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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

Advanced Design and Comparison of Multi-Channel UART

A.N.Sai Chakravarthy, K.Anil Kumar, K.Rajesh Kumar, S.Gangadhar

Dept. of Electronics & Communication, KMIT, JNTU, Hyderabad, India

ABSTRACT: This paper presents the efficient use and comparison of single UART and Multi-channel UART in accordance with the speed, time utilization, reliability, performance, number of flip-flops, area and number of modules. In basic communication transfer of data within bits of information from source can be occurred by serial/parallel transmission at the same time at destination data can be received by serial/parallel reception through means of a data bus/cable. In high speed media processors where the central processor takes data from several low speed peripherals simultaneously. This can be achieved by designing of a multi-channel UART with asynchronous FIFO (First-In-First-Out) it is easily to know that this controller can be used to communicate when master equipment and slaver equipment are at different Baud Rate. It also can be used to reduce synchronization errors time delays between sub-controllers of a complex control system to improve the synchronization of each sub-controller. The controller is reconfigurable and scalable.

KEYWORDS: UART (Universal Asynchronous Receiver Transmitter), FIFO (First In First Out), FPGA (Field Programmable Gate Array), MCUART (Multi-Channel UART), BER (Bit Error Rate).

I. INTRODUCTION

In actual control systems, it is difficult to attain the expected result for various factors affect the control systems such as control algorithms itself, capability of controllers, capability of implement equipment and states of control circumstance. Except those factors, communication parameters of control systems including Baud Rate, BER (Bit Error Rate) and synchronization between sub-systems also engender great effect. In order to improve precision of control system and make good use of modern control algorithms, we should pay much more attention on communication in control 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. Parallel communication needs a lot of multi-bit address bus and data bus and it is only convenient for short distance transmission. Serial communication is another way of communication used extensively because of its simple structure and long transmission distance.

In a system, the PC's Baud Rate is 115200bps and the Ep1 i.e. equipment 1's Baud Rate is 57600bps, equipment 2's Baud Rate is 19200bps, and other equipments are set at 9600bps or other Baud Rates. It is impossible to implement this multi-Baud Rate communication system without a special Baud Rate converter. In a 6-DOF robot, there are 6 sub-controllers which are all the same structure to be designed. The PC is used to implement the control algorithm of the robot and send control parameters to sub-controllers and sub-controllers are used to collect feedback signals and send them to the PC. The PC and sub-controllers communicate with each other on a RS485 BUS NET. Each sub-controller has a unique address number and the PC uses this number to identify each sub-controller. When the PC wants to send data to node 6, it has to access front 5 nodes, this engenders time delay and makes performance of the robot's each DOF not synchronization. So it reduces the control algorithm's precision and brings difficulties in researching of the control algorithm.

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

Multi-channel UART controller 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. FPGA (Field Programmable Gate Array) is using extensively and playing more and more important roles in the designing of digital circuit. Its programmable characteristics make circuit design much more flexible and shorten the time to market. Using FPGAs can also improve the system's integration, reliability and reduce power consumptions. FPGAs are always used to implement simple interface circuit or complex state machines to satisfy different system requirements. 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 used to complete communication between high speed device and low speed device or to complete communication between the same sub controllers. FIFO is the most important part of these systems and it works as a bridge between different devices. FIFO can be used to complete communication in parallel or serial port. To obtain fast and effective communication in modern complex control systems with different baud rate of sub controllers. To reduce the time delay between sub controllers of complex control systems. To improve the synchronization of each sub-controller. To make Controller is reconfigurable and scalable. Universal Asynchronous Receiver Transmitter is an integrated circuit, which is used for transmitting and receiving data asynchronously via the serial port on the computer. It contains a parallel-to-serial converter for data transmitted from the computer and a serial-to parallel converter for data coming in via the serial line. The UART also has a buffer for temporarily storing data from high-speed transmissions. In addition to the basic job of converting data from parallel to serial for transmission and from serial to parallel on reception, a UART will usually provide additional circuits for signals that can be used to indicate the state of the transmission media and to regulate the flow of data in the event that the remote device is not prepared to accept more data.

A UART is usually an individual (or part of an) integrated circuit used for serial communications over a computer or peripheral device serial port. UARTs are now commonly included in microcontrollers. A dual UART, or DUART, combines two UARTs into a single chip. Many modern ICs now come with a UART that can also communicate synchronously; these devices are called USARTs.

The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Serial transmission is commonly used with modems and for non-networked communication between computers, terminals and other devices.

There are two primary forms of serial transmission: Synchronous and Asynchronous. Synchronous serial transmission requires that the sender and receiver share a clock with one another, or that the sender provide a strobe or other timing signal so that the receiver knows when to "read" the next bit of the data. In most forms of serial Synchronous communication, if there is no data available at a given instant to transmit, a fill character must be sent instead so that data is always being transmitted. Synchronous communication is usually more efficient because only data bits are transmitted between sender and receiver, and synchronous communication can be more costly if extra wiring and circuits are required to share a clock signal between the sender and receiver.

A form of Synchronous transmission is used with printers and fixed disk devices in that the data is sent on one set of wires while a clock or strobe is sent on a different wire. Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. When a word is given to the UART for Asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter. These two clocks must be accurate enough to not have the frequency drift by more than 10% during the transmission of the remaining bits in the word. (This requirement was set in the days of mechanical teleprinters and is easily met by modern electronic equipment.) In the multi-channel controller, there are different blocks including UART block, Status Detectors, asynchronous FIFOs block and Baud Rate Generator block. It can be used to implement communications between MCUs in a complex system. It can also be used to complete

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

communication between high speed device and low speed device. It is possible to design small scale memorizer like FIFOs. Structure of the controller is showing in Figure 4.1.

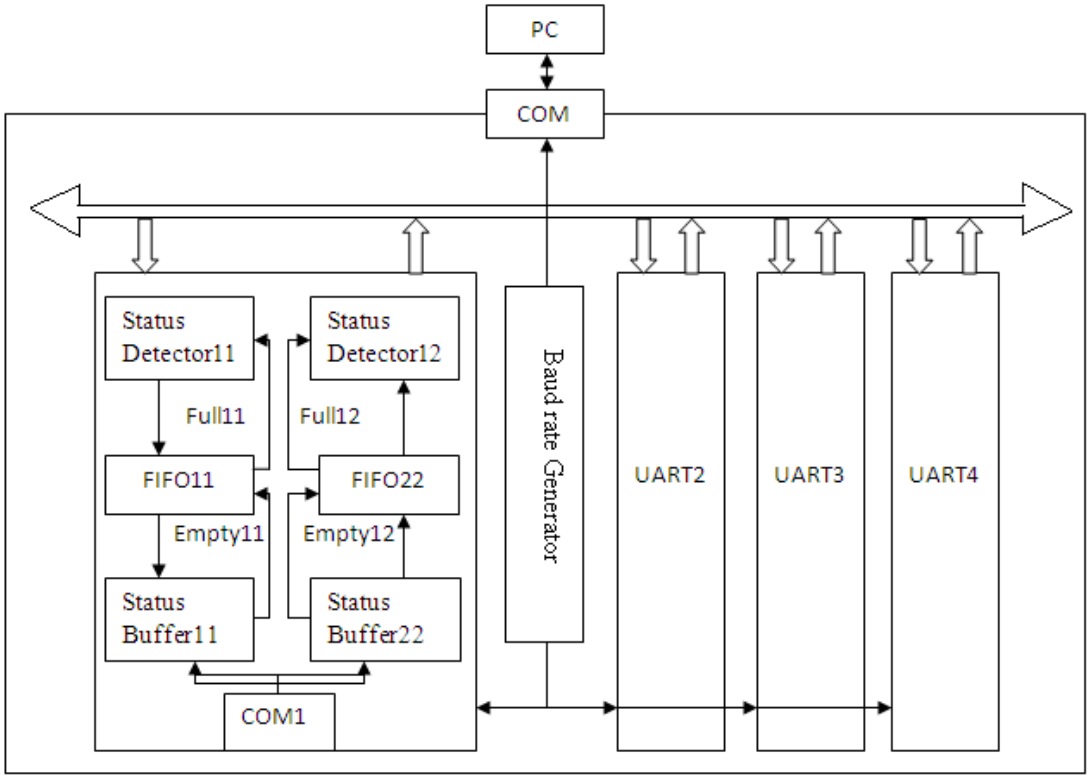

Figure 4.1: Structure of the multi channel UART controller

There 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. All sub-MCU can receive data at the same time. There are no time differences between sub-controller when the controller transmits data to sub-controllers at the same time. But in fact, there are hardware delays in FPGAs and these delays may cause sub controllers cannot receiver data from the controller at the same time precisely. Comparing with the delays in RS485 net these delays can be ignored. So using this controller can greatly improve synchronization of sub-controllers.

UART circuit block and its structure are shown in Figure 4.2(a). 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 sends it to the TXD pin one by one bit.

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

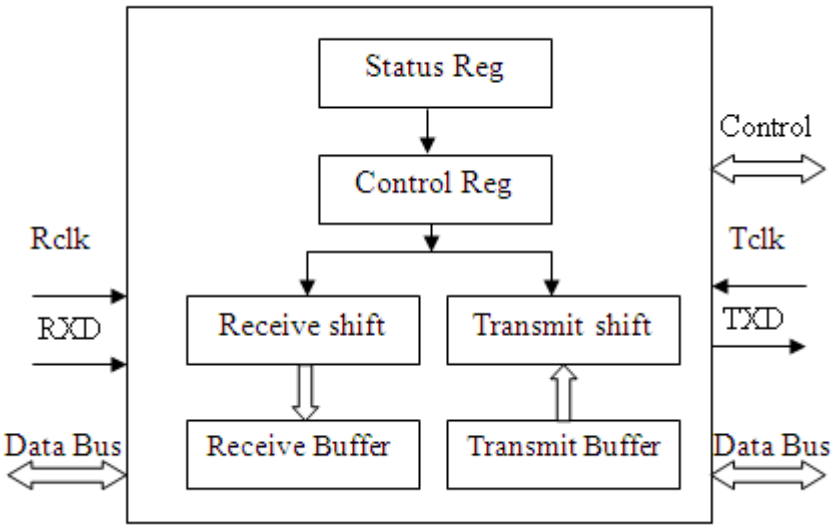

Figure 4.2(a): UART Block Diagram

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 Receive Buffer loads data from long-distance MCU and gets it ready for the local PC to read. 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. FIFOs are used to store data received from the PC and get ready for sub MCUs. When writing data into FIFOs and reading data out of FIFOs we could set different clock domains according to the PC's and 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. A baud rate generator is a programmable, bit timing device that is used to synchronize the bit duration of both the received and transmit sections of a serial communication device such as UART. Transmitter and receiver with independent 128-byte FIFOs to reduce the number of Interrupts to the processor.

- Accommodates 5, 6, 7, or 8 bits per character.
- Accommodates 1, 1.5, or 2 stop bits.
- Accommodates even, odd, or stick parity with enable/disable.
- Prioritized and independently controlled interrupts.
- Programmable baud generator.
- Modem control interface.
- Complete status reporting capabilities.
- Line break and false start bit detection.
- DMA signalling.
- Software compatible with 16550 UARTs.

The 8250 and 16450 are the first generation of UARTs and they are virtually obsolete today due to performance issues. There are several types of next generation, advanced UART standards, but the 16550 UART standard is the most popular and common UART. The architecture of the 16550 standard is based on the architecture of the preceding UART standards. The main difference between these UARTs is that each new generation offers increasingly deeper FIFO buffers and more features. Because these UARTs share the same base platform architecture, the designer can modify the source code provided with the 16550 UART core to build any type of UARTs provides a brief comparison of the different types of UART's.

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

Table 1: UART Types

UART STANDARD	Buffer Size (Bytes)	Feature Description
8250 / 16450	1	The original UART
16550	16	Most popular today.
16650	32	Sleep mode. "XOn/Xoff" special Character generation and detection.
16750	64	"Auto-RTS" and "Auto-CTS" flow rate control.
16850 / 16950	128	Advanced auto flow rate control. Support for 9-bit characters.

FIFO STRUCTURE:

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 is showing in Figure 4.3. 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.

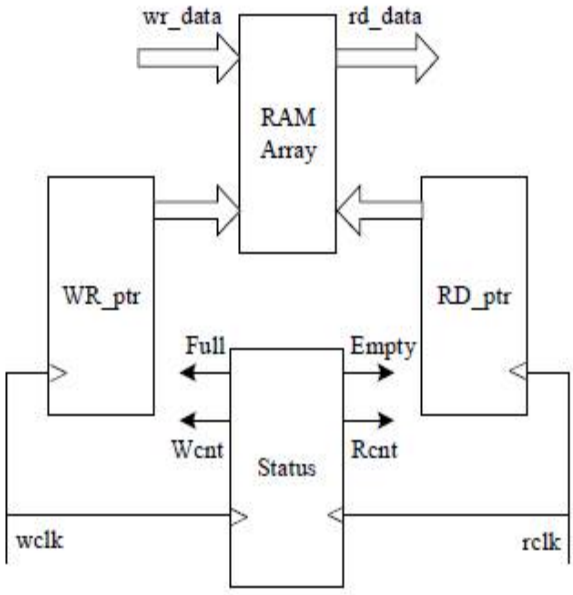

Figure 4.3: Asynchronous FIFO structure

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

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

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.

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. Generally, in an ordinary FIFO, when the read pointer equals to the writer pointer the FIFO is empty. But in a circular FIFO it is either empty or full when both of the pointers are equal. Because the full and empty signals can not only be decided by the pointers' value but also be influenced by the operation that caused the pointers to become equal. If a reset or read makes the pointers equal to each other, the FIFO is really empty. If a write makes the pointers equal, the FIFO is full.

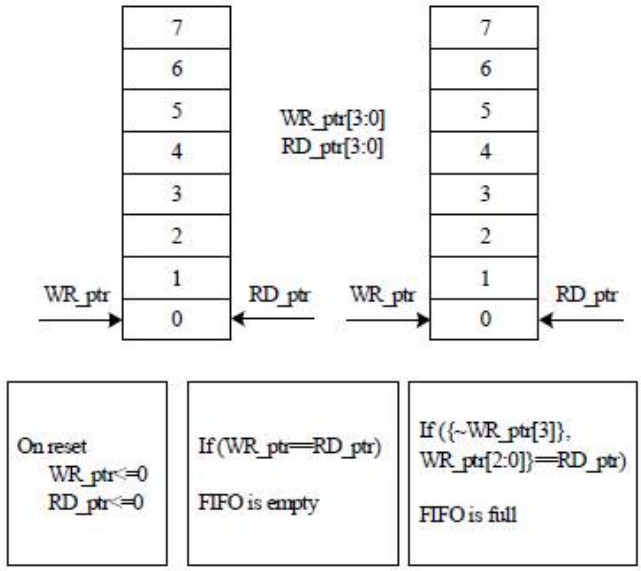

Figure 4.4: Asynchronous FIFO full and empty conditions

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.

II. META STABILITY

Usually the term is used to describe a state that doesn't settle into a stable '0' or '1' logic level within the time required for proper operation. This can cause the circuit to go into an undefined state and act in unpredictable ways, so it is considered a failure mode in a digital circuit.

Metastable states are believed to be inherent features of asynchronous digital systems and systems with more than one clock domain, but careful design can often make the probability of a system failing very small indeed. Metastable states do not occur in fully synchronous systems when the set-up time specifications on logic gates are satisfied. 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

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

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 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 codes are named after the person who originally patented the code back in 1953, Frank Gray. Gray code is different form binary code that is every next value differs from the previous in only one bit position.

$$g_n = b_n \oplus b_i \quad b_{i+1}\ i \neq n$$

and

$$b_n = g_n$$

$$b_i = g_i \quad b_{i+1}\ \forall\ i \neq n \quad \oplus$$

III. GRAY CODE CONVERSION

The 4-bit Gray code is a mirror image of the first half with the MSB inverted. To convert a 4-bit to a 3-bit Gray code, we do not want the LSBs of the second half of the 4-bit sequence to be a mirror image of the LSBs of the first half; instead we want the LSBs of the second half to repeat the 4-bit Subsequence of the first half. Upon closer examination, it is obvious that inverting the second MSB of the second half of the 4-bit Gray code will produce the desired 3-bit Gray code sequence in the three LSBs of the 4-bit sequence.

The only other problem is that the 3-bit Gray code with extra MSB is no longer a true Gray code (Gray 0100) to 8 (~Gray 1000) and again from 15 (~Gray 1100) to 0 (Gray 0000), two it's are changing instead of just one bit. A true Gray code only changes one bit between counts.

The binary-value incremented is conditioned with either an "if not full" or "if not empty" test to insure that the appropriate FIFO pointer will not increment during FIFO-full or FIFO-empty conditions that could lead to overflow or underflow of the FIFO buffer. If the logic block that sends data to the FIFO reliably stops sending data when a FIFO full condition is asserted, the FIFO design might be streamlined by removing the full-testing logic from the FIFO write pointer.

The FIFO pointer itself does not protect the FIFO buffer from being overwritten, but additional conditioning logic could be added to the FIFO memory buffer to insure that a write_enable signal could not be activated during a FIFO full condition. An additional "sticky" status bit, either ovf (overflow) or unf (underflow), could be added to the pointer design to indicate that an additional FIFO write operation occurred during full or an additional FIFO read operation occurred during empty to indicate error conditions that could only be cleared during reset.

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

IV. FIFO SIMULATION RESULT

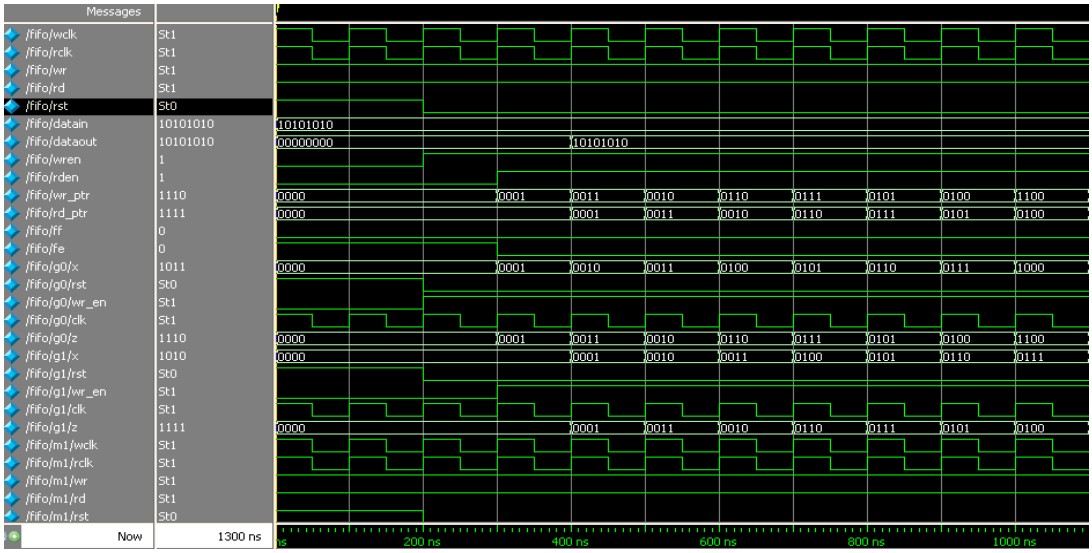

Figure 7.1 FIFO Simulation Result

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, where the two clock domains are asynchronous to each other. FIFO timing diagram is shown in Figure.7.1.when FIFO is full; we cannot write data in to the FIFO. And status block indicates FF signal to high. That means FIFO is full. When FIFO is empty, we cannot read data from the FIFO. And status block indicates FE signal to high. That means FIFO is empty.

UART SIMULATION RESULT:

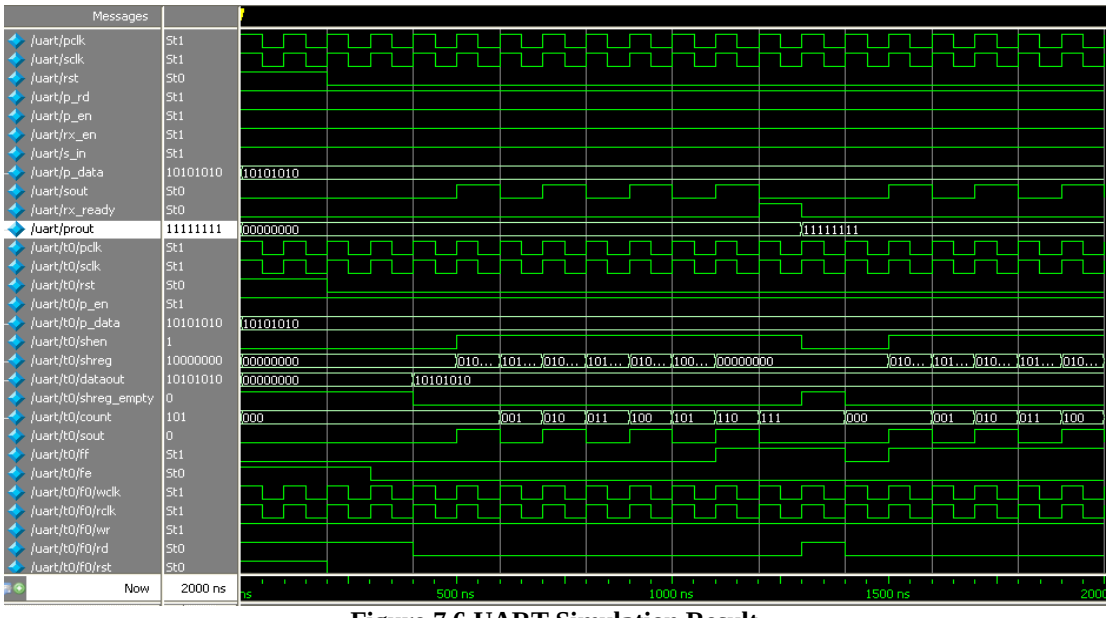

Figure 7.6 UART Simulation Result

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

UART is a universal asynchronous receiver and transmeter which has both transmitter block as well as receiver block. Transmitter input connected to high speed device and serial output pin is connected to the slow devices, for the receiver is vice versa.Its main application is to synchronoise the slow and high speed devices.

BAUD RATE SIMULATON RESULT:

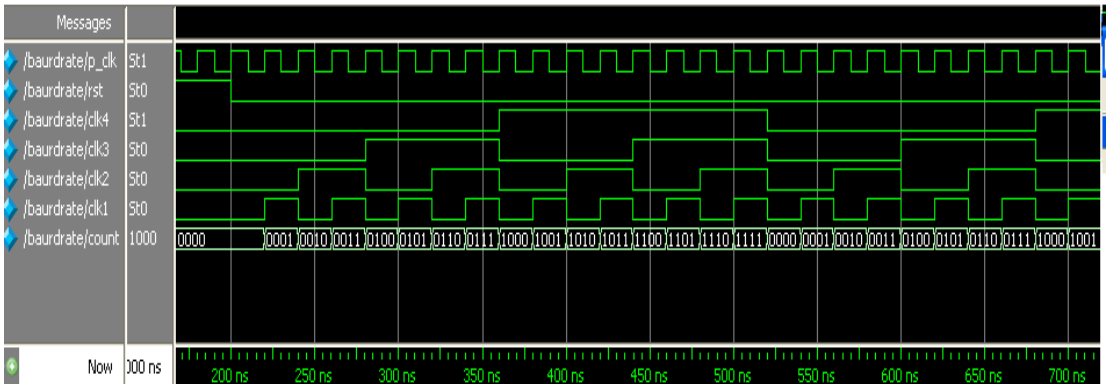

Figure 7.8 Baud Rate Simulation Result

Baud rate simulation result is shown in the Figure 7.8. A baud rate generator is a programmable, bit timing device that is used to synchronize the bit duration of both the receive and transmit sections of a serial communication device such as UART. The baud generator is capable of creating a clock by dividing the system clock by any divisor.

MCUART SIMULATION RESULT:

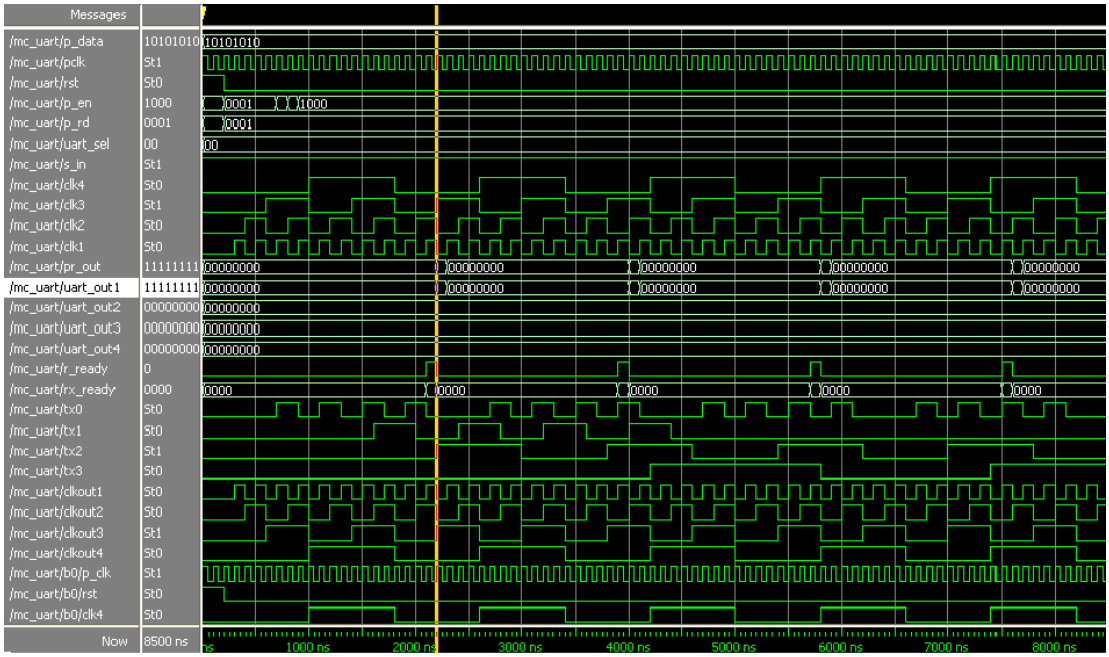

Figure7.10 MUART Simulation Result

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

MCUART is multichannel UART which has multiple UART to inter connect with the different high speed to low speed devices. By using MCUART controller we can controller the data transfer to the multiple UART's. The timing diagram of the MCUART is shown in the Figure 7.10. Data received from the PC or other main MCU will be stored in FIFOs within FPGA till the controller received the commands to order the controller to send data to sub-controllers. Then the controller will set a kind of Baud Rate according to commands desired. The controller is receiving data and store the data received to different FIFO waiting for read.

When sub-controllers are required to receive data at different Baud Rates, the controller can set each channel at its required Baud Rate to transmit data. The transmitting sequence is showing as following in Figure7.10. The controller sends data at the same time but at different Baud Rate. When sub-controllers are required to receive data at the same Baud Rate, the controller can also set all channels at the same Baud Rate to transmit data as showing in Figure7.10. All sub-MCU can receive data at the same time.

TRANSMITTER SIMULATION RESULT:

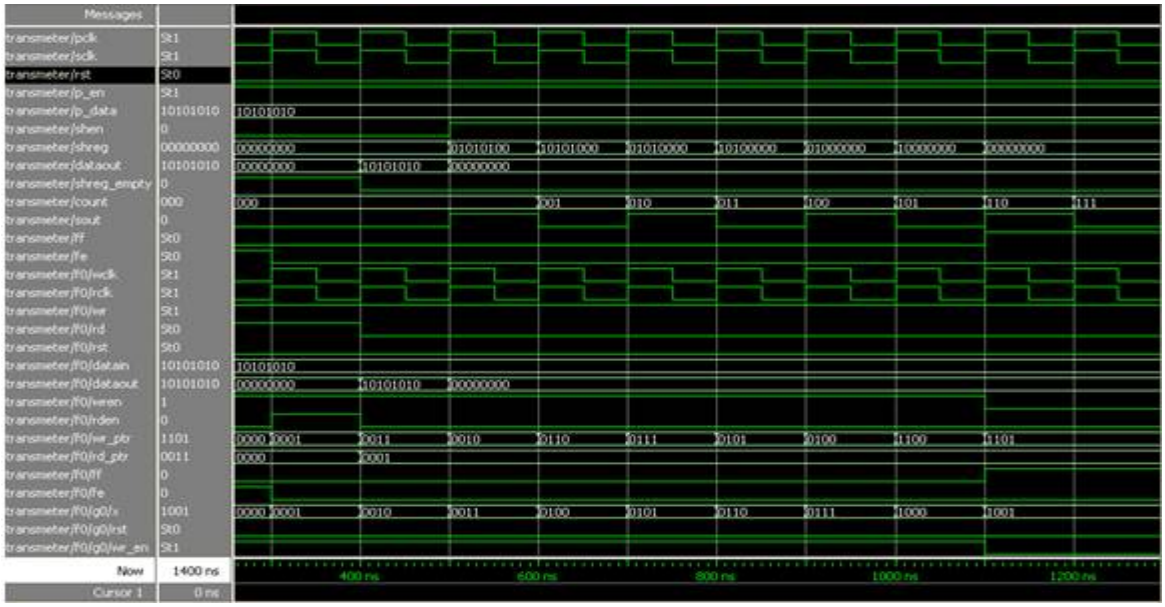

Figure 7.2 Transmitter Simulation Result

The stored data is then transferred out one bit at a time through the slow clk input to slow devices. For the Figure shown in 7.2 input data to the transmitter has given 10101010 at clock frequency of 10M HZ. So the result at the output of transmitter output at a freq of 2.5MHZ.

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

RECEIVER SIMULATION

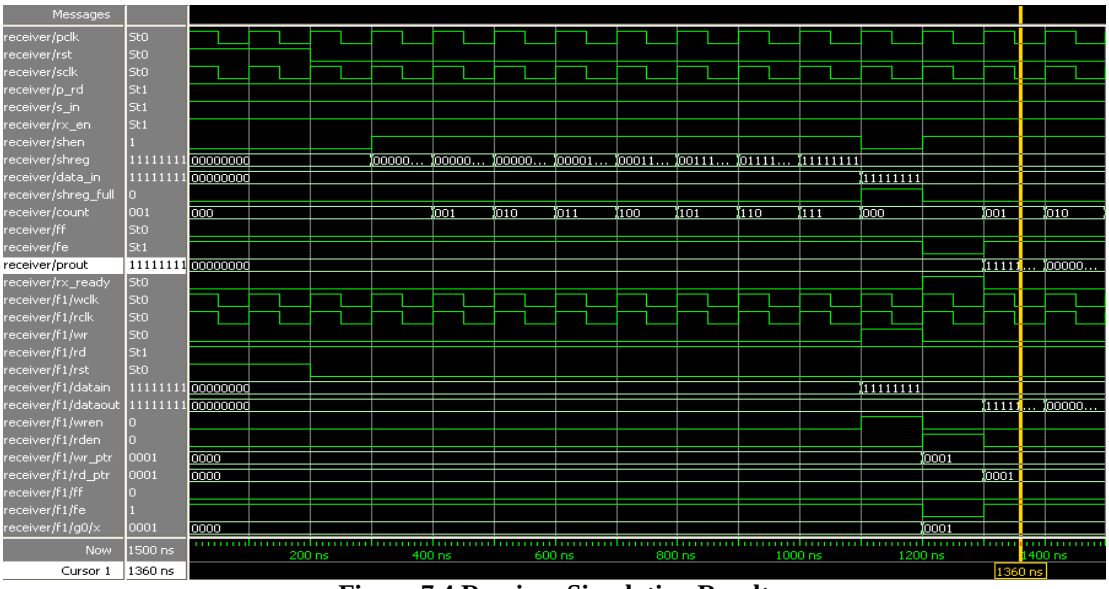

Figure 7.4 Receiver Simulation Result

V. RESULT

The Receive Buffer loads data from long-distance MCU and gets it ready for the local PC to read. The input and outputs are as shown in the Figure 7.4. For the receiver input is one bit at a time with a frequency of 10MHZ and the output is fed to the processor with 0.83MHZ frequency. The Receive Shift Register receives data from s_in one by one bit.

RESULT COMPARISION:

Table 3: Result comparison

Device utilization	For single UART	For FOUR UARTS	MCUART
Number of Slices:	256	1024	780
Number of Slice Flip Flops:	240	1040	776
Number of 4 input LUTs:	290	1060	1054
Minimum period:	6.338ns	6.338ns	6.276ns
Minimum input arrival time before clock:	7.508ns	7.508ns	7.613ns
Maximum output required time after clock:	8.285ns	8.285ns	9.029ns
Maximum combinational path delay:	7.074ns	7.074ns	8.911ns

ADVANTAGES:

1. Multi channel UART providing a high band width solution for communication systems requesting high data rate serial transmission protocol.

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

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

(An UGC Approved Journal)

Website: www.ijareeie.com

Vol. 6, Issue 8, August 2017

2. A multi channel UART is more hard ware efficient then many individual UART`s implemented individually component reuse between several channels gives scope for hard ware reduction.
3. The main advantage is UART controller is reconfigurable.

VI. CONCLUSION

The controller is used to implement communications in complex system with different Baud Rates of sub-controllers. It can be used to reduce time delays between sub-controllers of a complex control system to improve the synchronization of each sub-controller. The controller is reconfigurable and scalable. Using asynchronous FIFO technique implements a multi-channel UART controller with high speed and high reliability.

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

Websites:

- www.researchbooks.org
- www.freebsd.org
- www.xilinx.com
- www.model.com