International Journal of Electrical, Electronics and Computer Engineering **4**(1): 66-72(2015)

ISSN No. (Online): 2277-2626

Multi-channel UART Controller with Programmable Modes

Sanjeev. A. Shukla*, Narendra. P. Patil* and Dinesh. S. Bhadane*

**Department of Electronic and Communication Engineering, Sandip Foundation, MSBTE, Nasik, (MS), India*

(Corresponding author: Sanjeev. A. Shukla)

(Received 04 February, 2015 Accepted 07March, 2015)

(Published by Research Trend, Website: *www.researchtrend.net)*

ABSTRACT: In the recent years the development in communication systems requires the data transmission to be performed faster and faster. To meet this demand the paper presents a high speed multi channel UART controller based on FIFO (First In First Out) technique. An Asynchronous FIFO is designed with dual port ram array and with read and right pointers. The structure of controller is designed with UART (Universal Asynchronous Receiver Transmitter) and FIFO circuit design, the structure of the controller is scalable and reconfigurable design. This controller reduces the synchronization error between the sub systems in a system with other sub systems. Mainly the controller is used to operate or implement the communication system when master equipment and slave equipment are set at different baud rate.

Keywords: FIFO, UART, Programmable Modes.

I. INTRODUCTION

Now-a-days, Micro controllers and digital signal processors (DSPs), in complex control algorithms can be easily implemented to attain the desired system performance. But in proposed control systems, it is difficult to attain the exact result for various factors such as which affect the control of the system, it means control algorithms are capable of controlling and implementing equipment and states of control circumstance [1]. Except those factors, communication parameters of control systems include Bit Error Rate, Baud Rate and synchronization between sub-systems also causing great effect. In order to improve precision of control system and make better use of modern control algorithms, we should pay much more attention on communication methods in control systems. In this paper we propose a Multi-UART controller that will use a serial communication circuit UART. Universal Asynchronous Receiver Transmitter (UART) circuits are popular and widely used in several systems. A universal asynchronous receive/transmit (UART) is an integrated circuit which plays the most important role in serial communication. It handles the conversion between serial and parallel data. Serial communication reduces the distortion of a signal, therefore makes data transfer between two systems separated in great distance possible [2]. we design a multi-channel UART controller based on FIFO techniques. It can receive data with a UART block at a certain Baud Rate and transmit data to sub-equipment with a UART block at the same Baud Rate or at other kind of Baud Rate which is different from the receiving Baud Rate. And it also can be used to reduce time delay between sub controllers. In this paper, using FIFO technique, Baud Rate generator is designed to implement communications within equipments at different Baud Rates. FIFOs are usually used for clock domains crossing to safely pass data from one clock domain to another asynchronous clock domain. Using a FIFO to pass data from one clock domain to another clock domain requires multi-asynchronous clock design techniques. There are different ways to design a FIFO right. This paper details one method that is used to design, synthesize and analyze a safe FIFO between two different clock domains using Gray code. FIFO is the most important part of these systems and it works as a bridge between different devices. So the features and capabilities of the asynchronous FIFO determine the features of our controller. FIFO can be used to complete communication in parallel or serial port.

II. IMPLEMENTATION OF ASYNCHRONOUS FIFO'S

A. Introduction to asynchronous FIFO

An asynchronous FIFO refers to a FIFO design where data values are written to a FIFO buffer from one clock domain and the data value are read from the same FIFO buffer from another clock domain, which are asynchronous to each other. Asynchronous FIFOs are often used to quickly and safely pass data from one clock domain to another asynchronous clock domain.

In asynchronous clock circuit, periods and phases of each clock domain are completely independent so the probability of data loss is always not zero. This paper introduces a way of designing FIFO based on FPGAs with high write/read speed and high reliability.

Generally, a FIFO consists of a RAM Array block, a Status block, a writer pointer (WR_ptr) and a read point (RD_ptr) and its structure. A RAM array with separate read and write ports is used to stored data. The writer pointer points to the location that will be written next, and the read pointer points to the location that will be read currently. A write operation increments the writer pointer and a read operation increments the read pointer. On reset, both pointers are reset to zero, the FIFO is empty. The writer pointer happens to be the next FIFO location to be written and the reader pointer is pointing to invalid data. The responsibility of the status block is to generate the "Empty" and "Full" signals to the FIFO. If the "Full" is active then the FIFO cannot accommodate more data and if the "Empty" is active then the FIFO cannot provide more data to readout. When writing data into the FIFO "wclk" will be used as the clock domain and when reading data out of the FIFO "rclk" will be used as the clock domain. These both clock domains are asynchronous. In designing of asynchronous FIFOs, two difficult problems cannot be ignored. One is how to judge FIFOs status according to the writer pointer and read pointer. The other is how to design circuit to synchronize asynchronous clock domains to avoid Metastability. B. Status of Empty and Full of FIFO Creating empty and full signals is the most important part of designing a FIFO. No matter under what circumstance, the read and write pointers cannot point to the same address of the FIFO. So, the empty and full signals play very important roles within FIFO that they block access to further read or write respectively. The critical importance of this blocking lies in the fact that pointer positions are the only control that is over the FIFO, and write or read operation changes the pointers. In order to exactly know whether the FIFO is full or empty, we can set a direction flag keeps track of what causes the pointers to become equal to each other. The flag tells the status circuit the direction in which the FIFO is currently headed. The implementation of the direction flag is a little complex because you have to set the threshold of "going toward full" and "going toward empty". The status block fundamentally performs operations on the two pointers, and these run off two different clock domains. This is what causes the real difficulty. If you were to sample the read pointer with the write pointer (or vice versa), you will potentially run into a problem called metastability. Meta stability is the name for the physical phenomenon that happens when an event tries to sample another event. In a physical circuit the metastability causes the output uncertainty either be a logical 1 or a logical 0 or something between. In physical systems, sampling an event by another event yields unpredictable results. To eliminate these kinds of problems caused by metastability is a difficulty in designing a FIFO

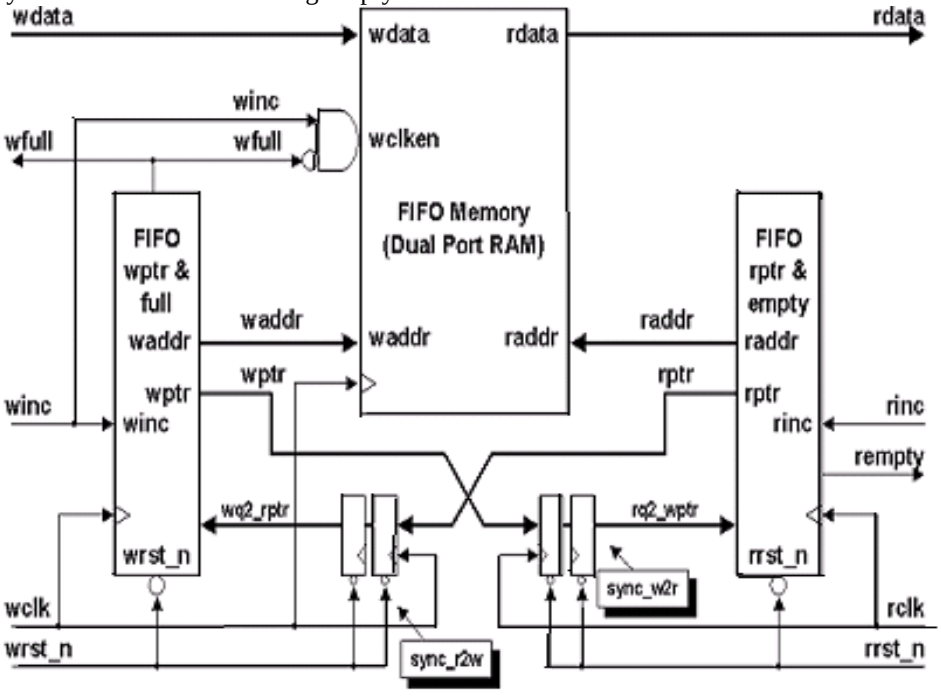
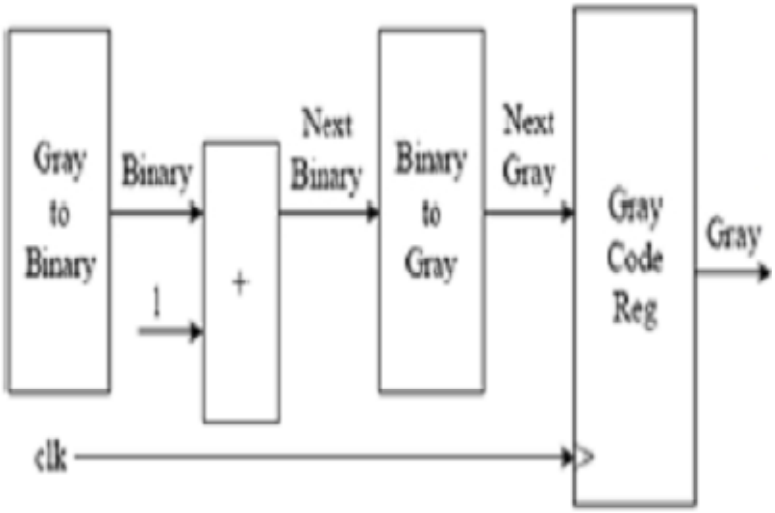


Fig. 1. Internal block diagram of asynchronous FIFO.

C. Solutions of Metastability

Metastability can cause unpredictable problems in a FIFO, so in the designing stage we should do the best to reduce the metastability. If asynchronous element is in a system, metastability is unavoidable. There is absolutely no way to eliminate metastability completely, so what we do is calculate a "probability" of error and express this in terms of time ie. MTBF (Mean Time between Failures). MTBF is a statistical measure of failure probability, and requires some much more complex, empirical and experimental data to arrive at. In a D flip-flop, when the input signal changes instantaneously from 0 to 1 at time, the value of Q is uncertain. This is metastability. In the FIFO, it needs to sample the value of a counter with a clock that is synchronous to the counter clock. Thus it will meet a situation where the counter is changing from FFFF to 0000, and every single bit goes metastable. This means that the counter would potentially read any value between FFFF to 0000 and the FIFO does not work. The most important things that must to be done are to make sure that not all bits of the counter will change simultaneously. In order to minimize the probability of occurrence of such errors, we should make sure that precisely one bit changes every time the counter increments. So we need a counter that counts in the Gray codes. Gray code is different form binary code that is every next value differs from the previous in only one bit position.

In a FIFO, converts the Gray code to Binary code, increments it and convert it back to the Gray code and store it. The Gray code counter assumes that the outputs of registers bits are the Gray code value. The Gray code outputs are then passed to the Gray to binary converter which is passed to a binary adder to generate the next binary value which is passed to the binary to Gray converter that generates the next Gray code value stored in register. The first fact to remember about a Gray code is that the code distance between any two adjacent words is just 1(only one bit can change from one Gray count to the next). The second fact to remember about a Gray code counter is that most useful Gray code counters must have power-of-2 counts in the sequence.



III. DESCRIPTION OF A MULTI-CHANNEL UART CONTROLLER

A. Hardware structure

In the multi-channel controller, there are different blocks including four UART's, two asynchronous FIFOs, one Baud Rate Generator, a register block and with a controller . Each block has different function in the controller. The first part is UART circuit block and its structure. It consists of three parts Receive Circuit, Transmit Circuit and Control/Status Registers. The Transmit Circuit consists of a Transmit Buffer and a Shift Register. Transmit Buffer loads data being transmitted from local CPU. And Shift Register accepts data from the Transmit Buffer and send it to the TXD pin one by one bit. The Receive Circuit consists of a Receive Shift Register and a Receive Buffer. The Receive Shift Register receives data from RXD one by one bit. The Control Register a special function register is used to control the UART and indicate status of it. According to each bit's value the UART will choose different kind of communication method and the UART knows what to do to receive or transmit data. . When writing data into FIFOs and reading data out of FIFOs we could set different clock domains according to the MCUs' Baud Rate. So it can be used to implement communications between MCUs at different Baud Rate.

The controller also has a block of Baud Rate Generator to engender different Baud Rates to content requirements for different kind of systems. This block is constituted by timers (32/16 bits timers), frequency dividers and a Baud Rate setting register.

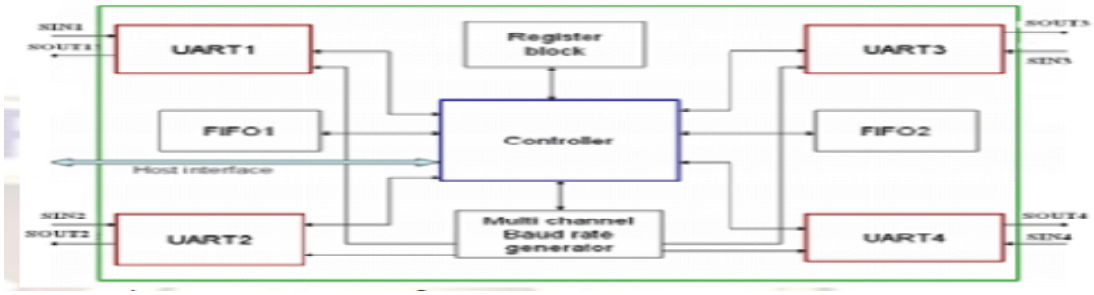


Fig. 3. Structure of Multi Channel UART.

A controller can also be used to complete communication between high speed device and low speed device.

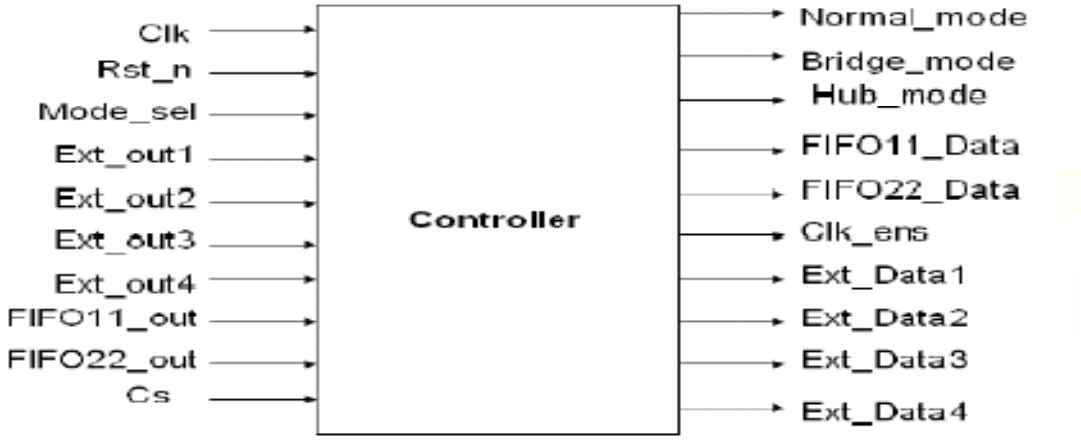


Fig. 4. Structure of the controller.

IV. PROGRAMMABLE MODES OF CONTROLLER

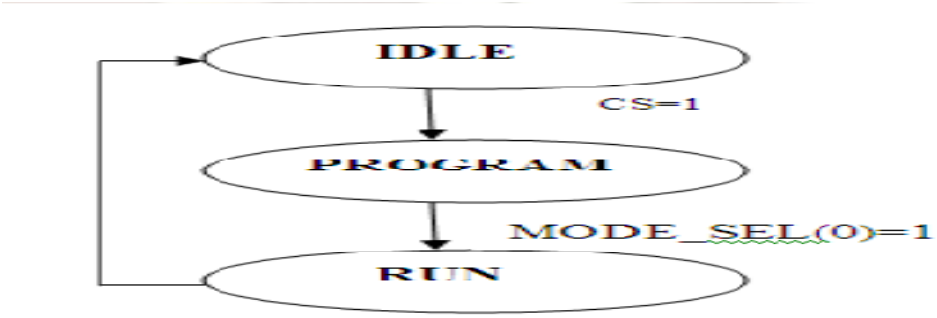


Fig. 5. States of controller to operate the modes.

If CS is low the controller will enter the IDLE state. The individual clocks of UART1, UART2, UART3 and UART4 are disabled in the IDLE state. If CS is high then the controller will enter the RUN state.

The Multi- UART controller can operate normally in the RUN state. The individual clocks of each of the UART are enabled in the RUN state.

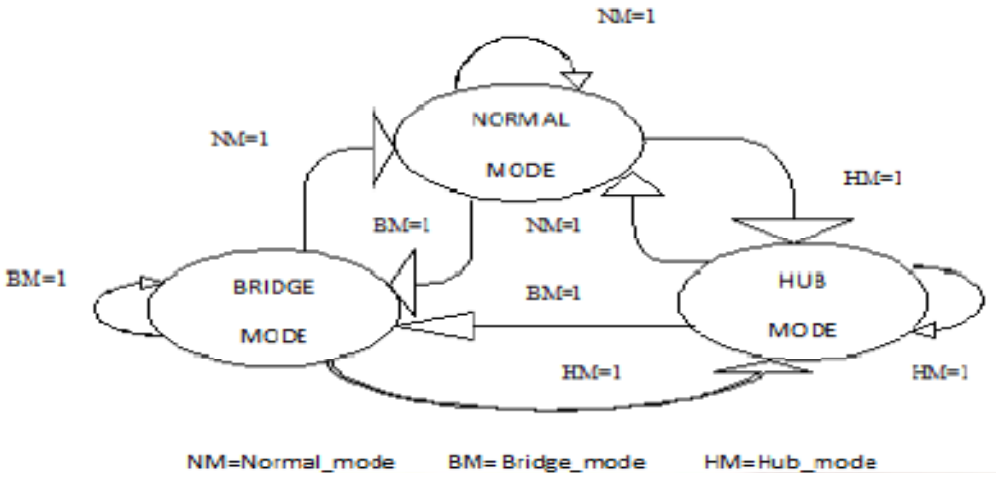


Fig. 6. State Diagram of Controller in RUN State.

In the RUN state the controller can operate in three different modes. The three different modes are—the normal mode, the hub mode and the bridge mode. Mode_sel is an input signal which determines the mode in which the controller is working. If the input to the Mode_sel[2:1] is "00" then the Normal Mode is activated and the Normal_mode which is an output signal is made high. Once the controller is in the normal mode, the controller sets the Normal Mode register in the Status Register block. All other mode registers are reset. In the normal mode UART1 will receive data and UART3 will transmit that data. Similarly in the normal mode UART2 will receive data and UART4 will transmit that data.UART1 and UART2 can independently receive data at different baud rates.UART3 and UART4 can independently transmit data at different baud rates. If the input to the Mode_sel[2:1] is "01"in binary or then the Hub Mode is activated and the output Hub_mode is made high while Normal_mode and Bridge_mode outputs go low. Once the controller is in the Hub Mode the controller sets the Hub Mode register in the Status block. The Normal Mode and Bridge Mode registers are reset. In the Hub Mode UART1 will receive data while UART2, UART3 and UART4 transmit that data. If the input to the Mode_sel[2:1] is "11" in binary then the Bridge Mode is activated and the output Bridge_mode is made high while Normal_mode and Hub_mode outputs go low. In theBridge Mode, the controller sets the Bridge Mode register in the Status Register block. The two other mode registers are reset. In the Bridge Mode, UART1 will receive data which is transmitted by UART2 at different baud rates. Simultaneously in the Bridge Mode UART3 will receive data and UART4 will transmit that data at different baud rates.

B. Software structure

You can use software codes in Verilog HDL to design FPGAs hardware architecture it is easy to create and adjust to satisfy requirements of applications. Here are one UART used to communicate with PC or other main MCU and there are also four other UARTs used to communication with sub MCUs. Each channel has two FIFOs, one for receiving data and the other for transmitting data. Each FIFO's depth is 64 Bytes. when FIFO is full you cannot write any more byte into the FIFO. At this time, the Status Detector will set CS high to indicate that the FIFO is full and stop writing to the FIFO. When FIFO is empty you cannot read from it any more. Then the Status Detector will set Empty high to indicate the status of FIFO and stop reading from it. When FIFO is not full or empty it will be written or read data according the control order. After finishing all write or read operation it will stop until next access is coming.

IV. SIMULATION AND VERIFICATION

To verify design of the controller a test bench is written to make verification in ISE simulator The software structure involved in the design of the following blocks-UART block, FIFO blocks, Status Register Block, Baud Generator block. The controller which interacts with all of the above was designed and its design was discussed earlier. Some components like UART and FIFO blocks are used more than once. A single UART was designed and verified. Then UART component was instantiated four times to obtain four independent UARTs. Similarly once FIFO block was designed and verified, it was instantiated twice to obtain FIFO1 and FIFO2.Codes in Verilog HDL were used to design the architecture of the Multi UART controller.

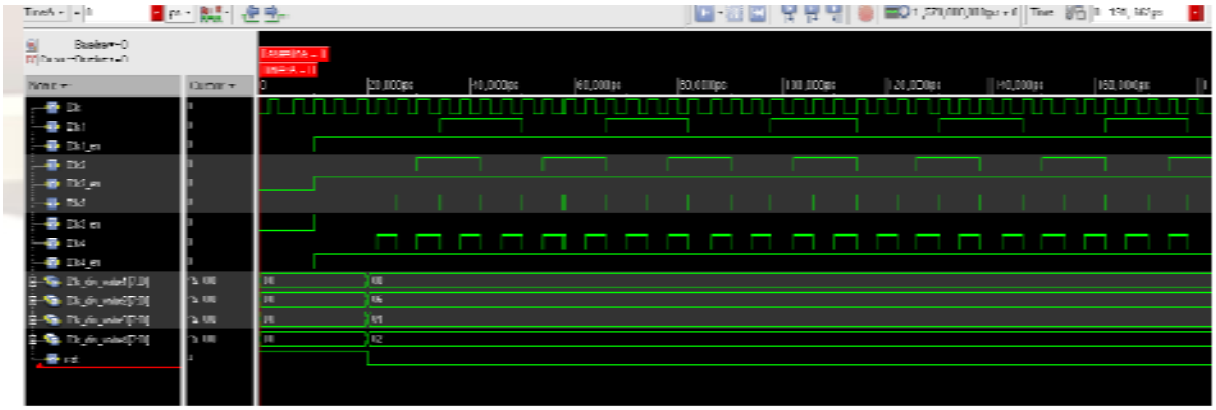


Fig. 7. Baud rate generator output.

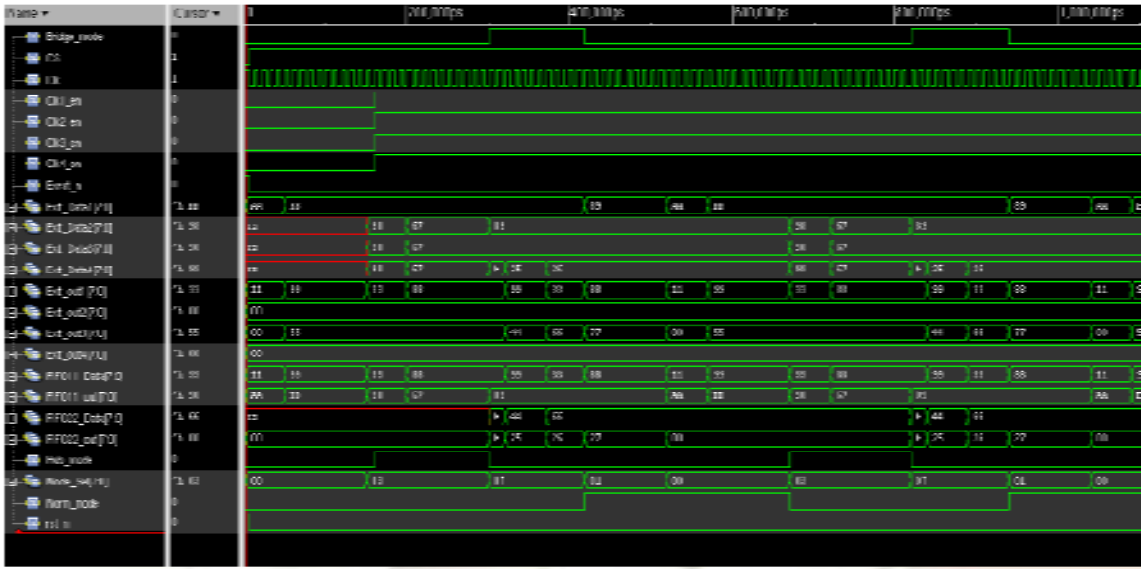


Fig. 8. Controller Output.

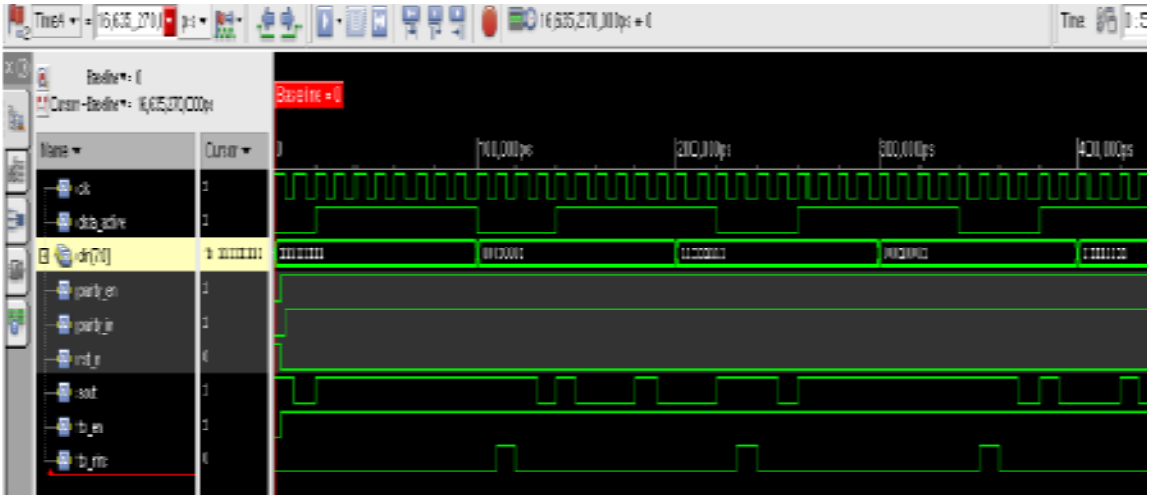


Fig. 9. Transmitting Sequence.

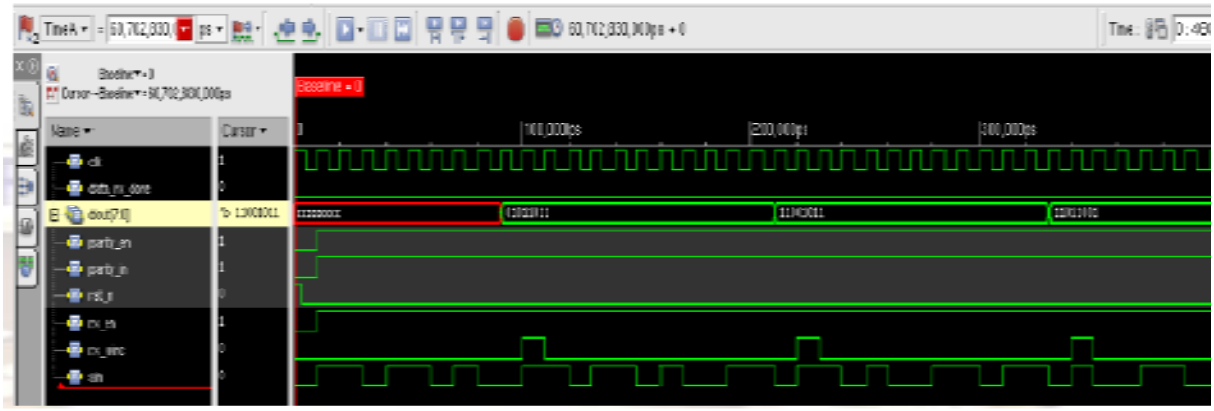


Fig. 10. Receiver Response.

V. CONCLUSIONS

This paper introduces a method to design a synchronous FIFO and using Asynchronous FIFO technique implements a multi-channel UART controller based on SRAM with high speed and high reliability. The controller is reconfigurable and scalable. The controller can be used to implement communications in complex system with different Baud Rates of sub-controllers. And it also can be used to reduce time delays between sub-controllers of a complex control system to improve the synchronization of each sub-controller

REFERENCES

[1]. S. E. Lyshevski, "Control Systems Theory with Engineering Applications", Birkhauser Boston, 2001.

[2]. L. K. Hu and Q.CH. Wang, "UART-based Reliable Communication and performance Analysis", *Computer Engineering*, Vol. **32** No. 10, May 2006, pp15-21.

[3]. F.S. Pan, F. ZHAO, J. Xi and Y. Luo, "Implement of Parallel Signal Processing Syttem Based on FPGA and Multi-DSP", *Computer Engineering* Vol. **32**, No. 23, Dec 2006, pp247-249.

[4]. X. D. Wu and B. Dai, "Design of Interface between High Speed A/D and DSP Based on FIFO", *Journal of Beijing Institute of Petrochemical Technology*, *Vol*. **14** No.12, June 2006, pp26-29.

[5]. C. E. Cummings, "Simulation and Synthesis Techniques for Asynchronous FIFO Design", *SNUG San Jose 2002*.

[6]. C. E. Cummings, "Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons", SNUG San Jose 2002.

[7]. Vijay A. Nebhrajan, "Asynchronous FIFO Architectures", www.eebyte.com

[8]. X., Yang, "Industrial Data Communication and Control Networks", Beijing: TUP, 2003.6.

[9]. B. Zeidman, "Designing with FPGAs & CPLDs", CMP Books, 2002.