim-Altera the architecture is run and compiled using TimeQuest Timing Analyzer to checking the summary of timing clock transmit and receive data through the shift converter. From this it is an ability to show the summary report and create timing Netlist using SDC language for all input and output timing clock and so on including specific device reports for metastability.

A. Simulation Waveform Result

The result as shown in Fig. 8 below indicates the transmitting waveform carried out for a bidirectional shift converter bit "conv_bit" between RS232 Protocol Engine to FIFO using 1ns/1ns timescale. Two packages 1-byte bus input data '11010010' and '10100100' respectively were applied that receive from TXD port in RS232 Protocol Engine one by one bit. While the rest of the inputs port "reset" and "load" were forced at certain logic to be implement in embedded converter from RS232 to USB. However, as shown in Fig. 8 below, inputs port "reset" and "load" was forced at logic '0' and '1' respectively. The input's data in "datain_tx" one by one bit were shifted serially every rising edge "clk" to the 1-byte bus at output data "dataout_tx" starting '10000000', '11000000' and so on from 24900ns until the last byte first package '11011110' at 25700ns then followed 25750ns to 26510ns per byte data bus for the second package. Furthermore, the output port "cs_tx" clearly show the currently state operation.

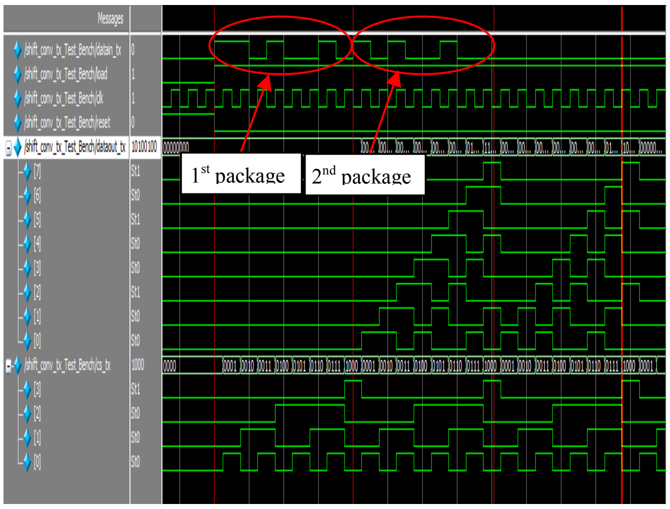

Fig. 8. Simulation waveform result for transmit signal to FIFO

The result shown in Fig. 9 below depicts the receiving waveform carried out for a converter bit between FIFO to RS232 Protocol Engine. Four inputs package 1-byte widths bus data '10110101', '10001101', '10010001' and '10001101' respectively was transmitted serially one by one to input port FIFO in 1-byte bus package data each time. The rest of the inputs port which are "reset" and "store" were forced at certain logic to check it functionality in order to be implement in embedded converter from RS232 to USB. However, as shown in Fig. 9 below inputs port "reset" and "store" was forced at "0" and "1" respectively. This input package of bus data "data_in" were loaded into "data_hold" before shifted serially to one by one bit at output port "data_rx." After it had done, it will continue with the next package bus data as describe above until the last package. Therefore, based on the two output waveform the output data was totally depending on the input data before fully transmit to UART shifted serially one by one all the receive package input data from FIFO.

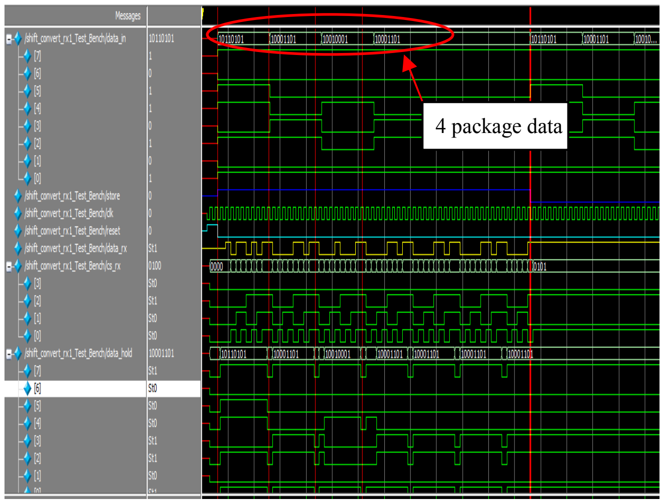

Fig. 9. Simulation waveform result for receive signal to UART

Then, the architecture is compiled and stimulates using TimeQuest Timing Analyzer simulation to check the timing summary of the database. Based on Fig. 10 below, it has shown the transmission-line clock delay for the data arrival is 2.685ns with zero metastability of a device. It is acceptable for reliable communication as theoretical expected. Besides that, a small delay occurred at the output data after rising clock edge, which is the most probably caused by net delay pins on EP2C35F672C6 FPGA. .

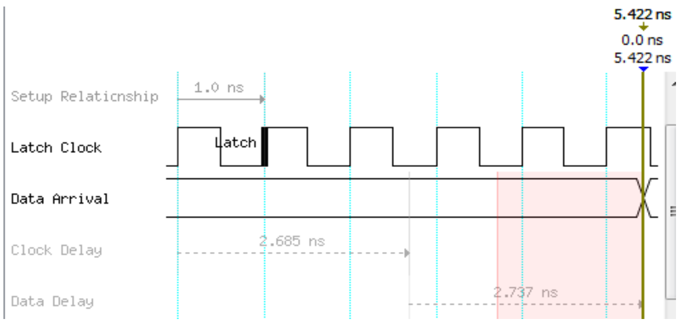

Fig. 10. Simulation waveform using Time Quest Timing Analyzer

V. CONCLUSION

Based on the results and discussions has been discussed in the previous section gives a clear perspective that the objective of this study has been successfully achieved. The entire physical layer design of bidirectional shift converter block's technique for UART on FIFO and FPGA was developed using Verilog HDL and simulated using ModelSim-Altera. The simulation results show that the converter bit model follows the theoretical development as a part implemented in embedded converter RS232 to USB. The model of a shift converter intended for embedded converter RS232 to USB design will be implemented as a future work for this study.

VI. ACKNOWLEDGEMENT

This work is supported by the Fundamental Research Grant Scheme (FRGS) of Ministry of Higher Education (MOHE)

REFERENCES

[1] R.Wyrmas,W. Zhang, M.J. Miller, and R.Anjaria, *"Multiple AccessbOption for Multimedia Wireless System"*, in Proc. 3rd Workshop on Third Generation, pp 289-294, Apr.1992.
[2] Shouqian, Y., Y. Lili, et al. (2007). Implementation of a Multi-channel UART Controller Based on FIFO Technique and FPGA. 2nd IEEE Conference on Industrial Electronics and Applications, 2007. ICIEA 2007.
[3] L. K. Hu and Q.CH. Wang, "UART-based Reliable Communication and performance Analysis" , Computer Engineering, Vol 32 No. 10, pp15-21,May 2006.
[4] Z. Salcic, " FPGAs and CPLDs a challenge for complex digital system design", IPENZ '97 Conference , Wellington, February 1997.
[5] S. Guccione, List of FPGA-based computing machines, http://www.io.com/~ guccione/HW-list.html.
[6] S.H.M Ludwig, The design of a coprocessor board using Xilinx's XC6200 FPGA--An experiment report, FPL '96, Darmstadt, Lecture Notes in Computer Science, Springer, 1996.
[7] Cavanagh, Joseph J.F, *"Digital Design and Verilog HDL Fundamental*",in Proc.1st,pp620-622, 2008, ISBN 978-1-4200-7415-4