Multi-Channel UART Controller with FIFO and FPGA

T.Janardhan [1] and G.Venkateswarlu[2]

[1]CVR College of Engineering/EIE Department, Hyderabad, India
Email: janardhanraju3001@gmail.com
[2] CVR College of Engineering/EIE Department, Hyderabad, India
Email: venkigummadilli@gmail.com

Abstract—**Communication between the complex control systems can be done by a multi-channel UART controller based on FIFO (First In First Out) technique and FPGA (Field Programmable Gate Array).The communication between the master equipment and slave equipment with different baud rate can be done by using this controller. This controller consists of a FIFO circuit block and UART circuit block. In this controller an asynchronous FIFO is used. The design method of an asynchronous FIFO is also presented. A universal Asynchronous Receiver-Transmitter usually (UART) converts data from parallel to serial and serial to parallel. It operates on any two independent clock domains.**

Index Terms: FIFO, FPGA, UART.

I. INTRODUCTION

In serial communication, a UART (universal asynchronous receive/transmit) plays an important role. A universal asynchronous receive/transmit (usually abbreviated UART) has a transmitter section and a receiver section. The distortion of a signal can be reduced in serial communication. Serial communication is preferred for long distance transmission. Parallel communication requires multi- bit address bus and data bus, and it is implemented for short distance transmission. We have a parallel or serial port, and the communication between any two systems may be serial or parallel communication. If the Master Control Unit (MCU) is sending or receiving the data at one baud rate and the sub-equipments are receiving or sending the data at another baud rate then there is a possibility of loss of data. To overcome this problem, an asynchronous FIFO is placed between MCU and sub-equipments. In designing a FIFO, Gray code is used for addressing the memory locations. The advantage of Gray code is that there is only one bit change in the address of the successive address memory location. This will overcome the Met-stability condition [1]. An asynchronous FIFO has either full or empty condition. FIFO is an important part of these systems and it acts like a link between the devices.

The features of multi-channel UART controller depend on the asynchronous FIFO. The communication between the MCU and the multi-sub equipments is shown in the below figure.1. The MCU baud rate is different to that of the sub-equipments. The baud rates of the sub-equipments are different to each other. A special baud rate converter is used to implement the communication between the MCU and the sub-equipment.

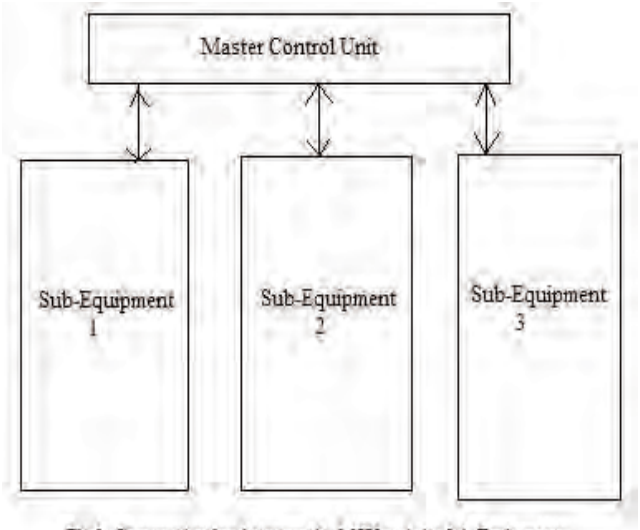

Fig.1: Communication between the MCU and the Sub-Equipments

II ASYNCHRONOUS FIFO DESIGN

A. Connections of an asynchronous FIFO

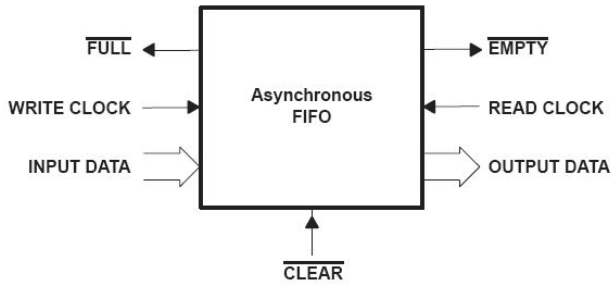

Fig.2. Connections of an asynchronous FIFO

In an asynchronous FIFO, the data is written based on the write clock pulse and the data is read based on the read clock pulse. The two operations in an asynchronous FIFO are read and write as shown in fig.2. The write and read operations takes place for two different clock domains. The status of an asynchronous FIFO can be determined by the FULL and EMPTY signals. When the FIFO is full there is no space for writing the data. When the FIFO is empty there is no data to read. The clear signal is used to erase the data in an asynchronous FIFO.

 DOI: 10.32377/cvrjst1017

B. ASM chart of an Asynchronous FIFO

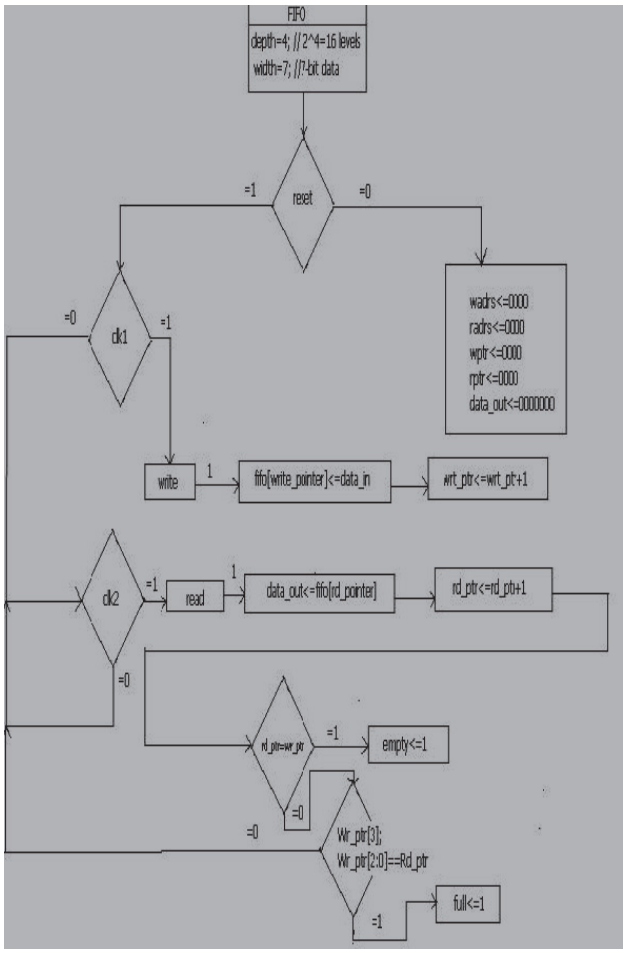

Fig.3: ASM chart of an Asynchronous FIFO

FIFOs are often used 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 [1].

An asynchronous FIFO refers to a FIFO design where data values are written to a FIFO buffer from one clock domain and the data values are read from the same FIFO buffer from another clock domain, where the two clock domains are asynchronous to each other. To determine full and empty status for an asynchronous FIFO design, the write and read pointers will have to be compared. 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 [1].

The ASM chart of an asynchronous FIFO is shown in the above figure.3. In this asynchronous FIFO, the numbers of memory locations are 16 and in each memory location 7-bits of data can be stored. When the reset control signal is 0, the write addresses, read address, write pointer, read pointer and data out are rested to zeros. When the reset control signal is 1,it checks the clock1 signal. If the clock1 signal is 1, then the write operation takes place. If the clock1 signal is 0, it checks the clock2 signal. If the clock2 signal is 1, then read operation takes place. When the clock2 signal is 0, again it checks the clock1 signal. If the read pointer and the write pointer points out to the same memory location, the FIFO is empty. If the MSB bit of write pointer is not equal to the MSB bit of read pointer and the remaining bits are equal then the FIFO is full.

C. Logic Involved in an asynchronous FIFO

Asynchronous FIFOs are used to safely pass data from one clock domain to another clock domain. Asynchronous refers to two different clock domains. The Master Control Unit (MCU) sends the data to FIFO at one clock domain and the sub-equipments receive the data from the FIFO at anther clock domain. A method is presented that is used to design, synthesize and analyze a safe FIFO between different clock domains using Gray code pointers that are synchronized into a different clock domain before testing for FIFO full‖ or FIFO empty condition [2]. For a FIFO design, we require the full component, empty component, and synch module component. In a FIFO, the data is written at clock1 frequency and the data is read at clock2 frequency. To determine full and empty status for an asynchronous FIFO design, the write and read pointers will have to be compared.

The FIFO is in full condition when

WPtr (4)/=WSync_rptr (4) and WPtr (3) /= WSync_rptr (3) and WPtr (2 downto 0) = WSync_rptr (2 downto 0).

The FIFO is in empty condition when

rptr = rsync_wptr

To avoid metastability, Gray code is used as an address of each memory location instead of binary code. The probability of occurrence of an error is very low by using Gray code.

The conversion between the Binary codes and the Gray codes is as following [3]:

$g_n=b_n$

$g_i=b_i \text{ Xor } b_{i+1} \quad i\neq n$ and

$b_n=g_n$

$b_i=g_i \text{ Xor } b_{i+1} \quad i\neq n$

D. Simulation result of an Asynchronous FIFO

Rst: When the control signal rst is _0‘ no operation takes place and when the control signal rst is _1‘ the write operation and read operation takes place, which is shown in Fig.4.

Clk1: When the control signal clk1 is _1‘ then write operation takes place.

Clk2: When the control signal clk2 is _1‘ then read operation takes place.

Radrs [3:0]: This control signal indicates the address of the memory location from where the data can be read.

Wadrs [3:0]: This control signal indicates the address of the memory location to which we can write the data is shown in Fig.4.

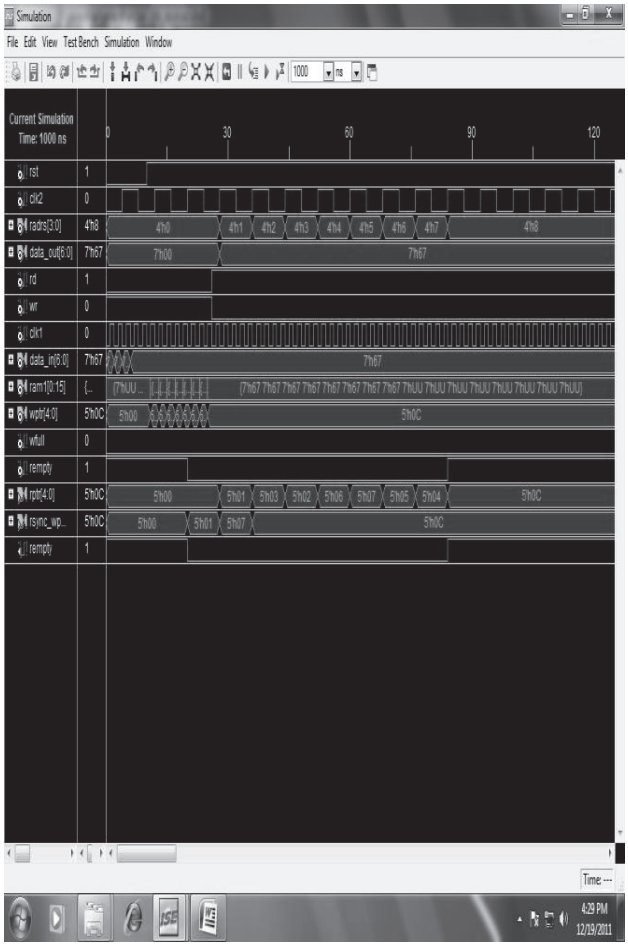

Fig. 4: Simulation result of an asynchronous FIFO

Data_out [6:0]: This control signal indicates the output data. The output data can be observed on the bus channel 21, bus channel 22, and the bus channel 23.

Rd: If the control signal rst is _1‘, clk2 is _1‘ and the rd is _1‘ then the read operation takes place.

Wr: If the control signal rst is _1‘, clk1 is _1‘and the wr is _1‘then the write operation takes place

Data_in [6:0]: This control signal indicates the data to be transmitted from the MCU to the Subs. The data can be observed on the bus channel 1.

Ram1 [0:15]: It is a part of a FIFO and it contains 16 memory locations. The data is placed in each of the memory location.

Wptr [4:0]: This control signal points out the memory location where the data can be written.

Rptr [4:0]: This control signal points out the memory location where the data can be read.

Wfull: If this control signal is _1‘ then the FIFO is full.

Rempty: If this control signal is _1‘ then the FIFO is empty.

E. ASM chart of UART Transmitter:

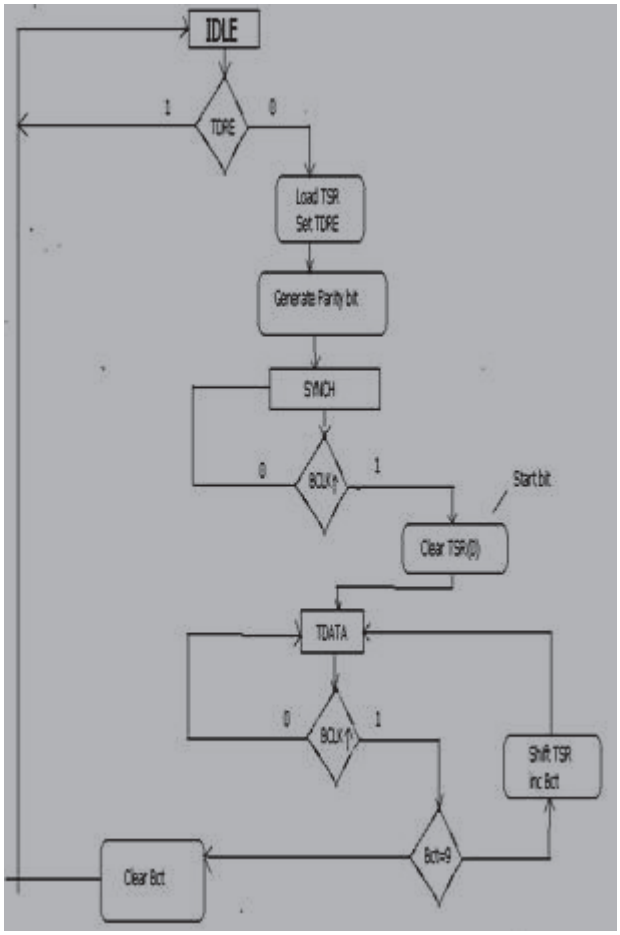

Fig.5: ASM chart of UART Transmitter

The UART circuit block consists of three parts Transmit circuit, Receive circuit and Control/Status registers. The transmit circuit consists of a transmit buffer and a shift register. If rst_n=1,wr=1 and txrdy=1 then tx_sts=1.

If transmitter is ready, then we need to load data from data line to data buffer register. A parity bit is also generated in order to check whether the received bits are correct or incorrect. If TBR (Transmission Buffer Register) contains data then a parity bit is generated by Exclusive operation of all the bits in TBR[3]. A start bit, data bits, a parity bit and a stop bit are loaded into TSR. For every four clock pulses, a baud clock is generated. Each baud clock pulse is used to transmit the data in TSR bit by bit.

The source code which is used to describe the UART transmitter as shown in Fig.5 consists of

DOI: 10.32377/cvrjst1017

(a) A module indicating the transmitter status.

(b) A module which is used for loading the data from data line to TBR.

(c) A module which is used for generating a parity bit..

(d) A module which is used for generating a baud clock.

(e) A module which is used to shift the bits in TBR to TSR bit by bit.

F. Logic Involved in UART TRANSMITTER

The UART circuit block 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. When a control signal tx_sts=1 then the transmitter is busy. The TBR can load the data of 7 bits. When the control signal txrdy=1, then we need to load data from data line to data buffer register . A parity bit is generated, whenever data is present in TBR. The logic involved for generating a parity bit is —the bits in TBR are XORed [4].

Parity bit = tbr(0) XOR tbr(1) XOR tbr(2) XOR tbr(3) XOR tbr(4) XOR tbr(5) XOR tbr(6)

For every four clock pulses a baud clock is generated. For each baud clock a bit is transmitted from TSR to the receiver circuit. The TSR can load the data of 10 bits. In the TSR, the first bit indicates start bit, the ninth bit indicates the parity bit, and the tenth bit indicates the stop bit and from the second bit to the eight bit indicates the bits from TBR. When TSR contains no data then txrdy=1 which indicates that the entire data has been sent to the receiver circuit.

G. Simulation Result of UART Transmitter

The simulation result of UART transmitter is shown in Fig.6. The UART block consists of a transmitter section and receiver section. The transmitter consists of TBR and TSR (Receiver Shift Register). The receiver section consists of RBR (Receiver Buffer Register) and RSR. The TBR consists of 8-bits of data. The TSR consists of start bit, stop bit, parity bit and 8-bits of data. The bits in TSR are shifted to RSR bit by bit. The data bits in RSR are shifted to RBR.

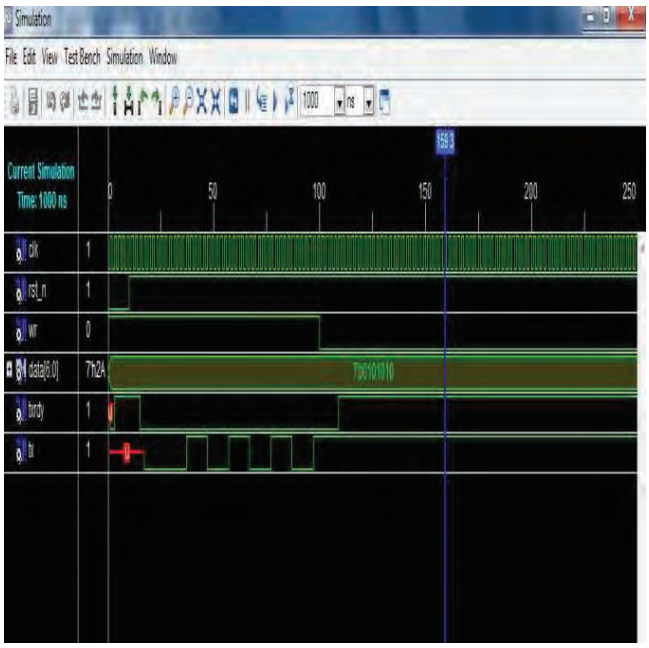

Fig.6: Simulation result of UART Transmitter

Txrdy: If this control signal is _1‘ then TSR has finished in sending the data. If this control signal is _0‘ then TSR has data to be sent.

Tx: This control signal indicates the data that is to be transmitted to the sub-equipments. The data is transmitted to the sub-equipments at different baud rates.

Wr: It is a control signal which indicates the write operation.

Rst_n: This is a control signal which indicates the reset operation.

H. ASM Chart of UART receiver

The ASM chart of UART receiver is shown in Fig.7.

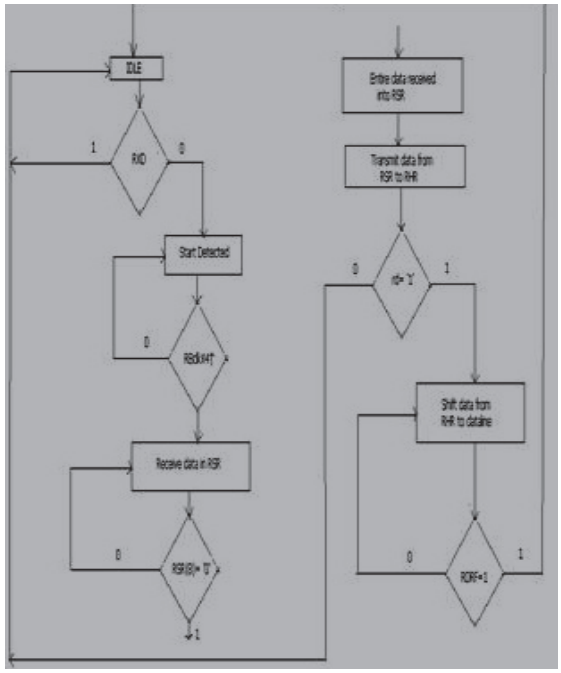

Fig.7: ASM chart of UART receiver

The UART circuit block consists of three parts receive circuit; transmit circuit and control/status registers. The receive circuit consists of a receive shift register and a receive buffer. If a start bit is received into RSR then the control signal det_rx is enabled as shown in Fig.7. The bits in the TSR(Transmitter Shift Register) are shifted bit by bit into the RSR(Receiver Shift Register). If the start bit occupies the first position or first bit of RSR then all the bits are received into receiver. The RSR is reset i.e., RSR contains all 1s. For every four clock pulses, a baud clock is generated. Each baud clock is used for receiving one bit. If the first position of RSR contains _0‘ then the data bits in RSR are shifted into the RHR(Receiver Hold Register). If the read, rd control signal is high then the data in RHR is shifted to data line. The status of the receiver whether it is ready or not can be determined by the control signal rxrdy. If the first position of RSR contains _0‘ then the receiver is ready. If the above condition is not satisfied the receiver is not ready.

I. Logic Involved in UART RECEIVER

The UART circuit block consists of three parts Receive circuit, Transmit circuit and Control/Status Registers. 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. If the start bit _0‘ is received, det_rx control signal is enabled. If the start bit _0‘ occupies the first position of RSR then all the bits are received into the receiver. For each four clock pulses a baud clock is generated. Based on the baud clock the bits are received from the transmitter circuit. The bits in RSR are shifted to the RHR except the first and the last bits. When the control signal read (rd=‘1‘), the data is shifted from RHR to data line.

J. Simulation result of UART receiver

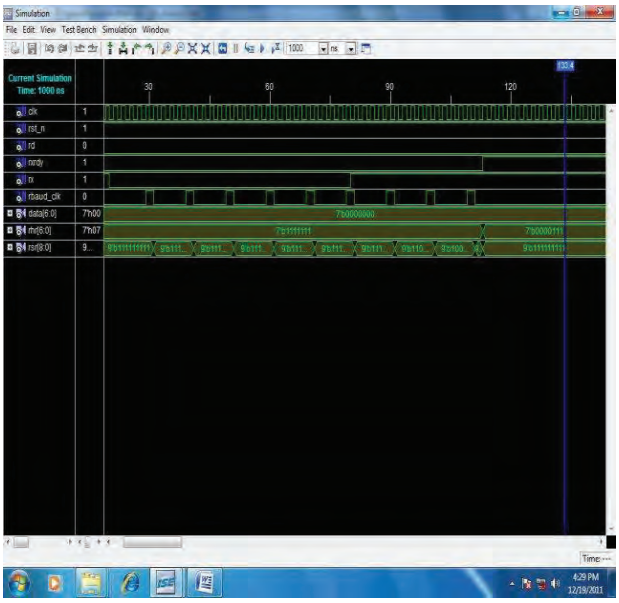

Fig.8: Simulation result of UART receiver

Clock: When the control signal clock is _1 then the receiver is ready to accept the data.

Rst_n: When the control signal rst_n is _0 no operation takes place and when it is _1‘ the data receiving operation takes place.

Rd: When the control signal rd is _1 then the data in

RHR can be read into the data line as shown in Fig.8.

Rxrdy: This control signal indicates whether the receiver is ready or not.

Rx: This control signal indicates the data to be received from the data line to the RSR.

Rbaud_clk: When this control signal is _1‘ then the data is received from the transmitter line to receiver i.e., to RSR as shown in Fig.8.

Data [6:0]: This control signal indicates the input data.

Rhr [6:0]: This control signal indicates the data in the Receiver Hold Register.

Rsr [8:0]: This control signal indicates the start bit, the data bits and the stop bit.

III. IMPLEMENTATION OF A M-CHANNEL UART CONTROLLER

In the multi-channel UART controller, there are various blocks which includes UART block, status detectors, asynchronous FIFOs block and baud rate generator block. All the blocks in the controller has its own function. The main part is UART circuit block and its structure is shown in figure. The communication between any two devices is broadly classified into two types i.e., the serial communication and the parallel communication. In serial communication, the data bits are transferred bit by bit whereas in parallel communication, all the bits are transferred at a time. Parallel communication requires less amount of time and serial communication requires more amount of time for transferring the data. The cost of parallel communication is more compare to the serial communication. If the communication is within the system then the parallel communication is preferred. Suppose, if the communication is between any two systems separated by a great distance then the serial communication is preferred. The UART block is used for converting serial data to parallel data and vice-versa. Asynchronous communication between any two systems will occur when the two systems are operating with different clock domains.

In asynchronous communication, a start bit, a stop bit, a parity bit and the data bits are used. The start bit indicates the start of communication between any two systems. After the start bit, the data or message bits are sent. For detecting the error, a parity bit is sent from the transmitter section to the receiver section. The parity bit

 DOI: 10.32377/cvrjst1017

may be an even parity or odd parity. The stop bit indicates the end of communication between any two systems.

A.UART Architecture

The main functional blocks of an UART as shown in below Fig.9 are:

CPU Bus Controller

Baud rate generator

Receiver /Transmitter

CPU Bus Controller

A bus is a group of wires. The bus is classified into three types. They are control bus, data bus and address bus. The bus controller controls these buses. The control bus is used for transmitting the control signals like bus request, bus grant, hold request, and hold acknowledge etc. The data bus carries the data bits from one block to the another block. The address bus carries the addresses of various memory locations in the memory.

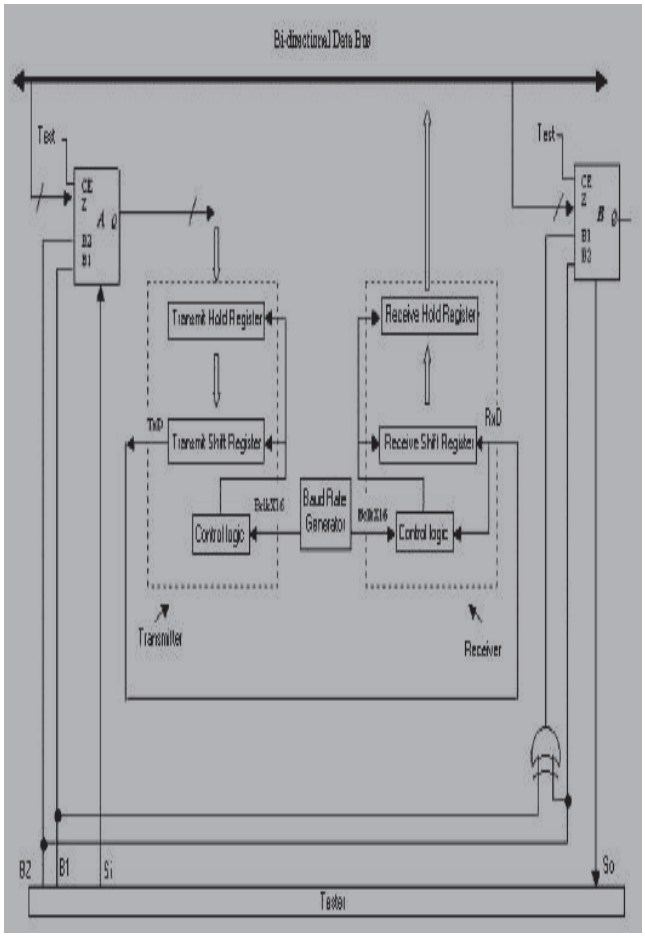

Fig.9:UART block diagram

Baud rate generator

The purpose of the baud rate generator is to generate the baud rate signals. The baud rate indicates the number of bits per second i.e., the transmitter section can send and the receiver section can receive. In UART there are two main blocks i.e., the transmitter section and the receiver section. The baud rate generator is connected to the transmitter section as well as to the receiver section. Based on the baud rate, the transmitter section sends the bits to the receiver section and the receiver section receives the bits from the transmitter section. By suitable selection of the baud rate, the bit loss can be overcome between the transmitter and the receiver section.

Receiver /Transmitter

The transmitter section of an UART block consists of a Transmitter Hold Register (THR) and Transmitter Shift Register (TSR). The receiver section of an UART block consists of a Receiver Hold Register (RHR) and Receiver Shift Register (RSR).The data on the bi-directional data bus is loaded into the THR of the transmitter section of UART.The data bits in THR are loaded into TSR and a start bit, a stop bit and a parity bit are added. These bits are shifted to RSR bit by bit. The RSR contains the data bits, the start bit, the stop bit and the parity bit. Only, the data bits in RSR are loaded into the RHR. The RHR contains the data or message bits.

B.ASM chart

The ASM chart of multi-channel UART controller is shown in Fig.10.The bits are loaded from MCU to RSR. The bits in RSR are loaded to RBR. Here, the RBR contains only the data bits. The bits in RBR are shifted to TBR and the bits in TBR are shifted to TSR. The bits are transmitted to the sub-equipments through the TxD pin. The FIFO is in full condition, when all the bits in wr _ptr are equal to the bits in rd_ptr except the MSB bit of wr_ptr.If all the bits in rd _ptr are equal to all the bits of wr_ptr, then the FIFO is in empty condition.

The UART consists of transmitter section and receiver section. The transmitter section consists of TBR (Transmit Buffer Register) and TSR (Transmit Shift Register).The receiver section consists of RBR (Receiver Buffer Register) and RSR (Receive Shift Register).The purpose of the UART is to convert the data from serial to parallel and parallel to serial. The communication between the Master Control Unit (MCU) and the Sub-equipments is performed by using the multi-channel UART controller. If the Master Control Unit (MCU) is sending the data at high baud-rate and the sub-equipments are receiving the data at low baud-rate then there will be the loss of data. This difficulty can be over commend by using an asynchronous FIFO between the Master Control Unit (MCU) and the sub-equipments. The asynchronous FIFO operates at two different clock domains. The data is read from the Master Control Unit (MCU) at one clock domain and the data is written into the sub-equipments at another clock domain. The MCU and the sub-equipments both contain the UART. The transmitter section of the UART in Master Control Unit (MCU) transmits the data to the sub-equipments and the receiver section of the UART in the sub-equipments receives the data. The Transmit Buffer Register (TBR) in the transmitter section of UART consists of data bits. The start bit, stop bit and a parity bit are added to the data bits and loaded into the Transmit Shift Register (TSR).The bits in Transmit Shift Register (TSR) are transferred into the Receiver Shift Register (RSR) of the receiver section of the

UART in the sub-equipments. Only the data bits in the Receiver Shift Register (RSR) are transferred into the Receiver Buffer Register (RBR) of the receiver section of the UART in the receiver section of the sub-equipments. The communication between the Master Control Unit (MCU) and the sub-equipments is described in the form of ASM chart as shown in Fig.10.

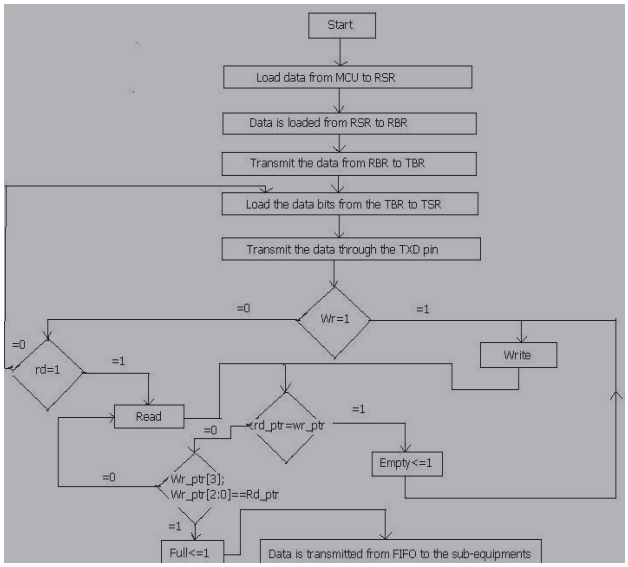

Fig.10.ASM chart of multi-channel UART Controller.

C. Block Diagram

The block diagram of a multi-channel UART controller is shown in the below figure. The communication between the Master Control Unit (MCU) and the sub-equipments is done through the FIFO as shown in Fig.11. If the baud rate of Master Control Unit (MCU) and the sub-equipments are different to each other, then there is a possibility of data loss. This can be overcome by using a FIFO in between the MCU and the sub-equipments. The bus channel 1 is used to transmit the data from MCU to the FIFO s and the bus channel 21, bus channel 22, and the bus channel 23 is used to transmit the data from the FIFO s to the sub-equipments. The UART block is used for converting serial to parallel and parallel to serial data

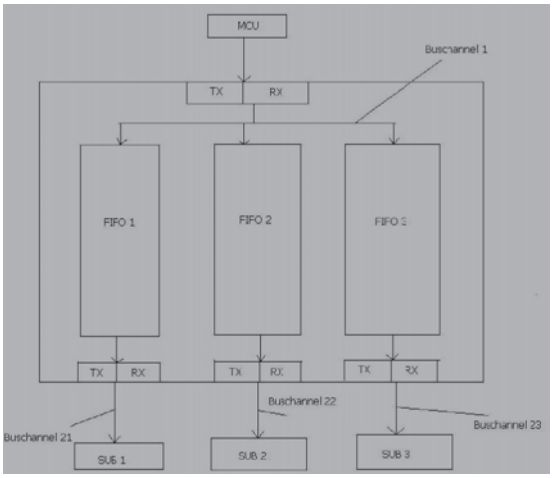

Fig.11.Block Diagram of Multichannel UART Controller.

D. Simulation Result

The simulation result of the top module is shown in the below figure.12. A multichannel UART controller is used for transmitting the data between the master equipment and the three sub-equipments. If the master equipment is sending the data at one baud rate and the three sub-equipments are receiving the data at different baud rates then there will be loss of data. To overcome this difficulty, we are using a FIFO in the controller. A FIFO can store the data and the data can be used latter by the sub-equipments. The structure of the controller consists of UART block, baud rate generator, status detector, FIFO and status buffer. The structure of UART block consists of Transmit Buffer Register (TBR), Transmit Shift Register (TSR), Receiver Buffer Register (RBR), and Receiver Shift Register (RSR).In the simulation description the input data exists on data_in and the output data exists on data_out[5]. The three sub-equipments will receive the data at different baud rates. This can be observed in the simulation diagram where they are labeled as tx. The baud rates of the three transmitters are also different[6]. The simulation result of the controller can be observed below.

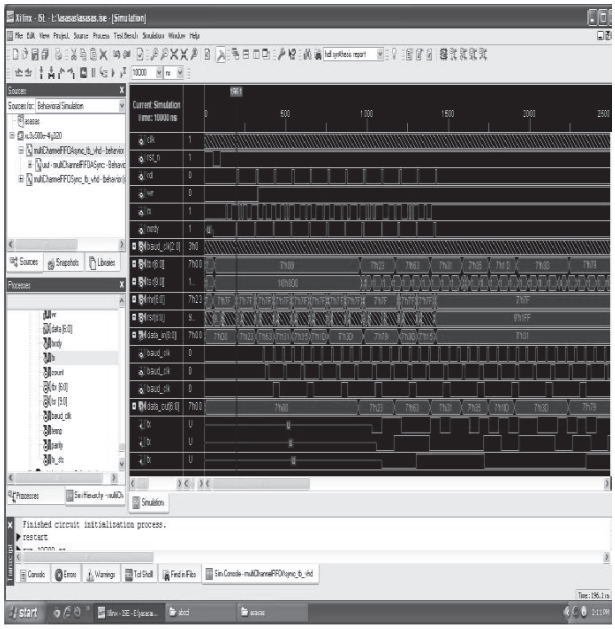

Fig.12.Simulation result of multichannel UART controller (Top module**).**

IV. CONCLUSIONS

This project introduces a method to design an asynchronous FIFO based on FPGA and using an asynchronous FIFO technique implements a multichannel UART controller within FPGA. 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. The controller is reconfigurable and scalable.

 DOI: 10.32377/cvrjst1017

REFERENCES

[1]. Clifford E. Cummings and Peter Alfke, "Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons," SNUG 2002 (Synopsys Users Group Conference, San Jose, CA, 2002) User Papers, March 2002, Section TB2, 3rd paper. Also available at www.sunburst-design.com/papers.

[2]. Stephen Brown and Zvonoko Vranesic "Fundamentals of Digital Logic Design with VHDL" Second Edition , Tata McGraw-Hill Education,pp:310-450,July 2002.

[3]. John M. YarBrough "Digital Logic Applications & Design"Thomson,2002.

[4]. Floyd & Jain "Digital Fundamentals" Eight Edition,2014.

[5]. Douglas L. Perry "VHDL programming by example "Fourth Edition, pp: 68-94, Mcgraw-Hill Book Comp., 1991.

[6]. Jayaram.Bhaskar "A VHDL primer" Third Edition, Third Edition,pp:98-124.May1,2013.